# SAMPLE, DON'T SEARCH: ONE-RUN ENSEMBLES FOR NODE PROPERTY PREDICTION

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Node property prediction often requires tuning graph propagation and model choices for each dataset. We introduce GraphPack, which explores these choices through graph-view sampling and ensemble selection in a single training run. A shared bank of propagated features and training-label views supplies different inputs to randomly configured MLPs. These MLPs train together as one packed module on a single GPU. Validation-based selection combines their snapshots with gradient-boosted trees, adapting the model and view mix to each graph under fixed sampling rules. On ten GraphLand datasets, GraphPack achieves the best or tied performance on seven against tree-based, tuned decoupled, and tuned message-passing pipelines with access to the same inputs. It also achieves higher mean scores on all ten datasets than an ensemble of the best configuration from a 50-trial SIGN search. That search-and-ensemble workflow takes 9.1 times the wall-clock time of GraphPack, measured by the geometric mean across datasets. Controlled ablations across six datasets show that distributing graph views across members improves over giving every member all views. These results identify graph-view diversity as a source of ensemble gains and support sampling and selection as a practical alternative to per-dataset tuning.

## 1 INTRODUCTION

Node property prediction involves graph-dependent choices. These include the number of aggregation hops, the normalization, whether to treat edge directions separately, and whether to use neighbors' training labels as inputs. The choice of model also matters: a message-passing GNN, a decoupled MLP, or a gradient-boosted tree may best suit the node features. The right choices change from graph to graph (You et al., 2020; Bazhenov et al., 2025). Recent re-evaluations find that carefully tuned classic models remain the strongest methods (Bazhenov et al., 2025; Luo et al., 2024), and no model family ranks first across GraphLand's industrial graphs. Per-dataset tuning is a common route to strong performance, but tuning one GNN can take hours to days of GPU time (Platonov et al., 2026).

Automated machine learning addresses these choices through automated search, often followed by ensembling the models trained during the search (Feurer et al., 2015; Erickson et al., 2025). Ensemble selection from a library of randomly configured models dates back to Caruana et al. (2004). On tabular data, TabPack (Gorishniy et al., 2026) runs such random search and selection as one packed job. It trains a pack of MLPs with randomly sampled hyperparameters as a single batched module and selects an ensemble from their checkpoints, matching tuned baselines in one run. Graphs introduce another source of variation: the propagation views determine which neighborhood information each member receives. This raises a question beyond hyperparameter tuning: can members with different graph views form a stronger ensemble than copies of one selected configuration?

We show that on graphs with tabular node features one run of sampling followed by selection can replace the search, provided the sampling covers structure as well as model family. As Figure 2 shows, GraphPack precomputes one bank of propagated feature views. Each of 64 MLP members draws its own subset of views and training hyperparameters. The view choices cover operator, hop budget, high-pass and min/max aggregations, edge direction, and a budget of diffused training labels with self-label exclusion. All members train as one packed module on one GPU, while LightGBM candidates train on the same bank in the background. Every epoch, greedy selection on the vali-



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.

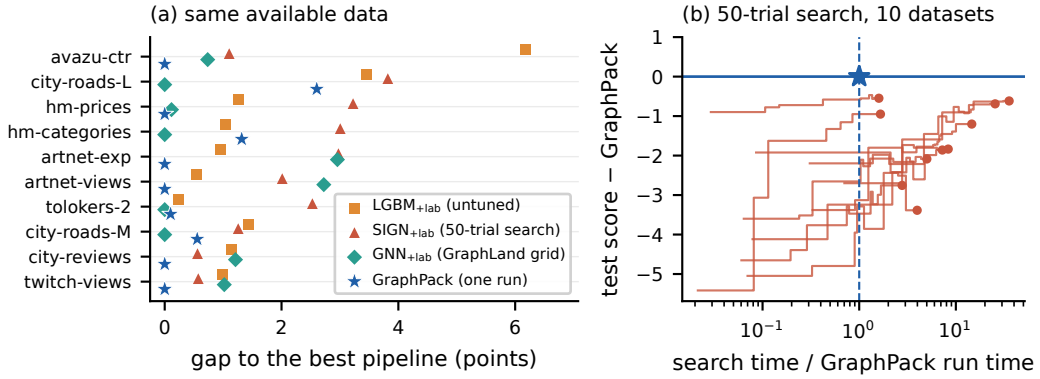


Figure 1: Sampling instead of searching with the same available data. (a) Score gap from each pipeline to the best of the four on the ten GraphLand datasets under random splits with 50% training labels (Table 2; means over three runs, ten for daggered published results). All four receive the same features, graph, and training labels, except where $\text{GNN}_{+\text{lab}}$ is GraphLand's higher label-free result (dagger in Table 2). GraphPack is best or tied on seven of ten graphs. (b) Test score throughout a 50-trial search over a label-equipped SIGN, with one curve per dataset. At each point, the score is for the configuration with the best validation score so far. Scores are relative to the mean of three GraphPack runs (horizontal line at 0). Wall-clock time is normalized by the mean GraphPack run time (vertical line at 1).

dation set rebuilds the ensemble from snapshots and candidates. Building on random configuration sampling and ensemble selection (Caruana et al., 2004; Zaidi et al., 2021), GraphPack organizes graph views into a shared bank that supports both member-specific inputs and a mixed neural–tree candidate pool. This design explores propagation choices within one training run and lets selection adapt their combination to the graph.

Figure 1 summarizes the evidence on the ten GraphLand graphs with all three baseline runs. With the same available data, GraphPack is best or tied on seven of ten graphs against three strong pipelines: an untuned LightGBM on the same bank, a 50-trial-tuned SIGN with label views, and GraphLand's own message-passing GNNs tuned on its grid and trained with the label trick. Its average gap to the best pipeline on each graph is the smallest (0.46 points), while each of the three falls 3.0–6.2 points behind on some graph. The 50-trial SIGN search costs 1.6–36× the wall-clock of the entire GraphPack run. Ensembling the tuned configuration improves its score on nine of ten datasets, while GraphPack remains ahead on all ten. Controlled ablations show that splitting the bank's views across members beats giving every member all views in 30 of 30 dataset–seed pairs. A deep ensemble of the best sampled configuration trails the sampled ensemble in 29 of 30. The selected view mix also varies with the graph: the share of members using label views tracks their measured benefit. Further evaluations cover temporal shift, heterophily, and a held-out 2.9M-node graph. Section 4.4 and Appendix D report these evaluations.

**Contributions.**

- **One-run sample-then-select for graphs.** A shared propagation bank supports a packed ensemble of MLPs with individually sampled graph views and training hyperparameters. Online ensemble selection draws from a mixed pool of MLP snapshots and GBDTs.
- **Comparison with the same available data.** Label-equipped GBDT, tuned decoupled, and tuned message-passing pipelines receive the same inputs. The evaluation reports gaps to the best pipeline on each graph, includes a tune-then-ensemble baseline, and accounts for search cost. Further graphs up to 1.63M nodes and a held-out 2.9M-node graph assess scale and performance across graph regimes.
- **Evidence for structural diversity and adaptive selection.** Controlled ablations show gains from distributing graph views across members, while composition analysis links selected label-view use to its measured benefit.

## 2 Related Work

**Decoupled and fixed-aggregation models.** SGC and SIGN precompute propagation under one or several operators and hop depths (Wu et al., 2019; Frasca et al., 2020). SAGN, GAMLP, FSGNN, and GPR-GNN learn how to weight hops or filters (Sun et al., 2021; Zhang et al., 2022b; Maurya et al., 2021; Chien et al., 2021). NARS combines propagated features of random relation subsets inside one model (Yu et al., 2020). Fixed aggregation features with a tuned MLP rival GNNs on most benchmarks (Rubio-Madrigal & Burkholz, 2026). PaSca searches this decoupled design space for one architecture per dataset (Zhang et al., 2022a), and graph NAS and design-space studies show that the best design varies across tasks (Gao et al., 2020; You et al., 2020). GraphPack keeps the design space but samples a different configuration for every ensemble member and selects among members afterwards, in place of one searched model with learned mixing.

**Labels as inputs.** Label propagation and C&S propagate training labels to predictions (Zhu et al., 2003; Huang et al., 2021), and SAGN and GAMLP reuse propagated labels as input features (Sun et al., 2021; Zhang et al., 2022b). The label trick feeds a random part of the training labels as input and predicts the rest (Wang et al., 2022; Shi et al., 2021). GraphCleaner removes self-label return paths by zeroing the diagonal of the propagation operator (Li et al., 2023). We adopt this rule. Our label views also separate the neighbor-label mean from its coverage, and their budget is sampled so that selection can drop them where they do not help on validation.

**Ensembles, ensemble search, and AutoML.** Snapshot, batched, and packed ensembles cut the cost of deep ensembles (Lakshminarayanan et al., 2017; Huang et al., 2017; Wen et al., 2020; Laurent et al., 2023), and diversity can come from hyperparameters (Wenzel et al., 2020), architectures (Zaidi et al., 2021), or random feature subspaces (Ho, 1998). AutoML systems ensemble the models their search trained (Caruana et al., 2004; Feurer et al., 2015; Erickson et al., 2025), and TabM and TabPack pack MLP ensembles for tabular data (Gorishniy et al., 2025; 2026). On graphs, architecture diversity ranks above hyperparameter diversity for GNN ensembles (Deng et al., 2025a). TAG and AutoHEnsGNN select ensembles of foundation-model or GNN members (Hayler et al., 2025; Xu et al., 2022). GBDT–GNN hybrids target tabular-feature graphs (Ivanov & Prokhorenkova, 2021; Chen et al., 2024). GraphPack brings member-specific graph views into packed ensemble training and combines MLP snapshots with GBDTs on a shared propagation bank. Its controlled ablations examine how structural sampling affects ensemble accuracy and error diversity.

**Graph foundation models.** G2T-FM (Eremeev et al., 2025), GraphPFN (Eremeev et al., 2026), GraphAny (Zhao et al., 2025), and TAG (Hayler et al., 2025) transfer pretrained tabular or graph priors to new graphs. A recent fair evaluation on GraphLand's low-label split finds PFN-based models ahead of 100-trial-tuned GNNs, with fine-tuned GraphPFN performing best. It also notes that none of the evaluated models handles million-node graphs (Platonov et al., 2026). GraphPack is complementary: it trains from scratch in one run and scales to million-node graphs. Pretrained predictions could join its candidate pool.

## 3 Method

**Setup and principle.** We consider transductive node classification and regression on a graph $G = (V, E)$ with $n$ nodes, node features $\mathbf{X} \in \mathbb{R}^{n \times d}$, and targets observed on a training set $V_{\mathrm{tr}}$. Validation and test sets $V_{\mathrm{val}}$ and $V_{\mathrm{te}}$ serve model selection and final evaluation. A conventional workflow searches for a configuration $c = (c^{\mathrm{str}}, c^{\mathrm{trn}})$ that combines structural choices (propagation operator and depth, direction handling, label reuse, model family) with training choices (learning rate, weight decay, dropout, depth). It retrains each candidate and keeps the one with the best validation score. GraphPack represents these choices in a fixed configuration space and uses *sample-then-select*. Every member of one packed ensemble draws its own $c$, and all members train once. Validation-based greedy selection chooses which members and epochs form the output. Figure 2 shows the four components.

### 3.1 Propagation bank

Following SGC and SIGN (Wu et al., 2019; Frasca et al., 2020), all graph computation happens once, before training, and training then uses i.i.d. node mini-batches. With $\mathbf{A}$ the symmetrized adjacency,

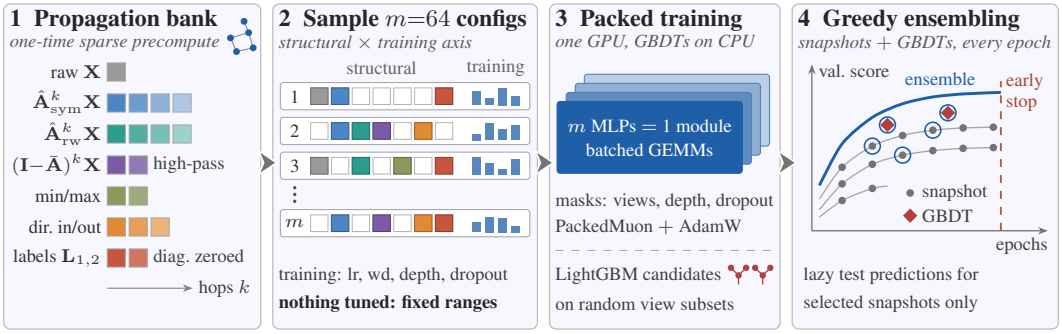


Figure 2: GraphPack overview. (1) A one-time CPU pass builds a bank of multi-hop, multi-operator, direction-aware, and diffused-label views with self-label exclusion. (2) Each of $m$ base models samples a view subset and training hyperparameters. (3) All models train as one packed module while GBDT candidates train on CPU. (4) Every epoch, greedy selection rebuilds the ensemble from snapshots and candidates, and ensemble-level early stopping ends the run.

$\hat{\mathbf{A}}_{\text{sym}} = \tilde{\mathbf{D}}^{-1/2}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-1/2}$ ($\tilde{\mathbf{A}} = \mathbf{A}+\mathbf{I}$), $\hat{\mathbf{A}}_{\text{rw}} = \mathbf{D}^{-1}\mathbf{A}$, and the self-loop-free $\bar{\mathbf{A}} = \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}$, the bank concatenates

$$\mathbf{B} = \big[\mathbf{X} \,\|\, \{\hat{\mathbf{A}}^k_{\text{sym}}\mathbf{X}\}_{k=1}^{K} \,\|\, \{\hat{\mathbf{A}}^k_{\text{rw}}\mathbf{X}\}_{k=1}^{K} \,\|\, \{(\mathbf{I}-\bar{\mathbf{A}})^k\mathbf{X}\}_{k=1}^{K_{\text{hp}}} \,\|\, \max_{\mathcal{N}}\mathbf{X} \,\|\, \min_{\mathcal{N}}\mathbf{X} \,\|\, \mathbf{L}\big], \quad (1)$$

where high-pass powers carry heterophilous signal and one-hop min/max aggregations add a non-linear neighborhood summary, as in GraphLand's neighborhood feature aggregation (Bazhenov et al., 2025). When the edge set is asymmetric, in- and out-neighborhood views $(\mathbf{D}_{\text{in}}^{-1}\mathbf{A}^\top)^k\mathbf{X}$ and $(\mathbf{D}_{\text{out}}^{-1}\mathbf{A})^k\mathbf{X}$ are added, since separating directions can restore homophily (Rossi et al., 2023). Dataset-specific edge preprocessing is described in Appendix C.

**Diffused-label views.** $\mathbf{L}$ holds propagated training labels. With $\mathbf{Y}$ the train-masked one-hot labels (standardized targets for regression, zeros outside $V_{\text{tr}}$), we use $\mathbf{L}_1 = \bar{\mathbf{A}}\mathbf{Y}$ and $\mathbf{L}_2 = \bar{\mathbf{A}}^2\mathbf{Y} - \text{diag}(\bar{\mathbf{A}}^2)\mathbf{Y}$: zeroing the diagonal of each label operator removes every path that returns a node's own label to it (Li et al., 2023). Each view is divided by its propagated label coverage. The coverage is kept as a separate label-free channel, so a member can distinguish a confident neighbor-label mean from one supported by few labels. Labels of validation and test nodes never enter $\mathbf{Y}$. On temporal splits, training labels come from the earliest period, so coverage can vary across periods. Appendix D.3 examines this variation.

### 3.2 Configuration Space

Table 1 defines two independent axes from which each member draws a configuration. The **structural axis** samples an operator ($\hat{\mathbf{A}}_{\text{sym}}$, $\hat{\mathbf{A}}_{\text{rw}}$, or both), a hop budget $k \le K$ ($k$=0 drops all low-pass views), optional high-pass, min/max, and directed views, and a label budget of 0, 1, or 2 hops. The selected views index a column subset of $\mathbf{B}$, and the remaining columns are masked. The population therefore spans plain MLPs, SIGN-like members, and high-pass, direction-aware, and label-propagation hybrids. The **training axis** follows TabPack and samples depth, dropout, learning rates, and weight decay. These sampling distributions are fixed across datasets and tasks.

### 3.3 Packed Training with a Mixed Candidate Pool

All $m$=64 MLPs are stored as stacked parameters and run as batched matrix multiplications on shared input batches, as in packed ensembles (Laurent et al., 2023; Gorishniy et al., 2026). Per-member masks implement input views, depth, and dropout. A packed optimizer keeps per-member state and learning rates, using Muon (Jordan et al., 2024) for all weight matrices and AdamW for biases. Because selection needs only predictions, a background CPU thread trains four LightGBM models on random view subsets of the same bank concurrently with the GPU pack. They are skipped when a task has more than 32 classes, since multiclass boosting grows one tree per class in each round. They are also skipped when the training block of the bank exceeds $2{\cdot}10^9$ cells.

Table 1: Configuration space of the MLP members (U = uniform, LogU = log-uniform), shared by every dataset, split, and task. GBDT candidate distributions are in Appendix B.

| *Structural axis* | |
|---|---|
| operator | U{$\hat{\mathbf{A}}_{\text{sym}}$, $\hat{\mathbf{A}}_{\text{rw}}$, both}; none if $k$=0 |
| hop budget $k$ | U{0..6} |
| high-pass powers | w.p. 0.4: U{1..3} |
| min/max views | w.p. 0.4 |
| drop raw view | w.p. 0.15 |
| directed views (if graph directed) | w.p. 0.6: U{1..4} hops |
| directed label views (if directed views) | w.p. 0.7 |
| diffused-label budget | $\{0, 1, 2\}$ w.p. $\{.5, .3, .2\}$ |

| *Training axis (following TabPack)* | |
|---|---|
| depth | U{1..3} |
| dropout | 0 w.p. 0.5, else U$[0, 0.5]$ |
| Muon learning rate | LogU$[10^{-3}, 10^{-1}]$ |
| AdamW learning rate (biases) | LogU$[10^{-4}, 5{\cdot}10^{-3}]$ |
| weight decay | LogU$[10^{-3}, 1]$ |
| *Fixed* | |
| $m$ / $m_{\text{ens}}$ / $p_{\text{base}}$ / $p_{\text{ens}}$ | 64 / 32 / 16 / 32 |
| width / GBDT candidates | 384 / 4 (LightGBM, CPU) |

### 3.4 Online greedy selection

After every epoch, each live member contributes its current validation predictions as a candidate, and the pool also holds the previous ensemble's members and finished GBDTs. Greedy forward selection (Caruana et al., 2004) starts from the best candidate. At each step, it adds the candidate that most improves the validation score of the averaged prediction. Selection is without replacement and stops when no candidate improves the score or the ensemble reaches $m_{\text{ens}}$=32 members. Test predictions are computed only for selected snapshots, which bounds memory on million-node graphs. Members stop after 16 epochs without improvement and the run stops after 32 epochs without ensemble improvement, returning the best ensemble as a plain average. The ensemble cap and patience values are shared across datasets. On the development graphs, bank construction takes seconds to 2.4 CPU minutes. One 64-member run costs 1.3–31$\times$ the mean single-model search trial. Full RH runs on the development graphs take 30 seconds (tolokers-2) to 135 minutes (pokec-regions) on one 24 GB GPU.

## 4 Experiments

### 4.1 Setup

**Data.** GraphLand (Bazhenov et al., 2025) provides industrial graphs with tabular node features under official random and temporal splits. We use three splits: RH (random, 50% labels), RL (random, 10%), and TH (temporal). We use eleven development datasets with 12K–1.63M nodes (two pairs share a graph), and the three tasks of GraphLand's 2.9M-node web graph. The metrics are AP for binary classification, accuracy for multiclass classification, and $R^2$ for regression. We add four heterophily benchmarks under their ten official splits (Platonov et al., 2023) and ogbn-arxiv (Hu et al., 2020). GraphPack numbers are means over three seeds (ten splits for the heterophily suite, two runs for the web tasks) with sample standard deviations.

**Development protocol.** We developed the configuration space on the eleven GraphLand datasets, the heterophily suite, and ogbn-arxiv. We evaluated the three web tasks with the same default configuration; Appendix C.2 gives the evaluation protocol.

**Baselines with the same available data.** We compare GraphPack with three pipelines that receive the same node features, graph, and training labels. *LGBM*$_{+\text{lab}}$ is LightGBM with fixed, untuned settings on the full bank. *SIGN*$_{+\text{lab}}$ is a SIGN whose hop depth, label budget, depth, dropout, learning rate, and weight decay are tuned in 50 Optuna trials (Akiba et al., 2019) using our trainer. *GNN*$_{+\text{lab}}$ runs GraphLand's official GNN code and 15-point tuning grid, starting from GraphLand's best family on each dataset. It uses fixed feature transforms and the label trick: in every step, half of the training labels are fed as input and the loss is computed on the other half (Wang et al., 2022). We also compare the no-label variant with GraphLand's published label-free baselines, which use a 15-point grid for GNNs and 100 Optuna trials for GBDTs. Hyperparameters, memory-driven alternatives, and fidelity checks are in Appendix C.3. As in GraphLand, a difference within the summed standard deviations counts as a tie.

Table 2: GraphLand RH split. Left: the best published tuned baseline, which uses no label inputs (Bazhenov et al., 2025), and GraphPack with label views disabled. Right: pipelines that receive the same inputs, including training labels (mean±std over three runs). A dagger marks GraphLand's published label-free GNN result (ten runs), kept where our $\text{GNN}_{+\text{lab}}$ run scored below it. pokec-regions has no corresponding baseline run (Appendix D). Bold: best of the right block, and in the held-out rows GraphPack above the published best. Bottom rows: score gap to the best in the right block.

| | Published | GraphPack | Same inputs, including training labels | | | |
|---|---|---|---|---|---|---|
| Dataset | label-free best | no-lab | $\text{LGBM}_{+\text{lab}}$ | $\text{SIGN}_{+\text{lab}}$ | $\text{GNN}_{+\text{lab}}$ | GraphPack |
| tolokers-2 | 63.76±1.30 GAT | 64.32±1.45 | 64.27±0.17 | 61.99±1.03 | **64.52±0.79 GAT** | 64.42±0.57 |
| city-reviews | 81.63±0.06 LGBM-NFA | 82.28±0.19 | 81.40±0.05 | 81.99±0.02 | 81.34±0.17 GAT | **82.55±0.06** |
| artnet-exp | 50.62±0.35 GAT | 52.07±0.30 | 52.67±0.20 | 50.65±0.23 | 50.67±0.59 GAT | **53.62±0.28** |
| hm-categories | 79.28±0.31 GT | 73.01±0.13 | 82.95±0.11 | 80.98±0.18 | **83.99±0.39 GT** | 82.67±0.59 |
| pokec-regions | 50.06±0.53 GT | 51.83±1.74 | – | – | – | 59.12±0.01 |
| hm-prices | 81.68±0.41 GAT | 80.87±0.23 | 81.84±0.08 | 79.88±0.27 | 82.99±0.41 GAT | **83.10±0.33** |
| avazu-ctr | 35.74±0.19 GAT | 36.49±0.19 | 30.30±0.42 | 35.37±0.17 | 35.74±0.19 GAT† | **36.48±0.20** |
| city-roads-M | 72.09±0.07 LGBM-NFA | 72.54±0.02 | 77.12±0.06 | 77.30±0.02 | **78.55±0.22 GAT** | 78.00±0.09 |
| city-roads-L | 66.24±0.08 LGBM-NFA | 66.19±1.18 | 76.07±0.02 | 75.70±0.05 | **79.52±0.10 GAT** | 76.91±0.10 |
| twitch-views | 77.12±0.11 GCN | 76.91±0.10 | 77.15±0.01 | 77.56±0.04 | 77.12±0.11 GCN† | **78.14±0.34** |
| artnet-views | 61.02±0.13 GCN | 63.15±0.22 | 63.21±0.08 | 61.73±0.09 | 61.02±0.13 GCN† | **63.74±0.19** |
| *Held out: GraphLand web graph, never used during development (mean±std of two runs)* | | | | | | |
| web-fraud | 20.47±0.17 SAGE | – | – | – | – | 19.85±0.28 |
| web-traffic | 85.70±0.08 GAT | – | – | – | – | **86.67±0.01** |
| web-topics | 50.58±0.04 GT | – | – | – | – | **51.14±0.12** |
| Max gap | | | 6.18 | 3.82 | 2.96 | 2.60 |
| Mean gap | | | 1.72 | 2.11 | 0.88 | 0.46 |

### 4.2 Comparison with the same available data

Table 2 shows the pattern of Figure 1(a): each single pipeline falls clearly behind on some graph. $\text{LGBM}_{+\text{lab}}$ falls 6.2 points behind on avazu-ctr. Its score of 30.30 is even below GraphLand's tuned label-free LightGBM with neighborhood feature aggregation (LGBM-NFA; 35.35). $\text{SIGN}_{+\text{lab}}$ falls 3.8 points behind on city-roads-L, and $\text{GNN}_{+\text{lab}}$ 3.0 on artnet-exp. GraphPack is best or tied on seven of the ten graphs, has the smallest mean gap to the per-dataset best (0.46 points), and its maximum gap is 2.60 points. On road networks and hm-categories, message passing with reused labels is stronger: GraphPack trails $\text{GNN}_{+\text{lab}}$ by 2.60 points on city-roads-L, 1.32 on hm-categories, and 0.56 on city-roads-M. Selecting the best of the three pipelines on validation gives a mean gap of 0.64, compared with 0.46 for GraphPack. This selection has a similar maximum gap of 2.53. Validation-based selection chooses the model families, views, and use of labels per graph. The LightGBM candidates contribute to this coverage: without them the MLP pack has a maximum gap of 3.93, comparable to a single pipeline. Adding them improves the score by up to 3.9 points on artnet-exp.

With label views disabled, GraphPack wins five of eleven graphs against the label-free published baselines, ties four, and trails on two. Enabling label views gives the highest mean on all eleven. Label views account for much of the gain over the published baselines on city-roads-L (+10.7), pokec-regions (+9.1), city-roads-M (+5.9), and hm-categories (+3.4). This comparison measures the full workflow against alternatives that also reuse training labels.

### 4.3 Sampling versus searching

$\text{SIGN}_{+\text{lab}}$ is also a cost reference, since its search runs on the same bank, trainer, and GPU. The 50-trial search alone takes 1.6–36× the wall-clock of a complete GraphPack run, from 1.6× on city-reviews to 36× on twitch-views. Its selected configuration, rerun with three seeds, still scores 0.6–3.2 points lower. The best observed *test* score among the 50 single-seed trials also stays below GraphPack on all ten searches. $\text{SIGN}_{+\text{lab}}$ searches a subset of GraphPack's views, trains with AdamW, and has no tree candidates. Appendix D.2 gives the per-dataset search results.

**Tuning followed by ensembling.** We take the validation-best $\text{SIGN}_{+\text{lab}}$ configuration from each 50-trial search and train 64 copies with different initializations, forming a deep ensemble (Lakshminarayanan et al., 2017). This *tune-then-ensemble* baseline uses the same packed trainer and online

Table 3: Sampling versus tuning followed by ensembling on GraphLand RH. Tuned $\text{SIGN}_{+\text{lab}}$: the validation-best configuration of a 50-trial search. Tune-then-ensemble: 64 copies of that configuration with different initializations, combined by online greedy selection. The last two columns use GraphPack's default sampling configuration, with GBDT candidates disabled or enabled. Scores are three-seed means, with sample standard deviations for the ensembles. Bold: highest mean in each row. Search and ensemble protocols are detailed in Appendix C.4.

| | Tuned | Tune-then- | GraphPack | |
|---|---|---|---|---|
| Dataset | $\text{SIGN}_{+\text{lab}}$ | ensemble | no GBDT | full |
| tolokers-2 | 61.99 | 61.92±0.26 | 63.36±0.68 | **64.42±0.57** |
| city-reviews | 81.99 | 82.15±0.04 | 82.05±0.11 | **82.55±0.06** |
| artnet-exp | 50.65 | 50.73±0.28 | 49.70±0.49 | **53.62±0.28** |
| hm-categories | 80.98 | 81.17±0.12 | 80.69±0.40 | **82.67±0.59** |
| hm-prices | 79.88 | 81.08±0.04 | 81.38±0.15 | **83.10±0.33** |
| avazu-ctr | 35.37 | 36.04±0.03 | 36.22±0.05 | **36.48±0.20** |
| city-roads-M | 77.30 | 77.53±0.00 | 77.80±0.14 | **78.00±0.09** |
| city-roads-L | 75.70 | 76.06±0.02 | 76.78±0.05 | **76.91±0.10** |
| twitch-views | 77.56 | 77.92±0.03 | 78.07±0.37 | **78.14±0.34** |
| artnet-views | 61.73 | 62.15±0.01 | 62.85±0.05 | **63.74±0.19** |

greedy selection as GraphPack, with AdamW as in the search. It follows the tune-then-select ensemble workflow (Zaidi et al., 2021). Table 3 compares the single tuned model, its ensemble, and GraphPack with and without tree candidates.

Ensembling improves the tuned SIGN on nine of ten datasets by mean, but full GraphPack remains ahead on all ten, by 0.22–2.89 points (nine wins, one tie, and zero losses under the tie rule). The search followed by packed ensemble training takes $9.1\times$ the wall-clock of one complete GraphPack run, as a geometric mean over datasets. The advantage also extends to the neural-only pack: without GBDT candidates, GraphPack is ahead on seven of ten by mean (six wins, three ties, and one loss). Tree candidates account for part of the full method's advantage, particularly on artnet-exp and hm-categories. Section 5 isolates graph-view sampling within a common bank, trainer, and selection procedure.

### 4.4 Held-out graph, scale, and temporal shift

**Held-out web graph.** We report the mean of two runs of the frozen configuration per task (4.1, 5.0, and 3.0 hours per run). Table 14 shows that GraphPack reaches $19.85 \pm 0.28$ AP on web-fraud against $20.47 \pm 0.17$ for GraphLand's tuned GraphSAGE (a loss), $86.67 \pm 0.01$ $R^2$ on web-traffic against $85.70 \pm 0.08$ for GAT (a win), and $51.14 \pm 0.12$ accuracy on web-topics against $50.58 \pm 0.04$ for GT (a win) under the tie rule. At 2.9M nodes the memory rule of Section 3.3 skipped the GBDT candidates, so these runs evaluate the MLP pack alone.

**Scale and temporal shift.** On the 1.63M-node pokec-regions, GraphPack scores 59.12 against the best published 50.06 in 2.2 hours, and 51.83 without label views. On the temporal split, where training, validation, and test nodes come from successive periods, GraphPack has the best average rank on the eight of our graphs with a temporal split (1.5, next GAT 2.8). It records five wins, two ties, and one loss, including a margin of +12.5 on pokec-regions. Against the same label-free baselines, disabling label views separates the contribution of the remaining components: GraphPack still wins three of the seven temporal graphs we reran (all but pokec-regions), ties three, and loses one. On hm-prices the mean without label views is higher (70.70 vs. 68.49, counted as a tie under GraphLand's rule).

Appendix D reports the low-label and heterophily comparisons, including settings where fine-tuned foundation models or deep message passing perform better.

Table 4: What sampling buys (test score, mean±std over seeds, GBDT candidates off). Best in row in bold. Last rows: wins, ties, and losses of *Both* under the tie rule, and the dataset–seed pairs in which *Both* is ahead.

| Dataset | Both | Structure | Train.-HP | All views | Sel. struct. | Tuned ens. | Init |
|---|---|---|---|---|---|---|---|
| amazon-ratings | 56.01±0.27 | **56.24±0.43** | 52.96±0.87 | 54.21±0.77 | 55.85±0.52 | 55.45±0.42 | 53.01±0.63 |
| artnet-exp | **49.69±0.39** | 49.16±0.41 | 48.62±0.15 | 45.36±0.54 | 48.96±0.51 | 48.25±0.29 | 47.33±0.38 |
| city-roads-M | **77.77±0.12** | 77.19±0.11 | 76.84±0.07 | 77.37±0.04 | 77.72±0.18 | 77.27±0.16 | 76.18±0.07 |
| hm-categories | 81.14±0.56 | **81.72±0.56** | 80.57±0.31 | 78.32±0.18 | 80.62±1.42 | 80.52±0.97 | 80.27±0.09 |
| roman-empire | **85.44±0.57** | 85.23±0.29 | 84.81±0.62 | 84.07±0.22 | 85.05±0.24 | 85.08±0.61 | 84.41±0.41 |
| tolokers-2 | **63.05±0.65** | 62.45±0.45 | 61.95±0.55 | 60.31±0.64 | 61.68±0.75 | 61.55±0.88 | 61.75±0.86 |
| *Both* W/T/L | – | 1/5/0 | 3/3/0 | 6/0/0 | 0/6/0 | 2/4/0 | 5/1/0 |
| *Both* ahead (pairs) | – | 19/30 | 29/30 | 30/30 | 23/30 | 29/30 | 30/30 |

## 5 What Sampling Buys on Graphs

We examine how graph-view sampling contributes to ensemble accuracy using seven arms that share the pack, trainer, and selection. They differ only in how member configurations are drawn. GBDT candidates are disabled, and the study covers six datasets. Appendix C.5 specifies the arms. *Both* samples structure and training hyperparameters as in GraphPack. *Structure* and *Training-HP* sample one axis with the other frozen. The frozen structure is a SIGN-style view set, and *Init* is a deep ensemble of that SIGN-style member. Three controls isolate the role of views: *All views* gives every member the entire bank. *Selected structure* fixes every member's view set to that of the member with the best validation score in a *Both* run. *Tuned ensemble* also fixes every member's hyperparameters to those of that selected member, following the identical-configuration control of TabPack (Gorishniy et al., 2026). Paired comparisons match dataset and seed. Table 4 reports per-dataset means and win/tie/loss counts, with paired-run counts as a measure of consistency across seeds.

**Sampling views finds structure and supplies diversity.** Structural sampling has a higher mean score than hyperparameter sampling on all six datasets. The gain reaches 3.3 points on amazon-ratings, with three wins and three ties under the tie rule across the datasets. Structural sampling is ahead in 26 of 30 dataset–seed pairs. Giving every member all views is worse than *Both* in 30 of 30 pairs, by 4.3 points on artnet-exp and 2.8 on hm-categories. Thus, partitioning the views across members beats concatenating them, as in the random subspace method (Ho, 1998). Sampling also beats copying a good configuration. A deep ensemble of the validation-selected configuration trails *Both* in 29 of 30 pairs (ties on four of six datasets). Here the frozen configuration is selected from the 64 sampled members; Table 3 separately evaluates an ensemble built after a 50-trial search. Freezing every member at the validation-selected view set and sampling only hyperparameters is counted as tied with *Both* on all six datasets, but *Both* is ahead in 23 of 30 dataset–seed pairs. The small, consistent gain over a selected view set complements the larger gains over giving every member all views: sampling finds useful structures and distributes them across the ensemble.

**Weaker best members, stronger ensembles.** Against the *Training-HP* arm (Figure 3(a)), the best single member of the *Structure* arm is worse on all six datasets, by up to 3.5 points. Yet its selected ensemble scores higher on four of the six datasets on validation and on all six on test. Across all pairs of members in a *Both* pack, Figure 3(b) shows that the correlation of their test residuals falls with structural distance. This distance is the Jaccard distance between their view sets. The association persists in a joint regression that accounts for hyperparameter distance, label use, and member loss (Appendix E.1). This pattern supports error diversity as one contributor to the ensemble gains.

**Selection adapts the composition.** Figure 4 shows how the selected model and view mix changes across datasets under the same sampling rules. Within GraphLand, a given seed draws the same 64 MLP configurations on every graph. Validation-based selection therefore adapts the MLP composition from a common sampled pool. The fraction of selected members that use label views, including GBDT members, tracks the measured benefit of labels across the eleven GraphLand datasets (Spearman $\rho = 0.86$, one-sided permutation $p < 0.001$). The benefit is the score difference between full GraphPack and its no-label variant. On city-roads-L, where labels add 10.7 points, 0.92 of selected members use them. On avazu-ctr, where they add nothing, that share is 0.42. Thus, selection retains label views more often on datasets where their measured benefit is larger.

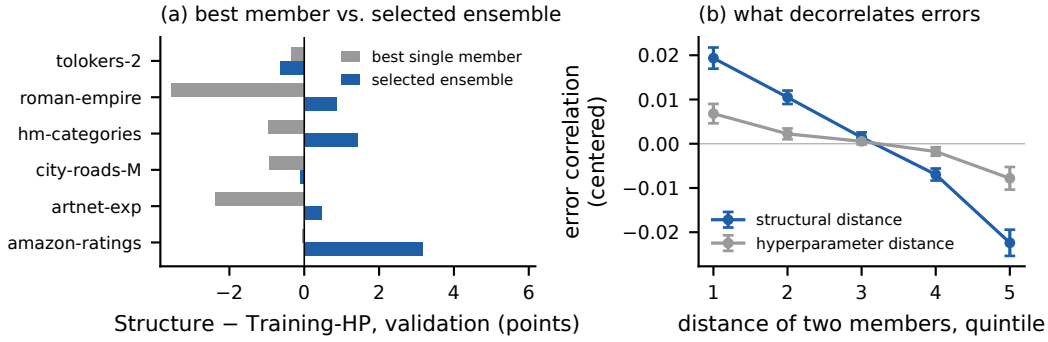


Figure 3: Why sampled views help (GBDT candidates off). (a) *Structure* minus *Training-HP* arm on validation, for the best single member and the selected ensemble. The *Training-HP* member is the best of 64 hyperparameter settings. (b) Test residual correlation of two members of a *Both* pack against the structural or hyperparameter distance of the pair, in quintile bins (centered per run, pooled over 18 runs on six datasets; mean ± s.e.).

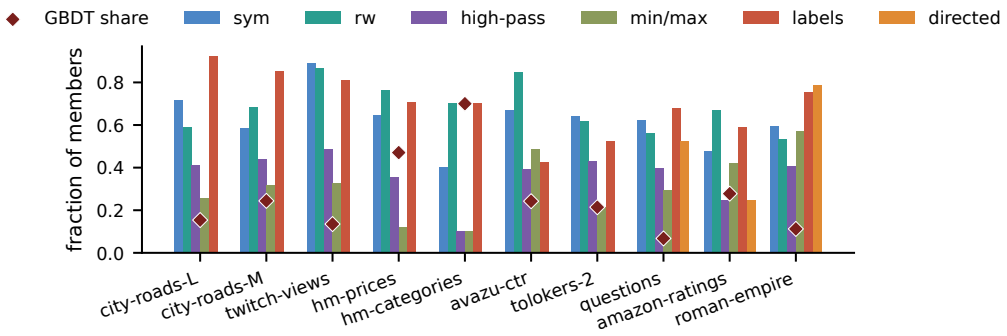


Figure 4: One sampling configuration, different selected ensembles. Bars show the fraction of selected members whose inputs contain each view family. Diamonds show the GBDT share. View families can overlap within a member, so the bars need not sum to one. Results are pooled over three seeds on GraphLand RH and ten official splits on the heterophily datasets shown. Directed views are available only on graphs with asymmetric edges. Table 15 gives all sixteen datasets.

The selected model families also vary. GBDT candidates form 70% of the ensemble on hm-categories, compared with 3% on minesweeper. On roman-empire, where separating edge directions restores homophily, directed views appear in 0.79 of selected members. The mixed pool thus supplies different combinations without a per-dataset choice of model family. Appendix E.2 compares selected view shares with their sampling frequencies, and Appendix E.3 reports the complete compositions and hop budgets.

## 6 Conclusion

GraphPack replaces per-dataset search with one packed run of sampling and selection. Given the same inputs, it is best or tied on seven of ten graphs against three strong pipelines. The search behind the tuned decoupled pipeline costs 1.6–36× the whole run. The useful diversity comes from what each member sees: splitting the views across members beats giving every member all of them. An ensemble of the best sampled configuration does not replace the sampled ensemble. Selection also adapts the model and view mix to each graph under the same sampling rules. The results across graph regimes motivate extending the candidate pool with message-passing and pretrained predictors and studying selection under temporal shift.

### AI USE STATEMENT

We used generative AI tools to help implement parts of the experiment code, and write table and figure scripts. They also helped edit the paper. We take responsibility for the final content, including all AI-assisted text, code, and analyses.

### ETHICS STATEMENT

We use publicly released benchmark datasets under their published terms and collect no new personal data or conduct human-subject studies. Applications to social and commercial graphs may raise privacy and fairness concerns; our experiments evaluate prediction performance rather than deployment in such settings.

### REPRODUCIBILITY STATEMENT

Table 1 and Appendix B specify the method and its fixed defaults. Appendix C documents the data, splits, preprocessing, metrics, seeds, hardware, and baseline protocols; Appendix F lists individual GraphPack scores. Code: `https://anonymous.4open.science/r/graphpack`.

## REFERENCES


Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A next-generation hyperparameter optimization framework. In *KDD*, 2019.

Gleb Bazhenov, Oleg Platonov, and Liudmila Prokhorenkova. Graphland: Evaluating graph machine learning models on diverse industrial data. In *NeurIPS Datasets and Benchmarks Track*, 2025.

James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 13:281–305, 2012.

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

Jiuhai Chen, Jonas Mueller, Vassilis N. Ioannidis, Soji Adeshina, Yangkun Wang, Tom Goldstein, and David Wipf. Does your graph need a confidence boost? Convergent boosted smoothing on graphs with tabular node features. In *ICLR*, 2022.

Jiuhai Chen, Jonas Mueller, Vassilis N. Ioannidis, Tom Goldstein, and David Wipf. Graph neural networks formed via layer-wise ensembles of heterogeneous base models. *Transactions on Machine Learning Research*, 2024.

Eli Chien, Jianhao Peng, Pan Li, and Olgica Milenkovic. Adaptive universal generalized PageRank graph neural network. In *ICLR*, 2021.

Chenhui Deng, Zichao Yue, and Zhiru Zhang. Polynormer: Polynomial-expressive graph transformer in linear time. In *ICLR*, 2024.

Gangda Deng, Yuxin Yang, Ömer Faruk Akgül, Hanqing Zeng, Yinglong Xia, Rajgopal Kannan, and Viktor Prasanna. Training diverse graph experts for ensembles: A systematic empirical study. *arXiv:2510.18370*, 2025a.

Gangda Deng, Hongkuan Zhou, Rajgopal Kannan, and Viktor Prasanna. Mixture of scope experts at test: Generalizing deeper graph neural networks with shallow variants. In *NeurIPS*, 2025b.

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

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

Nick Erickson, Jonas Mueller, Alexander Shirkov, Hang Zhang, Pedro Larroy, Mu Li, and Alexander Smola. AutoGluon-tabular: Robust and accurate AutoML for structured data. *arXiv:2003.06505*, 2020.

Nick Erickson, Lennart Purucker, Andrej Tschalzev, David Holzmüller, Prateek Desai, David Salinas, and Frank Hutter. TabArena: A living benchmark for machine learning on tabular data. In *NeurIPS Datasets and Benchmarks Track*, 2025.

Matthias Feurer, Aaron Klein, Katharina Eggensperger, Jost Tobias Springenberg, Manuel Blum, and Frank Hutter. Efficient and robust automated machine learning. In *NeurIPS*, 2015.

Ben Finkelshtein, Xingyue Huang, Michael Bronstein, and İsmail İlkan Ceylan. Cooperative graph neural networks. In *ICML*, 2024.

Fabrizio Frasca, Emanuele Rossi, Davide Eynard, Ben Chamberlain, Michael Bronstein, and Federico Monti. SIGN: Scalable inception graph neural networks. *arXiv:2004.11198*, 2020.

Yang Gao, Hong Yang, Peng Zhang, Chuan Zhou, and Yue Hu. Graph neural architecture search. In *IJCAI*, 2020.

Yury Gorishniy, Akim Kotelnikov, and Artem Babenko. Tabm: Advancing tabular deep learning with parameter-efficient ensembling. In *ICLR*, 2025.

Yury Gorishniy, Akim Kotelnikov, Ivan Rubachev, and Artem Babenko. Tabpack: Efficient hyperparameter ensembles for tabular deep learning. In *ICML*, 2026.

Léo Grinsztajn, Edouard Oyallon, and Gaël Varoquaux. Why do tree-based models still outperform deep learning on typical tabular data? In *NeurIPS*, 2022.

William L. Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *NeurIPS*, 2017.

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

Tin Kam Ho. The random subspace method for constructing decision forests. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(8):832–844, 1998.

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

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*, 2020.

Gao Huang, Yixuan Li, Geoff Pleiss, Zhuang Liu, John E. Hopcroft, and Kilian Q. Weinberger. Snapshot ensembles: Train 1, get M for free. In *ICLR*, 2017.

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

Sergei Ivanov and Liudmila Prokhorenkova. Boost then convolve: Gradient boosting meets graph neural networks. In *ICLR*, 2021.

Keller Jordan, Yuchen Jin, Vlado Boza, Jiacheng You, Franz Cesista, Laker Newhouse, and Jeremy Bernstein. Muon: An optimizer for hidden layers in neural networks. https://kellerjordan.github.io/posts/muon/, 2024.

Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. LightGBM: A highly efficient gradient boosting decision tree. In *NeurIPS*, 2017.

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

Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In *NeurIPS*, 2017.

Olivier Laurent, Adrien Lafage, Enzo Tartaglione, Geoffrey Daniel, Jean-Marc Martinez, Andrei Bursuc, and Gianni Franchi. Packed-ensembles for efficient uncertainty estimation. In *ICLR*, 2023.

Yuwen Li, Miao Xiong, and Bryan Hooi. GraphCleaner: Detecting mislabelled samples in popular graph learning benchmarks. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 20195–20209. PMLR, 2023. URL https://proceedings.mlr.press/v202/li23ai.html.

Lequan Lin, Dai Shi, Andi Han, Zhiyong Wang, and Junbin Gao. Diffusing to the top: Boost graph neural networks with minimal hyperparameter tuning. In *ICLR*, 2025.

Yuankai Luo, Lei Shi, and Xiao-Ming Wu. Classic GNNs are strong baselines: Reassessing GNNs for node classification. In *NeurIPS*, 2024.

Sunil Kumar Maurya, Xin Liu, and Tsuyoshi Murata. Improving graph neural networks with simple architecture design. *arXiv:2105.07634*, 2021.

Amrit Nagarajan, Jacob R. Stevens, and Anand Raghunathan. Efficient ensembles of graph neural networks. In *DAC*, 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 *ICLR*, 2023.

Oleg Platonov, Gleb Bazhenov, Dmitry Eremeev, and Liudmila Prokhorenkova. A fair evaluation of graph foundation models for node property prediction. *arXiv:2606.24509*, 2026. ICML 2026 Workshop on Graph Foundation Models.

Emanuele Rossi, Bertrand Charpentier, Francesco Di Giovanni, Fabrizio Frasca, Stephan Günnemann, and Michael Bronstein. Edge directionality improves learning on heterophilic graphs. In *LoG*, 2023.

Celia Rubio-Madrigal and Rebekka Burkholz. Fixed aggregation features can rival GNNs. In *ICML*, 2026.

Yunsheng Shi, Zhengjie Huang, Shikun Feng, Hui Zhong, Wenjing Wang, and Yu Sun. Masked label prediction: Unified message passing model for semi-supervised classification. In *IJCAI*, 2021.

Chuxiong Sun, Hongming Gu, and Jie Hu. Scalable and adaptive graph neural networks with self-label-enhanced training. *arXiv:2104.09376*, 2021.

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

Yangkun Wang, Jiarui Jin, Weinan Zhang, Yongyi Yang, Jiuhai Chen, Quan Gan, Yong Yu, Zheng Zhang, Zengfeng Huang, and David Wipf. Why propagate alone? parallel use of labels and features on graphs. In *ICLR*, 2022.

Wenqi Wei, Mu Qiao, and Divyesh Jadav. GNN-ensemble: Towards random decision graph neural networks. In *IEEE BigData*, 2023.

Yeming Wen, Dustin Tran, and Jimmy Ba. BatchEnsemble: An alternative approach to efficient ensemble and lifelong learning. In *ICLR*, 2020.

Florian Wenzel, Jasper Snoek, Dustin Tran, and Rodolphe Jenatton. Hyperparameter ensembles for robustness and uncertainty quantification. In *NeurIPS*, 2020.

Felix Wu, Amauri Souza, Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Weinberger. Simplifying graph convolutional networks. In *ICML*, 2019.

Qitian Wu, Wentao Zhao, Chenxiao Yang, Hengrui Zhang, Fan Nie, Haitian Jiang, Yatao Bian, and Junchi Yan. SGFormer: Simplifying and empowering transformers for large-graph representations. In *NeurIPS*, 2023.

Jin Xu, Mingjian Chen, Jianqiang Huang, Xingyuan Tang, Ke Hu, Jian Li, Jia Cheng, and Jun Lei. AutoHEnsGNN: Winning solution to AutoGraph challenge for KDD Cup 2020. In *ICDE*, 2022.

Jiaxuan You, Rex Ying, and Jure Leskovec. Design space for graph neural networks. In *NeurIPS*, 2020.

Lingfan Yu, Jiajun Shen, Jinyang Li, and Adam Lerer. Scalable graph neural networks for heterogeneous graphs. *arXiv:2011.09679*, 2020.

Sheheryar Zaidi, Arber Zela, Thomas Elsken, Chris C. Holmes, Frank Hutter, and Yee Whye Teh. Neural ensemble search for uncertainty estimation and dataset shift. In *NeurIPS*, 2021.

Wentao Zhang, Yu Shen, Zheyu Lin, Yang Li, Xiaosen Li, Wen Ouyang, Yangyu Tao, Zhi Yang, and Bin Cui. PaSca: A graph neural architecture search system under the scalable paradigm. In *WWW*, 2022a.

Wentao Zhang, Ziqi Yin, Zeang Sheng, Yang Li, Wen Ouyang, Xiaosen Li, Yangyu Tao, Zhi Yang, and Bin Cui. Graph attention multi-layer perceptron. In *KDD*, 2022b.

Xin Zhang, Daochen Zha, and Qiaoyu Tan. E2GNN: Efficient graph neural network ensembles for semi-supervised classification. *arXiv:2405.03401*, 2024.

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

Xiaojin Zhu, Zoubin Ghahramani, and John Lafferty. Semi-supervised learning using gaussian fields and harmonic functions. In *ICML*, 2003.

## A Extended Related Work

This appendix expands Section 2.

**Tuned classic models and the cost of search.** The ranking of message-passing models, from GCN, GraphSAGE, and GAT (Kipf & Welling, 2017; Hamilton et al., 2017; Veličković et al., 2018) to recent graph transformers (Wu et al., 2023; Deng et al., 2024; Finkelshtein et al., 2024), changes across datasets. Careful re-evaluations find that standard GNNs match or beat most specialized heterophily architectures (Platonov et al., 2023), and that tuned classic GNNs with residual connections, normalization, and dropout match recent graph transformers (Luo et al., 2024). On GraphLand, the best tuned baseline is a GNN on some graphs and a GBDT with neighborhood feature aggregation on others (Bazhenov et al., 2025). GraphPack targets the cost of this per-dataset search by training and selecting a sampled population in one run.

**Decoupled propagation and fixed aggregations.** SGC reduces a GCN to one precomputed propagation and a linear model (Wu et al., 2019). SIGN feeds several precomputed operators and hop depths to one MLP (Frasca et al., 2020). SAGN and GAMLP attend over hops (Sun et al., 2021; Zhang et al., 2022b). FSGNN learns a soft selection of hop features (Maurya et al., 2021). GPR-GNN learns Generalized PageRank weights that adapt to homophilous or heterophilous label patterns (Chien et al., 2021). NARS samples subsets of relation types in heterogeneous graphs, computes neighbor-averaged features on each induced subgraph, and combines all of them with a learned 1D convolution inside one SIGN classifier (Yu et al., 2020). Fixed aggregation features (FAF) concatenate mean, sum, max, and min aggregations over several hops and train one tuned MLP on them (Rubio-Madrigal & Burkholz, 2026). They rival or beat state-of-the-art GNNs on 12 of 14 benchmarks and fall behind on roman-empire and minesweeper, which favor deep message passing. GraphPack adds directed views, which can restore homophily on directed graphs (Rossi et al., 2023), as well as high-pass powers and one-hop min/max aggregations. The difference from all of these models is how views are chosen. Each GraphPack member draws its own view subset and training hyperparameters, and validation-based selection over trained members replaces both the per-dataset search and the learned hop weights. NARS is the closest precedent for sampling view subsets. It fuses the subsets inside one model, and GraphPack keeps one subset per member and combines members at the output.

**Graph architecture search and design spaces.** GraphNAS searches message-passing architectures with a reinforcement-learning controller (Gao et al., 2020). You et al. (2020) define a GNN design space, show that the best design differs across tasks, and transfer designs between similar tasks. PaSca searches a 150k-design space of scalable GNNs for designs that balance accuracy and efficiency (Zhang et al., 2022a). Each runs an outer search loop and returns one architecture per dataset. GraphPack fixes one sampling distribution over a design space of this kind for every dataset and runs no outer loop.

**Labels as inputs.** Label propagation predicts from graph-smoothed training labels (Zhu et al., 2003), and C&S corrects a simple base predictor with propagated residuals and labels (Huang et al., 2021). UniMP embeds labels with features and trains with masked label prediction (Shi et al., 2021), an early published form of the label trick. Wang et al. (2022) formalize the trick as a random split of the training labels into inputs and targets. SAGN with self-label-enhanced training and GAMLP reuse propagated training labels together with stage-wise training on reliable pseudo-labels (Sun et al., 2021; Zhang et al., 2022b). GraphCleaner zeroes the diagonal of multi-hop propagation operators so that no node receives its own label (Li et al., 2023). We adopt this rule. GraphPack differs in three ways. Its label views are bank columns computed once, so training needs no label masking. Each view is divided by its label coverage, which enters as a separate channel. The label budget is sampled per member, so selection can drop label views where they do not help. Our $\text{GNN}_{+\text{lab}}$ baseline uses the label trick of Wang et al. (2022).

**Efficient and diverse ensembles.** Deep ensembles (Lakshminarayanan et al., 2017) multiply training cost. Snapshot ensembles reuse one trajectory (Huang et al., 2017), and BatchEnsemble and packed ensembles share or fuse member parameters (Wen et al., 2020; Laurent et al., 2023). Hyperdeep ensembles vary hyperparameters as well as initializations (Wenzel et al., 2020). Neural ensemble search (NES) varies architectures and selects members greedily on validation data (Zaidi et al., 2021). With the same budget, it also beats the same selection procedure applied to different random initializations of one architecture (DeepEns+ES). The random subspace method trains each tree on a random feature subset (Ho, 1998). On tabular data, TabM made parameter-efficient MLP ensembles a strong default (Gorishniy et al., 2025), and TabPack samples hyperparameters per member and selects an ensemble from checkpoints during one packed run (Gorishniy et al., 2026). GraphPack adds a structural axis to this approach. Members differ in which views of one shared bank they see, a graph analogue of random subspaces. Section 5 finds this axis more effective than the hyperparameter axis on graphs.

**Graph ensembles and GBDT–GNN hybrids.** Earlier graph ensembles target robustness or efficiency (Wei et al., 2023; Nagarajan et al., 2022; Zhang et al., 2024). Deng et al. (2025a) compare 20 diversification strategies for separately trained graph experts. In increasing order of diversity, their ranking is reinitialization, directional modeling, hyperparameters, architectures, and training-data selection. Moscat trains experts of depth 0 to $L_{\max}$ and mixes them with a node-wise gate fitted on a holdout set (Deng et al., 2025b). AutoHEnsGNN won the KDD Cup 2020 AutoGraph challenge with a proxy-selected pool of GNNs and a hierarchical ensemble (Xu et al., 2022). Because GBDTs remain strong on tabular data (Grinsztajn et al., 2022), several methods pair them with GNNs. BGNN fits new trees to the gradient updates of a jointly trained GNN (Ivanov & Prokhorenkova, 2021), and EBBS alternates boosting with graph propagation (Chen et al., 2022). BestowGNN stacks bagged GBDTs, MLPs, and other learners with propagation between layers (Chen et al., 2024). GNN-Diff generates GNN parameters with a diffusion model trained on checkpoints of a light search (Lin et al., 2025). In GraphPack, one sampled population trained as one packed module supplies the diversity. GBDTs join MLP snapshots in one greedy selection pool, with no stacking or joint training. Its ablation ranks structure above hyperparameters, in line with Deng et al. (2025a).

**AutoML, post-hoc ensembling, and NAS.** Random search is more efficient than grid search for hyperparameter optimization (Bergstra & Bengio, 2012). auto-sklearn builds an ensemble of size 50 from the models evaluated during Bayesian optimization by greedy selection with replacement (Feurer et al., 2015; Caruana et al., 2004). AutoGluon bags and stacks many model families (Erickson et al., 2020). TabArena applies greedy selection to the 200 configurations evaluated during tuning and finds that the validation protocol and post-hoc ensembling matter for evaluating a model family at its full potential (Erickson et al., 2025). GraphPack draws configurations randomly from a fixed graph-view space and trains the neural population together on one GPU. Its shared bank makes different propagation choices available to both MLP and GBDT candidates, and online

selection combines their predictions throughout training. Section 4.3 compares this workflow with tuning followed by a selected deep ensemble, the analogue of DeepEns+ES.

**Graph foundation models.** Tabular in-context models (Hollmann et al., 2023) now extend to graphs. G2T-FM applies a tabular foundation model to node features augmented with neighborhood aggregations and structural embeddings (Eremeev et al., 2025). GraphPFN adds neighborhood attention layers to the LimiX tabular backbone and pretrains it on synthetic graphs (Eremeev et al., 2026). GraphAny fuses closed-form linear GNNs by attention without gradient training on the new graph (Zhao et al., 2025). TAG runs TabPFN on subsampled tables of the graph alongside linear GNNs and combines them by greedy ensemble selection (Hayler et al., 2025). Its ablation finds that the linear GNNs help through less correlated errors despite weaker standalone accuracy. We measure the same pattern for structural members. Platonov et al. (2026) compare nine such models with GNNs tuned by 100 TPE trials on ten GraphLand graphs under the RL split. Only PFN-based models beat the tuned GNNs, and fine-tuned GraphPFN is best on all ten. None of the evaluated models handles GraphLand's million-node graphs. GraphPack trains on the target graph from scratch and scales to million-node graphs. Table 12 compares accuracy and warm-inference latency. Greedy selection needs only predictions, so foundation-model outputs could join its candidate pool.

# B Background, Notation, Algorithms, and Cost

We give the notation, sampling details, LightGBM candidate settings, algorithms, and cost analysis behind the method in Section 3.

## B.1 Notation and label views

Table 5 summarizes the notation used throughout the paper.

Table 5: Notation used in the method.

| Symbol | Meaning |
|---|---|
| $G = (V, E)$, $n$ | graph and number of nodes |
| $\mathbf{A}$, $\mathbf{A}_{\text{dir}}$ | symmetrized and directed adjacency |
| $\mathbf{D}$, $\mathbf{D}_{\text{in}}$, $\mathbf{D}_{\text{out}}$ | degree matrices |
| $\hat{\mathbf{A}}_{\text{sym}}$, $\hat{\mathbf{A}}_{\text{rw}}$ | normalized propagation operators |
| $\bar{\mathbf{A}}$ | self-loop-free symmetric operator |
| $\mathbf{X}$, $d$ | node features and their dimension |
| $\mathbf{Y}$, $\mathbf{c}$, $C$ | train-masked targets, label coverage, number of classes |
| $\mathbf{B}$, $D$ | propagation bank and its width |
| $\mathbf{L}_1$, $\mathbf{L}_2$ | diffused-label views |
| $K$, $K_{\text{hp}}$, $K_{\text{dir}}$ | low-pass, high-pass, and directed hop limits |
| $m$, $m_{\text{ens}}$ | pack size and ensemble-size cap |
| $p_{\text{base}}$, $p_{\text{ens}}$ | member and ensemble patience |
| $w$, $b$ | hidden width and batch size |
| $Q$, $\mathcal{M}$, $P_q$ | candidate pool, ensemble, predictions of candidate $q$ |
| $s$ | task metric on validation nodes |
| $V_{\text{tr}}$, $V_{\text{val}}$, $V_{\text{te}}$ | train, validation, and test nodes |

The hop limits are $K = 6$, $K_{\text{hp}} = \min(K, 3)$, and $K_{\text{dir}} = \min(K, 4)$, and $C = 1$ for regression. Inverse degrees of isolated nodes are set to zero in every operator. The train-masked target matrix of Section 3.1 is

$$\mathbf{Y}_i = \begin{cases} \mathbf{e}_{y_i}, & i \in V_{\text{tr}} \text{ (classification)}, \\ (y_i - \mu_{\text{tr}})/\sigma_{\text{tr}}, & i \in V_{\text{tr}} \text{ (regression)}, \\ \mathbf{0}, & i \notin V_{\text{tr}}, \end{cases} \tag{2}$$

where $\mu_{\text{tr}}$ and $\sigma_{\text{tr}}$ are computed from training targets only. With the coverage indicator $\mathbf{c}$ ($\mathbf{c}_i = 1$ for $i \in V_{\text{tr}}$ and 0 otherwise), the diffused-label views are

$$\mathbf{L}_k = \Big[ (\mathbf{P}_k \mathbf{Y}) \oslash \max(\mathbf{P}_k \mathbf{c}, \epsilon) \,\Big\|\, \mathbf{P}_k \mathbf{c} \Big], \qquad \mathbf{P}_1 = \bar{\mathbf{A}}, \quad \mathbf{P}_2 = \bar{\mathbf{A}}^2 - \operatorname{diag}(\bar{\mathbf{A}}^2), \tag{3}$$

where the division is row-wise and $\epsilon = 10^{-6}$. The first block is the mean training label over the labeled $k$-hop mass of a node, and the second block is that mass. Because $\bar{\mathbf{A}}$ has zero diagonal, $\mathbf{P}_1$ never returns a node's own label. For $\mathbf{P}_2$, the diagonal $\mathrm{diag}(\bar{\mathbf{A}}^2)_{ii} = \sum_j \bar{\mathbf{A}}^2_{ij}$ is computed exactly from the symmetric operator and subtracted. The directed label views apply the same construction to $\mathbf{D}_{\mathrm{out}}^{-1}\mathbf{A}_{\mathrm{dir}}$ and $\mathbf{D}_{\mathrm{in}}^{-1}\mathbf{A}_{\mathrm{dir}}^{\top}$, after the diagonal of $\mathbf{A}_{\mathrm{dir}}$ is set to zero. Random splits distribute observed training labels throughout the graph. Temporal splits restrict them to the earliest period, which can change label coverage across periods (Appendix D.3).

### B.2 Sampling details

Table 1 lists the sampling distributions of the MLP members. The following details complete it. The hop budget $k$ is drawn from $\{0, \ldots, K\}$ and the operator from $\{\hat{\mathbf{A}}_{\mathrm{sym}}, \hat{\mathbf{A}}_{\mathrm{rw}}, \mathrm{both}\}$. A member with $k = 0$ uses no low-pass views, so the operator *none* occurs exactly when $k = 0$. A member drops the raw view with probability $0.15$ only if it has selected low-pass, high-pass, or min/max views. Directed views use $k_{\mathrm{dir}} \sim U\{1, \ldots, K_{\mathrm{dir}}\}$ hops in both directions. Dropout is $0$ with probability $0.5$ and is otherwise drawn from $U[0, 0.5]$. When labels are enabled and a member has drawn directed views, it adds the two directed one-hop label views with probability $0.7$, independently of its label budget. A member therefore includes at least one label view with probability $0.5 + 0.5 \cdot 0.6 \cdot 0.7 = 0.71$ on graphs with directed views, and with probability $0.5$ elsewhere.

The directed adjacency $\mathbf{A}_{\mathrm{dir}}$ is kept only when the stored edge list is asymmetric. The GraphLand graphs, including the web graph, are symmetrized when loaded, as in GraphLand's own protocol (Bazhenov et al., 2025), and so is ogbn-arxiv. Directed views and directed label views therefore occur only on the four heterophily benchmarks, whose stored edge lists are asymmetric. Member configurations are drawn from a generator seeded with the run seed.

### B.3 Greedy ensemble construction

Let $Q$ be a candidate pool and $P_q$ the validation predictions of candidate $q \in Q$. Greedy forward selection (Caruana et al., 2004) initializes $\mathcal{M} = \{\arg\max_{q \in Q} s(P_q)\}$ and then selects

$$q^\star = \arg \max_{q \in Q \setminus \mathcal{M}} s\left( \frac{\sum_{j \in \mathcal{M}} P_j + P_q}{|\mathcal{M}| + 1} \right). \tag{4}$$

We add $q^\star$ only when it improves the current ensemble score and stop when no candidate improves or $|\mathcal{M}| = m_{\mathrm{ens}}$. Caruana et al. (2004) select with replacement. We select without replacement, so each candidate enters the average at most once. The deployed prediction is the same average optimized in Eq. (4). MLP snapshots and GBDTs are therefore handled uniformly, and no learned combiner is needed.

### B.4 GBDT candidates

Each run trains four LightGBM candidates (Ke et al., 2017) one after another in a background CPU thread while the MLP pack trains on the GPU. Each candidate draws a view subset from the structural axis of Table 1 with the distributions of Appendix B.2, except that it never draws directed views or directed label views. Label views are available to it whenever the run uses them. The candidate is trained on the selected bank columns of the nodes in $V_{\mathrm{tr}}$, with these settings:

- objective: L2 regression, binary log-loss, or multiclass softmax, according to the task.
- trees: at most $1000$, with early stopping after $50$ rounds without improvement of LightGBM's default loss for the objective on the scored validation nodes (at most $10^5$, Appendix B.6).
- learning rate: LogU$[0.02, 0.2]$.
- number of leaves: $\lfloor 2^u \rfloor$ with $u \sim U[4, 8]$, a log-uniform integer from 16 to 255.
- row subsampling: fraction U$[0.6, 1.0]$, redrawn at every boosting iteration.
- column subsampling per tree: fraction U$[0.5, 1.0]$.
- minimum samples per leaf: $\lfloor 2^u \rfloor$ with $u \sim U[3, 7]$, from $8$ to $127$.
- threads: $16$.

A finished candidate predicts the scored validation nodes and all test nodes once. It outputs class probabilities (the positive-class probability for binary tasks) or standardized regression values. These formats match the MLP outputs, so greedy averaging treats both kinds of candidate in the same way. The candidate enters the pool at the next epoch. When the MLP pack stops first, the run waits at most 30 minutes for the remaining candidates and then runs one final greedy pass (Algorithm 1). Candidate configurations and LightGBM seeds derive from the run seed. Table 3 reports GraphPack with the GBDT candidates disabled on the ten GraphLand graphs with equal-information baselines.

Two rules skip all GBDT candidates of a run. The class rule applies to multiclass tasks with more than 32 classes, because LightGBM's multiclass objective grows one tree per class in every boosting round. The memory rule applies when $|V_{\mathrm{tr}}| \cdot D > 2 \cdot 10^9$ for the full bank width $D$, which corresponds to an 8 GB float32 training matrix. On the benchmark graphs, GBDT candidates are absent only on pokec-regions. The class rule applies there on every split (183 classes), and the memory rule also applies under RH and TH ($2.7 \cdot 10^9$ cells). On the other benchmark graphs, $|V_{\mathrm{tr}}| \cdot D$ stays below $3 \cdot 10^8$ on every split. The reported ogbn-arxiv runs are an exception to the class rule: they include GBDT candidates for its 40 classes. On the held-out web graph, web-fraud has $|V_{\mathrm{tr}}| \cdot D = 1{,}444{,}671 \cdot 16{,}710 \approx 2.4 \cdot 10^{10}$. The memory rule therefore removes all GBDT candidates, and the run trains the MLP pack alone. This applies to every web task (Table 14).

The separate $\mathrm{LGBM}_{+\mathrm{lab}}$ baseline uses the full bank and fixed hyperparameters, as specified in Appendix C.3.

### B.5 Algorithms

Algorithm 1 summarizes one GraphPack run. Algorithm 2 details the selection pass of one epoch. It includes the lazy rule for test predictions that bounds memory on million-node graphs.

**Algorithm 1:** GraphPack: one run

**Input:** graph $G$, features $\mathbf{X}$, targets on $V_{\mathrm{tr}}$

$\mathbf{B} \leftarrow$ propagation bank with label views (Section 3.1);
$\{c_i\}_{i=1}^{m} \sim$ configuration space (Table 1);
initialize the packed model and packed optimizer from $\{c_i\}$;
start a CPU thread that trains GBDT candidates (Appendix B.4);
$\mathcal{M} \leftarrow \emptyset$ ; `// current ensemble`
**for** epoch $= 1, 2, \ldots$ **do**
    one packed epoch over mini-batches of $V_{\mathrm{tr}}$ (per-member losses, updates masked for stopped members);
    $P \leftarrow$ validation predictions of all members;
    stop each member whose validation score has not improved for more than $p_{\mathrm{base}}$ epochs;
    $Q \leftarrow$ current snapshots of live members $\cup\, \mathcal{M} \cup$ finished GBDT candidates;
    $\mathcal{M} \leftarrow$ greedy selection on $Q$ (Algorithm 2);
    record $\mathcal{M}$ as the best ensemble if its validation score is the highest so far;
    **if** *no new best for more than* $p_{\mathrm{ens}}$ *epochs or no live member* **then break**;
**end**
wait at most 30 minutes for unfinished GBDT candidates;
rerun greedy selection on the best ensemble and all GBDT candidates, and keep the result if its validation score is higher;

**Output:** average of the best ensemble's cached test predictions

### B.6 Cost analysis

**Precomputation.** The bank costs $O\big(\sum_{\mathrm{op}} K_{\mathrm{op}}\, \mathrm{nnz}(\mathbf{A})\, d\big)$ sparse–dense multiplications on CPU, performed once. The min/max views add $O(\mathrm{nnz}(\mathbf{A})\, d)$ and the label views $O(\mathrm{nnz}(\mathbf{A})\, C)$. Measured bank construction, label views included, takes 0.3 to 0.4 seconds on minesweeper and 138 to 145 seconds on pokec-regions under RH, the extremes over the benchmark graphs. Pokec-regions has 1.63M nodes and 30.6M directed edges, 22.3M after symmetrization.

**Training.** One packed step on a batch of $b$ training nodes costs $O\big(m\, b\, (D\, w + (L_{\max} - 1)\, w^2 + w\, C)\big)$ for bank width $D$, hidden width $w = 384$, and maximum depth $L_{\max} \le 3$. The first layer multiplies the full bank batch by masked weights. The first term is the largest whenever $D > 2w + C$, which

**Algorithm 2:** Online greedy selection (one epoch's rebuild)

**Input:** pool $Q$, validation predictions $P_q$ for all $q \in Q$ on at most $10^5$ validation nodes, cached test predictions of earlier selections

$\mathcal{M} \leftarrow \{\arg\max_{q \in Q} s(P_q)\}$;
**while** $|\mathcal{M}| < m_{\text{ens}}$ **do**
  $q^\star \leftarrow \arg\max_{q \in Q \setminus \mathcal{M}} s\big(\text{mean}(P_{\mathcal{M} \cup \{q\}})\big)$;
  **if** $s\big(\text{mean}(P_{\mathcal{M} \cup \{q^\star\}})\big) \leq s\big(\text{mean}(P_{\mathcal{M}})\big)$ **then break**;
  $\mathcal{M} \leftarrow \mathcal{M} \cup \{q^\star\}$;
**end**
**if** *some members of* $\mathcal{M}$ *have no cached test predictions* **then**
  run one packed forward pass over $V_{\text{te}}$ and keep only the outputs of these members (lazy);
  cache them with the members (float16 for more than 16 classes);
**end**

**Output:** $\mathcal{M}$ with cached validation and test predictions

holds on 13 of our 16 benchmark graphs ($D$ ranges from 194 to 7,876, Appendix C). The batch size $b$ is 256 for fewer than $2 \cdot 10^4$ training nodes. It is 512 for larger training sets with fewer than $10^5$ nodes, and 1024 otherwise. Stopped members stay in the packed forward pass, and only their losses and updates are masked. The cost per step therefore does not fall as members stop. Each epoch adds one packed forward pass on the scored validation nodes. On graphs with more than $10^5$ validation nodes, we score a fixed random subsample of $10^5$ nodes, which also serves as the early-stopping set of the GBDT candidates. A greedy rebuild evaluates the metric $O(m_{\text{ens}}\,|Q|)$ times on predictions of size $|V_{\text{val}}| \times C$, with $|Q| \leq m + m_{\text{ens}} + 4 = 100$. The ranking metrics AP and AUC add one sort per evaluation.

**Measured cost.** Every search in Table 10 runs exactly 50 trials. One complete GraphPack run, including bank construction, therefore takes $50/r$ times the mean trial time, where $r$ is the time ratio in the table. Over the fourteen searches this ranges from 1.4 trials (twitch-views) to 31 trials (city-reviews), and ranges from 1.6 to 17 over the four label-free searches. A trial trains one model with the same trainer on a bank without min/max views. These ratios compare the measured workflows, including their stopping times. Complete runs on the GraphLand graphs under RH and on the heterophily suite take 30 seconds (tolokers-2) to 2.2 hours (pokec-regions, 8,089 seconds averaged over three seeds) on one 24 GB GPU. On the held-out web graph, web-fraud (two runs) has a bank of width 16,710, built in 417 seconds, and a complete run takes 4.1 hours on average on the same GPU. Table 14 gives the bank widths, bank times, and run times of all three web tasks.

**Memory and placement.** Three rules keep million-node, many-class runs within one 24 GB GPU and the host memory of one machine. First, the bank is placed by size: float32 on the GPU up to 6 GB, float16 on the GPU up to 22 GB, and host memory with per-batch transfer beyond that. Among the benchmark graphs, only pokec-regions (21.7 GB in float32) leaves the first tier, and it trains from a float16 bank on the GPU. The web-fraud bank takes 193 GB in float32, so that run keeps the bank in host memory and copies each batch to the GPU. Second, for multiclass tasks with more than 16 classes, predictions are stored in float16, and greedy selection upcasts them to float32 in chunks of at most about 2 GB. Third, test predictions are computed lazily (Algorithm 2). Only test predictions of snapshots that enter the ensemble are stored. This bounds test-prediction memory by $O(m_{\text{ens}}\,|V_{\text{te}}|\,C)$. On pokec-regions under RL (183 classes, 1.31M test nodes), float16 test predictions of all 64 members would take about 31 GB of host memory at every epoch. The selected ensembles there have 3 to 10 members over three seeds, far below $m_{\text{ens}}$.

# C Experimental Details

## C.1 Datasets, splits, and preprocessing

Table 6 lists the sixteen development graphs, their sizes, their metrics, and the width of the bank built by GraphPack on each graph. All scores are multiplied by 100, and higher is better for every metric.

**GraphLand.** We use the official RH, RL, and TH split masks and the metrics of Bazhenov et al. (2025): average precision (AP) for binary classification, accuracy for multiclass classification, and $R^2$ for regression. RH divides the nodes at random into $50\%$ training, $25\%$ validation, and $25\%$ test nodes. Nodes without a target are excluded from all three sets. GraphLand converts directed graphs to undirected ones and removes duplicate edges. We do the same, so GraphPack builds no directed views on GraphLand graphs.

**Heterophily suite and ogbn-arxiv.** Roman-empire and amazon-ratings are scored by accuracy, and minesweeper and questions by ROC AUC (Platonov et al., 2023). Run seed $s$ uses official split $s$, so ten runs cover the ten fixed splits once each. The released edge lists of these four graphs are asymmetric, and GraphPack adds its directed views there. On minesweeper, the asymmetry reflects the storage order of the grid edges. Ogbn-arxiv uses a symmetrized graph and its official split with three seeds (Hu et al., 2020).

**Held-out web graph.** GraphLand's web graph has 2.9M nodes, which are websites, and 12.4M edges after conversion to an undirected graph (Bazhenov et al., 2025). Each raw edge is directed and records that a user followed a link from one site to the other. As on the other GraphLand graphs, we symmetrize the edges. Nodes carry 263 to 267 numerical and categorical features, depending on the task. Three tasks share the graph: web-fraud (strongly imbalanced binary classification, AP), web-traffic (regression of the visitor count, $R^2$), and web-topics (28 classes, accuracy). We evaluate the RH split. GraphLand reports its baselines on these tasks over five runs. LightGBM, LightGBM-NFA, and CatBoost exceeded its 24-hour limit on all three tasks, and XGBoost did so on web-topics and web-traffic. ResMLP-NFA exceeded its 80 GB memory limit. The web graph is not part of Table 6. Table 14 lists the split sizes and bank widths of all three web tasks.

**Preprocessing.** GraphPack, $\text{LGBM}_{+\text{lab}}$, $\text{SIGN}_{+\text{lab}}$, tune-then-ensemble, and all ablation arms share the same preprocessing procedure. Numerical features are quantile-transformed to a normal distribution, with the transform fitted on training nodes. Missing values are replaced by the training median and flagged in an indicator column. Categorical features are one-hot encoded, and levels outside the 63 most frequent training levels share one bucket. Regression targets are standardized with training statistics. $\text{GNN}_{+\text{lab}}$ uses GraphLand's own loader and preprocessing.

**Size and class rules.** The fixed configuration adapts memory use to graph size and class count. Greedy selection and early stopping score candidates on a random subsample of $10^5$ validation nodes when the validation set is larger. Test scores always use the full test set. GBDT candidates are skipped under the class and memory rules of Appendix B.4. Among the development graphs, the validation subsample and memory rule affect only pokec-regions, and the class rule disables GBDT candidates only on pokec-regions. The reported ogbn-arxiv runs include GBDT candidates despite its 40 classes (Appendix B.4). On all three web tasks, the validation subsample and memory rule apply, but the class rule does not (Table 14).

### C.2 Development protocol and held-out evaluation

We designed the configuration space, the bank, and the selection procedure while running GraphPack on the eleven GraphLand graphs under all three splits, on the heterophily suite, and on ogbn-arxiv. These datasets informed the method design; all use the resulting fixed sampling configuration.

The web graph is the held-out evaluation. Its three tasks use the same default configuration as the development datasets, with two runs per task (seeds 0 and 1).

Table 14 reports accuracy and resource use for each task.

### C.3 Equal-information baselines

The three baselines of Table 2 receive the node features, the graph, and the training labels. Like GraphPack, they use validation labels only for model selection and early stopping, and no pipeline receives validation or test labels as input. This comparison covers the ten RH datasets of Table 2. On pokec-regions, LightGBM's multiclass objective grows one tree per class in every boosting round, which is impractical for its 183 classes. GraphLand itself trains its GNNs there on an 80 GB GPU with a reduced hidden size, and a 50-trial search exceeds our compute budget. Pokec-regions and the web tasks use GraphLand's published baselines as references.

Table 6: Development datasets: node count, training and test sizes under the official RH split (official split for the heterophily suite and ogbn-arxiv), bank width, and metric (AP = average precision). The held-out web graph is described in Appendix C.1.

| Dataset | #Nodes | #Train | #Test | Bank dim | Metric |
|---|---|---|---|---|---|
| tolokers-2 | 11,758 | 5,831 | 2,974 | 492 | AP |
| city-reviews | 148,801 | 69,439 | 34,720 | 3,732 | AP |
| artnet-exp | 50,405 | 24,977 | 12,791 | 2,328 | AP |
| hm-categories | 46,563 | 23,223 | 11,657 | 2,276 | ACC |
| pokec-regions | 1,632,803 | 815,874 | 408,252 | 3,320 | ACC |
| hm-prices | 46,563 | 23,223 | 11,657 | 4,954 | $R^2$ |
| avazu-ctr | 76,269 | 38,074 | 19,114 | 4,684 | $R^2$ |
| city-roads-M | 57,073 | 18,053 | 9,027 | 1,732 | $R^2$ |
| city-roads-L | 142,257 | 61,733 | 30,867 | 2,596 | $R^2$ |
| twitch-views | 168,114 | 84,000 | 42,085 | 562 | $R^2$ |
| artnet-views | 50,405 | 24,977 | 12,791 | 1,876 | $R^2$ |
| roman-empire | 22,662 | 11,331 | 5,666 | 7,876 | ACC |
| amazon-ratings | 24,492 | 12,246 | 6,123 | 7,824 | ACC |
| minesweeper | 10,000 | 5,000 | 2,500 | 194 | AUC |
| questions | 48,921 | 24,460 | 12,231 | 7,838 | AUC |
| ogbn-arxiv | 169,343 | 90,941 | 48,603 | 2,386 | ACC |

**LGBM**$_{+\text{lab}}$ is an untuned LightGBM (Ke et al., 2017) on the complete bank of Equation (1), including both diffused-label views and their coverage channels. It uses fixed settings: 1000 trees, learning rate 0.05, 64 leaves, column subsampling 0.8, and early stopping after 50 rounds without validation improvement. All other parameters keep LightGBM's defaults. We run it with seeds 0, 1, and 2.

**SIGN**$_{+\text{lab}}$ is a SIGN-style MLP (Frasca et al., 2020). Its input concatenates the raw features, the symmetric hops $\hat{\mathbf{A}}^k_{\text{sym}}\mathbf{X}$ for $k \le K$, and the first $b$ diffused-label views. Optuna's tree-structured Parzen estimator (TPE) sampler (Akiba et al., 2019) runs 50 trials over the label budget $b \in \{0, 1, 2\}$, the hop depth $K \in \{1, \ldots, 6\}$, the MLP depth $\{1, 2, 3\}$, dropout U$[0, 0.5]$, learning rate LogU$[10^{-4}, 5 \cdot 10^{-3}]$, and weight decay LogU$[10^{-3}, 1]$. Each trial trains one model with our trainer and AdamW, using the batch sizes and the validation-based early stopping of GraphPack. We keep the trial with the best validation score and rerun its configuration with seeds 1 and 2. With the seed-0 trial, this gives three runs. Its views and hyperparameter ranges form a subset of those GraphPack samples. It has no random-walk, high-pass, min/max, or directed views, and its hops are contiguous. It trains all weights with AdamW, and GraphPack uses Muon for its hidden matrices.

The label-free searches of Table 10 follow the same procedure with $b = 0$. On roman-empire, this search and its reruns use official split 0, on which GraphPack scores 85.47 with seed 0. In Table 10, the best-test column is the highest test score among the 50 trials. The search time sums the training time of the 50 trials and excludes bank construction, the reruns, and the TPE overhead. The GraphPack time is the full wall-clock of one run, including data loading, preprocessing, bank construction, training, selection, and test prediction.

**GNN**$_{+\text{lab}}$ runs GraphLand's official code[1] with the label trick (Wang et al., 2022; Shi et al., 2021). A wrapper implements label masking in the training and evaluation functions. In every training step, a random half of the training nodes reveal their target as extra input channels, together with a flag that marks revealed nodes. The revealed target is a one-hot class, the binary label, or the transformed regression target. The loss is computed on the other half only, so a node's own target is never part of its input when its loss is computed. At evaluation, all training nodes reveal their targets. Validation and test targets are never revealed.

For each dataset we first run the GNN family that is best under RH in GraphLand's results. This is GAT on tolokers-2, city-reviews, artnet-exp, hm-prices, avazu-ctr, city-roads-M, and city-roads-L, GT on hm-categories, and GCN on twitch-views and artnet-views. Training follows GraphLand's protocol and defaults. The model has three residual message-passing blocks with hidden size 512

[1] https://github.com/yandex-research/graphland

and 4 attention heads. It trains for 1000 full-batch steps, with the best step chosen on validation. The 15-point grid crosses the learning rates $\{3 \cdot 10^{-5}, 10^{-4}, 3 \cdot 10^{-4}, 10^{-3}, 3 \cdot 10^{-3}\}$ with dropout $\{0, 0.1, 0.2\}$. Each grid point runs once, and the validation-best configuration runs three times.

On hm-prices and avazu-ctr, this family runs out of memory in 24 GB with hidden size 512, while on hm-categories it fits at this size. On these three graphs we also run GraphSAGE with hidden size 512 and the same family with hidden size 256. On avazu-ctr, GraphSAGE with fp16 mixed precision produced NaN predictions with either fraction-feature transform. The runs on this graph therefore use full precision, as do the GraphSAGE runs on hm-prices and hm-categories. A variant counts as completed when its three final runs finish with a nonzero score. Table 2 takes, for each graph, the completed variant with the best validation score. Validation selects GAT with hidden size 256 on hm-prices and GT with hidden size 512 on hm-categories. On avazu-ctr, GraphSAGE with hidden size 512 is the only completed variant. If GraphLand's best published label-free GNN of any family has a higher test mean than the selected variant, the table reports that published score with a dagger (avazu-ctr, twitch-views, and artnet-views).

GraphLand's code offers a per-dataset default for the feature and target transforms. We fix them to one of its options: quantile-normal numerical features, untransformed fraction features, and standard-scaled regression targets. The exception is the completed avazu-ctr run (GraphSAGE with hidden size 512). It also applies the quantile-normal transform to the fraction features, which is GraphLand's default on this dataset. To check the pipeline, we also run it without labels on tolokers-2, artnet-exp, and city-roads-M. Table 7 compares these runs with GraphLand's published score for the same model. Our label-free runs differ from the published scores by $+0.40$, $-0.12$, and $-0.53$ points on tolokers-2, artnet-exp, and city-roads-M (ours minus published). In this comparison, all three differences lie within the sum of the two standard deviations.

Table 7: Fidelity of the GNN$_{+\mathrm{lab}}$ pipeline on the RH split. We run GraphLand's official code with the transforms given in the text and its 15-point grid, without labels and with the label trick (mean$\pm$std over three runs of the validation-best configuration). The table also lists GraphLand's published score for the same family at GraphLand's hidden size 512 (ten runs) and GraphPack. GAT-256 marks our hidden size 256 on hm-prices.

| Dataset | model | published | ours, no labels | ours, label trick | GraphPack |
|---|---|---|---|---|---|
| tolokers-2 | GAT | 63.76 | 64.16±0.64 | 64.52±0.79 | 64.42 |
| city-reviews | GAT | 81.10 | – | 81.34±0.17 | 82.55 |
| artnet-exp | GAT | 50.62 | 50.50±0.33 | 50.67±0.59 | 53.62 |
| hm-categories | GT | 79.28 | – | 83.99±0.39 | 82.67 |
| hm-prices | GAT-256 | 81.68 | – | 82.99±0.41 | 83.10 |
| avazu-ctr | GraphSAGE | 35.20 | – | 34.63±0.57 | 36.48 |
| city-roads-M | GAT | 70.53 | 70.00±0.26 | 78.55±0.22 | 78.00 |
| city-roads-L | GAT | 66.03 | – | 79.52±0.10 | 76.91 |
| twitch-views | GCN | 77.12 | – | 74.70±1.19 | 78.14 |
| artnet-views | GCN | 61.02 | – | 54.49±0.46 | 63.74 |

### C.4 TUNE-THEN-ENSEMBLE PROTOCOL

Tune-then-ensemble (Table 3) trains 64 packed copies of the validation-best SIGN$_{+\mathrm{lab}}$ configuration of each dataset. Every member receives the searched views, depth, dropout, learning rate, and weight decay. All members train with AdamW, as in the search. Pack members share their mini-batches, so the copies differ only in initialization and dropout masks. Selection can still combine snapshots from different epochs. Greedy selection, member and ensemble early stopping (16 and 32 epochs), and the ensemble size cap of 32 are those of GraphPack. GBDT candidates are off, and we run seeds 0, 1, and 2. The column "GraphPack (no GBDT)" is the default configuration with GBDT candidates disabled, which isolates the neural pack. Recorded pack training times exclude the preceding search; the workflow cost in Table 8 includes both the search and packed ensemble training.

### C.5 DIVERSITY-AXIS ABLATION ARMS

The seven arms of Table 4 start from the same 64 configurations, drawn by the default sampler with the run seed. Each arm overwrites a fixed set of fields, and all other fields keep their sampled values. The frozen training hyperparameters are depth 2, dropout 0.15, Muon learning rate $10^{-2}$, AdamW

learning rate $7 \cdot 10^{-4}$ for biases, and weight decay $3 \cdot 10^{-2}$, near the middle of the ranges in Table 1. The canonical SIGN-style view set holds the raw features, $\hat{\mathbf{A}}^k_{\text{sym}}\mathbf{X}$ for $k \leq 4$, $\hat{\mathbf{A}}^k_{\text{rw}}\mathbf{X}$ for $k \leq 2$, and the one-hop label view. On directed graphs it adds the in- and out-neighbor views of one and two hops and the two directed one-hop label views.

- *Both* uses the default sampler unchanged.
- *Structure* keeps the sampled views and freezes the training hyperparameters.
- *Training-HP* freezes the canonical view set and keeps the sampled training hyperparameters.
- *All views* gives every member every view of the bank and keeps the sampled training hyperparameters. The bank holds the raw features, six symmetric and six random-walk hops, three high-pass powers, the one-hop min and max, and both label views. On directed graphs it also holds four in- and out-neighbor hops and the directed label views.
- *Selected structure* gives every member the view set of the best single member of the *Both* run with the same dataset and seed. This is the MLP member with the highest validation score at its best epoch. Training hyperparameters stay sampled.
- *Tuned ensemble* copies both the views and the training hyperparameters of that member into every member, which gives a deep ensemble of a validation-selected configuration.
- *Init* freezes the canonical view set and the training hyperparameters, which gives a plain deep ensemble.

The arms run on amazon-ratings and roman-empire (official split $s$ with seed $s$) and on artnet-exp, city-roads-M, hm-categories, and tolokers-2 (RH). Every arm runs seeds 0 to 4 on every graph. Table 4 and the paired counts of Section 5 use the dataset–seed pairs on which every arm has finished. GBDT candidates are off in every arm, and all other settings are the defaults. *Selected structure* and *Tuned ensemble* need a complete *Both* run first, so each of them costs two runs.

### C.6 Statistics

GraphPack results are means over seeds 0, 1, and 2 on the GraphLand graphs (all splits) and on ogbn-arxiv. On the heterophily suite, results are means over the ten official splits. The web results use two runs per task, with seeds 0 and 1. On GraphLand and ogbn-arxiv the split stays fixed across seeds, and on the heterophily suite seed $s$ uses official split $s$. The seed also changes member sampling, initialization, batch order, the GBDT candidates, and the validation subsample. Tune-then-ensemble uses the same seeds, and the ablation arms use the seeds of Appendix C.5. The symbol $\pm$ denotes the sample standard deviation (denominator $n - 1$) of our runs. Published baselines keep the standard deviation reported by their source. For GraphLand this is over ten runs, or five on pokec-regions and the web tasks.

Win, tie, and loss counts follow the rule of Bazhenov et al. (2025). Two methods tie when their means differ by no more than the sum of their standard deviations. A pipeline's gap on a dataset is the best mean among the equal-information pipelines minus its own mean. For the ablations, Table 4 summarizes the means and win/tie/loss counts across the six datasets. Paired-run counts match arms by dataset and seed. They describe how consistently a difference has the same direction across runs. Dataset-level summaries aggregate over seeds, which measure run variability within each dataset. The rank correlation in Section 5 is Spearman's $\rho$ across datasets, with a one-sided permutation test over $10^5$ random permutations.

### C.7 Hardware and software

All training runs used one server with four RTX 4090 GPUs (24 GB each, one GPU per run), 144 CPU cores, and 503 GB of RAM. Runs shared the server, and some batches placed up to three runs on one GPU. Reported wall-clock times include this contention. GraphPack and our baselines run under PyTorch 2.x with CUDA 13 in float32. Packed predictions of wide multiclass outputs are cached in fp16. The bank is built once on CPU with SciPy sparse kernels. It is placed on the GPU in fp32, on the GPU in fp16, or in CPU memory with per-batch transfer, whichever fits. LightGBM candidates train on 16 CPU threads concurrently with the GPU pack. $\text{GNN}_{+\text{lab}}$ runs in a separate

environment with GraphLand's dependencies and its default mixed precision, except for the full-precision runs listed in Appendix C.3.

**Implementation check.** A single SIGN-style configuration trained with our bank and trainer on ogbn-arxiv (seed 0) reaches 70.76. This is 1.2 points below the published tuned SIGN (71.95) and far above the feature-only MLP (55.50) (Frasca et al., 2020; Hu et al., 2020).

### C.8 Warm-inference timing protocol

Deployment latency is measured separately on one NVIDIA A800 80 GB with seed 0. Models and data are loaded before timing, and each reported value is the median of five synchronized forward passes over the complete test split. Training, fine-tuning, preprocessing, and model selection are excluded. For the dense architectures compared here, forward latency is set by the deployed architecture and the input size. Learned weight values do not change it. On graphs without a locally fine-tuned checkpoint, we therefore instantiate each method's published fine-tuning architecture without repeating the adaptation. For GraphPack on pokec-regions, we likewise time the selected member architectures without fitted weights.

### C.9 Table conventions and omitted baselines

In all tables, "–" marks a score that is not reported or not run, and "N/S" a deployment that is unsupported in the compared setting. ResMLP and LGBM denote GraphLand's tabular baselines, LGBM-NFA is LightGBM with neighborhood feature aggregation, and GT is GraphLand's local-attention graph transformer. GraphLand tuned each baseline per dataset, with its 15-point grid for GNNs and 100 Optuna trials for GBDTs (Bazhenov et al., 2025).

The "best" column of Table 2 is the best of ResMLP, LGBM, LGBM-NFA, GCN, GraphSAGE, GAT, and GT. GraphLand also reports XGBoost, CatBoost, and ResMLP-NFA, which never exceed that value on any RH dataset. Table 11 omits ResMLP, XGBoost, CatBoost, and ResMLP-NFA, which never exceed the best shown value on any TH dataset. In Table 12, the fine-tuned GraphPFN score on city-roads-L ($^{\ddagger}$) comes from Platonov et al. (2026), since the original paper has no entry.

Table 13 shows eight of the seventeen baselines of Platonov et al. (2023): the tabular, classic GNN, and ego/neighbor-separated families and FSGNN. The nine omitted models (GT, ResNet+adj, and seven heterophily-specific architectures) never exceed the best value shown for their dataset. The Polynormer numbers quoted in the text use the same ten official splits (Deng et al., 2024). FAF ($^{\dagger}$) uses only official splits 0 to 2 (Rubio-Madrigal & Burkholz, 2026), compared with ten splits for GraphPack. Moscat's ten-run result is discussed separately in Appendix D.5. Numbers from Luo et al. (2024) are five-seed means. These source protocols are retained in the reported comparisons.

### C.10 Provenance

We generated the result tables, figures, and numerical text from run-level measurements. The residual-correlation analysis also uses member predictions. We checked published baseline values against their cited sources and split protocols.

## D Additional Results

This appendix gives the complete tables behind Section 4 and the details of each comparison. Published baselines are shown with their reported mean±std. GraphPack numbers are means over three seeds (ten official splits for the heterophily suite, two runs for the held-out web tasks) with sample standard deviations. Following GraphLand, a difference within the summed standard deviations counts as a tie.

### D.1 Complete RH baselines

Table 9 lists GraphLand's published RH baselines next to both GraphPack variants. No family has the best published score on every graph. GAT leads on four graphs, LGBM-NFA on three, GCN on two, and GT on two. Full GraphPack has a higher mean than every published baseline on all eleven

Table 8: Score gaps under equal information on the 10 GraphLand RH graphs of Table 2. For each graph, the gap is the best mean score among $\text{LGBM}_{+\text{lab}}$, $\text{SIGN}_{+\text{lab}}$, $\text{GNN}_{+\text{lab}}$, and GraphPack, minus the pipeline's mean score, in points. The table also reports GraphPack's wins, ties, and losses against each pipeline. Compute is wall-clock time relative to one GraphPack run, using the geometric mean over graphs. "–" means the pipeline was not timed on dedicated GPUs. "Best of the three on validation" picks, per graph, the pipeline with the highest validation score.

| Pipeline (same inputs) | compute | mean gap | max gap | GraphPack W/T/L |
|---|---|---|---|---|
| $\text{LGBM}_{+\text{lab}}$ (untuned) | 0.12× | 1.72 | 6.18 | 8/2/0 |
| $\text{SIGN}_{+\text{lab}}$ (50-trial search) | 6.4× | 2.11 | 3.82 | 10/0/0 |
| $\text{GNN}_{+\text{lab}}$ (GraphLand grid) | – | 0.88 | 2.96 | 5/2/3 |
| Best of the three on validation | – | 0.64 | 2.53 | 6/1/3 |
| Tune-then-ensemble (64 tuned SIGN) | 9.1× | 1.75 | 3.46 | 9/1/0 |
| GraphPack without GBDT candidates | – | 1.53 | 3.93 | 6/4/0 |
| GraphPack | 1× | **0.46** | 2.60 | – |

Table 9: GraphLand RH split with seven of GraphLand's ten published baselines. Each was tuned per dataset by Bazhenov et al. (2025) and uses no label inputs (mean±std over ten runs, five for pokec-regions). XGBoost, CatBoost, and ResMLP-NFA are omitted (Appendix C). GraphPack no-lab disables label views, and full is the default configuration (three seeds). Underlined: best published mean. Bold: GraphPack mean above it. Last row: average rank over the nine columns.

| | Tabular models | | | Tuned GNNs | | | | GraphPack (one run) | |
|---|---|---|---|---|---|---|---|---|---|
| Dataset | ResMLP | LGBM | LGBM-NFA | GCN | SAGE | GAT | GT | no-lab | full |
| *Classification (AP / accuracy)* | | | | | | | | | |
| tolokers-2 | 45.96±0.46 | 49.92±0.19 | 62.19±0.35 | 60.49±0.86 | 58.42±0.92 | 63.76±1.30 | 60.32±1.21 | **64.32±1.45** | **64.42±0.57** |
| city-reviews | 75.21±0.08 | 77.88±0.13 | 81.63±0.06 | 81.05±0.10 | 80.75±0.06 | 81.10±0.11 | 80.50±0.14 | **82.28±0.19** | **82.55±0.06** |
| artnet-exp | 43.55±0.23 | 44.98±0.80 | 49.26±0.18 | 49.80±0.35 | 48.49±0.37 | 50.62±0.35 | 49.32±1.00 | **52.07±0.30** | **53.62±0.28** |
| hm-categories | 43.12±0.25 | 47.28±0.14 | 67.09±0.15 | 73.38±0.42 | 73.34±0.68 | 79.19±0.21 | 79.28±0.31 | 73.01±0.13 | **82.67±0.59** |
| pokec-regions | 5.09±0.01 | 5.10±0.01 | 12.15±0.02 | 35.08±0.62 | 40.76±0.21 | 46.72±0.69 | 50.06±0.53 | **51.83±1.74** | **59.12±0.01** |
| *Regression ($R^2$)* | | | | | | | | | |
| hm-prices | 70.11±0.48 | 73.99±0.11 | 79.78±0.09 | 79.76±0.76 | 79.89±0.46 | 81.68±0.41 | 80.90±0.42 | 80.87±0.23 | **83.10±0.33** |
| avazu-ctr | 28.03±0.22 | 29.60±0.02 | 35.35±0.04 | 34.96±0.11 | 35.20±0.20 | 35.74±0.19 | 34.38±0.31 | **36.49±0.19** | **36.48±0.20** |
| city-roads-M | 62.43±0.32 | 70.01±0.19 | 72.09±0.07 | 69.95±0.11 | 70.20±0.59 | 70.53±0.40 | 67.45±0.82 | **72.54±0.02** | **78.00±0.09** |
| city-roads-L | 53.09±0.17 | 63.66±0.09 | 66.24±0.08 | 64.65±0.27 | 65.77±0.43 | 66.03±0.24 | 64.02±0.59 | 66.19±1.18 | **76.91±0.10** |
| twitch-views | 13.36±0.01 | 13.37±0.00 | 62.14±0.01 | 77.12±0.11 | 72.02±0.16 | 76.06±0.30 | 75.57±0.15 | 76.91±0.10 | **78.14±0.34** |
| artnet-views | 36.10±0.17 | 36.38±0.08 | 60.30±0.07 | 61.02±0.13 | 56.65±0.50 | 59.01±0.52 | 58.97±0.25 | **63.15±0.22** | **63.74±0.19** |
| Avg. rank | 9.0 | 7.8 | 4.8 | 5.0 | 5.7 | 3.5 | 5.4 | 2.6 | 1.1 |

graphs and an average rank of 1.1. The no-label variant and the neural-only runs of Table 3 quantify the contributions of label views and tree candidates.

**Label views.** On city-roads-L the margin is 10.7 points, and GraphPack without label views ties LGBM-NFA (66.19 vs. 66.24). Label views supply 5.5 of the 5.9 points on city-roads-M and 7.3 of the 9.1 points on pokec-regions (Appendix D.6). On hm-categories they lift GraphPack from 6.3 points below GT to 3.4 points above it. The city-roads targets are strongly assortative, with target assortativity 0.74 on city-roads-M and 0.72 on city-roads-L. Strong homophily is not required, as hm-categories shows. Its edge homophily is only 0.16, but each node has 461 neighbors on average. A vote over the training labels of one-hop neighbors already reaches 51.0 test accuracy, against 19.5 for the majority class. Diffused label views pass such neighbor-label distributions to the members, which learn how they map to targets. Appendix E.2 relates these gains to the share of selected members using label views.

**GBDT candidates.** On artnet-exp the margin is 3.0 points. Without GBDT candidates, GraphPack reaches 49.70, below GAT's 50.62. With them, it reaches 53.62 (Table 3). The candidates form only 19% of the selected members there. On hm-categories they add 2.0 points and form 70% of the selected ensemble. LightGBM on the full bank alone reaches 82.95, second only to $\text{GNN}_{+\text{lab}}$ among the equal-information pipelines (Table 2).

On tolokers-2 and avazu-ctr, the gains from label views are smaller: they add 0.1 points and nothing. The margin over GAT is 0.7 points on tolokers-2, a tie, and 0.7 points on avazu-ctr. These comparisons show how label reuse and the mixed model pool contribute to the gains over published baselines.

### D.2 SEARCH COST

Table 10: Search cost on the same bank, trainer, and GPU. Each search runs 50 TPE trials over a SIGN sub-space of GraphPack's configuration space (see text). Search time sums the training time of the 50 trials and excludes bank construction and reruns. GraphPack time is the mean wall-clock time of one complete run, including bank construction. Test (val.-best): three-seed mean of the configuration with the best validation score. Best test: highest observed test score among the 50 trials. Time ratio: search time over GraphPack time. Upper block: the search includes a label budget ($\text{SIGN}_{+\text{lab}}$ of Table 2). Lower block: label-free searches, compared with full GraphPack. roman-empire is searched on official split 0, and its GraphPack entry is the ten-split mean (85.47 on split 0).

| | 50-trial search | | | GraphPack (one run) | | |
|---|---|---|---|---|---|---|
| Dataset | time | test (val.-best) | best test | time | test | time ratio |
| *SIGN with label views (equal information)* | | | | | | |
| artnet-exp | 5 m | 50.65 | 51.08 | 2 m | **53.62** | 2.8× |
| artnet-views | 7 m | 61.73 | 61.73 | 1 m | **63.74** | 5.0× |
| avazu-ctr | 8 m | 35.37 | 35.53 | 5 m | **36.48** | 1.7× |
| city-reviews | 15 m | 81.99 | 82.09 | 9 m | **82.55** | 1.6× |
| city-roads-L | 77 m | 75.70 | 75.71 | 5 m | **76.91** | 14.6× |
| city-roads-M | 52 m | 77.30 | 77.33 | 2 m | **78.00** | 25.5× |
| hm-categories | 39 m | 80.98 | 81.16 | 5 m | **82.67** | 8.3× |
| hm-prices | 24 m | 79.88 | 80.76 | 6 m | **83.10** | 4.0× |
| tolokers-2 | 4 m | 61.99 | 63.47 | 30 s | **64.42** | 7.2× |
| twitch-views | 147 m | 77.56 | 77.52 | 4 m | **78.14** | 35.6× |
| *SIGN without label views (classic workflow)* | | | | | | |
| city-roads-M | 66 m | 69.60 | 69.80 | 2 m | **78.00** | 32.1× |
| hm-categories | 57 m | 61.09 | 61.59 | 5 m | **82.67** | 12.2× |
| roman-empire | 21 m | 78.49 | 78.68 | 7 m | **85.69** | 3.0× |
| tolokers-2 | 8 m | 60.69 | 62.79 | 30 s | **64.42** | 15.2× |

Table 10 reports one search per graph and setting, using the SIGN search space, reruns, and cost accounting of Appendix C.3.

Across the ten equal-information searches, the search alone takes 1.6 to 35.6 times the wall-clock of a complete GraphPack run (Figure 1(b) shows the anytime curves). The selected configuration still scores 0.6–3.2 points below GraphPack. The best-test column gives the highest observed test score in each search. This reference stays below GraphPack on all ten graphs. The smallest gap is 0.46 points (city-reviews). On artnet-views and twitch-views the three-seed mean of the selected configuration is at or above the best single trial, because the reruns use other seeds. The label-free searches, the classic workflow, fall further behind. On city-roads-M, hm-categories, and tolokers-2, even GraphPack without label views (72.54, 73.01, and 64.32, Table 9) exceeds their best test trial.

For context, a separate study reports message-passing GNN search costs. Platonov et al. (2026) tune each GNN with 100 TPE trials on one A100 80 GB GPU. On the six GraphLand graphs they time, all on the RL split, this takes 1.2 hours to 7.6 days per GNN and dataset. On the same six graphs and split, one GraphPack run takes 8 to 165 seconds on one RTX 4090.

### D.3 TEMPORAL SPLIT

GraphLand defines the TH split on ten of its fourteen datasets. Of our eleven GraphLand graphs, eight have one, and city-reviews, city-roads-M, and city-roads-L do not. In a TH split, training, validation, and test nodes come from successive periods. GraphPack has the best average rank (1.5, next GAT 2.8, Table 11). Under the tie rule it wins on five graphs, ties on tolokers-2 and hm-categories, and loses on hm-prices. The largest margin is on pokec-regions, 37.62 against GT's 25.09 (+12.5).

Two diagnostics characterize the temporal shift. First, validation scores exceed test scores. On hm-categories the selected ensemble scores 73.7 on validation and 61.3 on test. On hm-prices, the corresponding scores are 77.9 and 68.5. Both early stopping and greedy selection use the validation period. Pokec-regions has a gap of 20.2 points (57.8 vs. 37.6) and the largest margin. Second, label coverage shifts. Under RH, training nodes are drawn at random, so training, validation, and test nodes have the same expected fraction of neighbors that are labeled training nodes. Under TH this

Table 11: GraphLand TH (temporal) split on the eight evaluated graphs that have one. Six of GraphLand's published baselines, each tuned per dataset without label inputs (mean±std over ten runs, five for pokec-regions) (Bazhenov et al., 2025). ResMLP, XGBoost, CatBoost, and ResMLP-NFA are omitted (Appendix C). GraphPack: default configuration, three seeds. Underlined: best published mean. Bold: GraphPack mean above it. Last row: average rank.

| | Tabular models | | Tuned GNNs | | | | GraphPack |
|---|---|---|---|---|---|---|---|
| Dataset | LGBM | LGBM-NFA | GCN | SAGE | GAT | GT | (one run) |
| *Classification (AP / accuracy)* | | | | | | | |
| tolokers-2 | 15.85±3.89 | 31.81±7.51 | 46.72±1.19 | 17.05±7.65 | 38.59±6.19 | 34.15±4.81 | **47.82±2.54** |
| artnet-exp | 37.34±0.18 | 39.89±0.37 | 40.64±0.40 | 40.50±0.84 | 41.85±0.63 | 40.10±0.60 | **43.46±0.55** |
| hm-categories | 32.08±0.25 | 52.45±0.18 | 56.91±0.55 | 59.62±0.51 | 61.28±0.97 | 63.31±0.45 | 61.29±2.34 |
| pokec-regions | 4.13±0.01 | 5.54±0.01 | 11.88±0.31 | 16.60±0.28 | 20.43±0.55 | 25.09±0.58 | **37.62±0.09** |
| *Regression ($R^2$)* | | | | | | | |
| hm-prices | 63.56±0.39 | 71.36±0.41 | 65.20±0.84 | 67.93±1.24 | 70.83±0.99 | 69.70±0.84 | 68.49±0.84 |
| avazu-ctr | 26.23±0.04 | 38.69±0.03 | 37.49±0.26 | 38.38±0.39 | 39.21±0.17 | 38.27±0.27 | **40.07±0.08** |
| twitch-views | -9.60±0.03 | 43.60±0.03 | 68.17±0.24 | 61.46±0.68 | 66.32±0.59 | 65.83±0.24 | **69.73±0.39** |
| artnet-views | 42.87±0.08 | 52.42±0.06 | 54.44±0.43 | 51.87±0.48 | 52.30±0.34 | 51.67±0.54 | **56.47±0.24** |
| Avg. rank | 7.0 | 4.5 | 3.9 | 4.6 | 2.8 | 3.8 | 1.5 |

fraction falls from 0.85 for training nodes to 0.46 for test nodes on hm-categories and hm-prices, which share one graph. The coverage channel (Section 3.1) exposes this quantity to every member, but members are fitted on well-covered training nodes. On pokec-regions, where GraphPack wins, coverage falls further, from 0.79 to 0.27.

### D.4 Low-label split

Table 12: GraphLand RL split (10% of nodes labeled): pretrained graph foundation models and GraphPack. G2T-FM and GraphPFN: pretrained and fine-tuned (PT+FT) scores reported by Eremeev et al. (2025) and Eremeev et al. (2026). ‡GraphPFN FT from Platonov et al. (2026), where the original paper has no entry. Bold score: best in row where two or more methods report one. Infer. (s): warm-inference latency, the median of five forward passes over the full test split on one A800 80 GB GPU (RL split, seed 0, Appendix C). It excludes training, fine-tuning, preprocessing, bank construction, and selection. For GraphPack, we time the complete selected ensemble. For each foundation model, we time one predictor built from its published fine-tuning architecture (three for GraphPFN on the 21-class hm-categories). Bold latency: fastest where two or more latencies are measured. N/S: not supported or not reported. GraphPack: mean±std over three seeds.

| Dataset | G2T-FM (PT+FT) | | GraphPFN (PT+FT) | | GraphPack (one run) | |
|---|---|---|---|---|---|---|
| | Score | Infer. (s) | Score | Infer. (s) | Score | Infer. (s) |
| tolokers-2 | 59.75 | 0.547 | **62.80** | 0.532 | 58.79±0.48 | **0.063** |
| city-reviews | 80.65 | 44.638 | **80.90** | 12.437 | 79.25±0.16 | **3.162** |
| artnet-exp | 50.39 | 12.729 | **54.65** | 4.773 | 48.45±0.53 | **0.431** |
| hm-categories | – | N/S | **78.67** | 36.977 | 68.17±1.80 | **2.938** |
| hm-prices | 77.37 | 31.329 | **81.06** | 13.784 | 72.29±0.70 | **2.116** |
| avazu-ctr | 34.09 | 23.778 | **35.08** | 24.018 | 33.99±0.26 | **1.598** |
| city-roads-M | 66.29 | 3.057 | **67.53** | 1.567 | 64.14±0.21 | **0.806** |
| city-roads-L | – | N/S | **64.12**‡ | N/S | 59.48±0.23 | 4.228 |
| twitch-views | 74.91 | 26.859 | **79.19** | 7.823 | 75.45±0.30 | **0.913** |
| artnet-views | 63.24 | 5.788 | **65.35** | 3.272 | 61.03±0.40 | **0.686** |
| *Beyond the foundation models' reported range* | | | | | | |
| pokec-regions | – | N/S | – | N/S | 53.37±0.25 | 2.697 |

With 10% of nodes labeled, fine-tuned GraphPFN is more accurate than GraphPack on all ten graphs it covers (Table 12), including city-roads-L (64.12 vs. 59.48). Fine-tuned G2T-FM is above GraphPack on seven of the eight graphs it reports, all except twitch-views.

Under the warm-inference protocol, GraphPack has lower forward-pass latency than both foundation-model architectures. Its selected ensemble takes 0.06 to 4.2 seconds, making it 1.9 to 15.0 times faster than GraphPFN and 3.8 to 29.6 times faster than G2T-FM. Neither foundation-model paper reports pokec-regions, and Platonov et al. (2026) exclude GraphLand's million-node

graphs because none of the foundation models they consider handles them. On pokec-regions, GraphPack scores 53.37.

### D.5 HETEROPHILY BENCHMARK AND OGBN-ARXIV

Table 13: Heterophily benchmark under the ten official splits (accuracy for roman-empire and amazon-ratings, ROC AUC for minesweeper and questions, mean±std). The first eight columns are baselines of Platonov et al. (2023) (Appendix C). FAF† is the validation-selected fixed-aggregation-feature model of Rubio-Madrigal & Burkholz (2026), run on official splits 0–2 only. GraphPack: one run per official split. Bold: best mean in each row of this table.

| | Platonov et al. (2023) | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| Dataset | ResNet | ResNet+SGC | GCN | SAGE | GAT | GAT-sep | GT-sep | FSGNN | FAF† | GraphPack |
| roman-empire | 65.88±0.38 | 73.90±0.51 | 73.69±0.74 | 85.74±0.67 | 80.87±0.30 | **88.75±0.41** | 87.32±0.39 | 79.92±0.56 | 78.11±0.38 | 85.69±0.51 |
| amazon-ratings | 45.90±0.52 | 50.66±0.48 | 48.70±0.63 | 53.63±0.39 | 49.09±0.63 | 52.70±0.62 | 52.18±0.80 | 52.74±0.83 | 55.09±0.24 | **56.16±0.58** |
| minesweeper | 50.89±1.39 | 70.88±0.90 | 89.75±0.52 | 93.51±0.57 | 92.01±0.68 | **93.91±0.35** | 92.29±0.47 | 90.08±0.70 | 90.00±0.39 | 92.30±0.59 |
| questions | 70.34±0.76 | 75.81±0.96 | 76.09±1.27 | 76.44±0.62 | 77.43±1.20 | 76.79±0.71 | 78.05±0.93 | **78.86±0.92** | 78.69±0.50 | 78.30±0.81 |

Table 13 places GraphPack among the baselines of Platonov et al. (2023) and fixed aggregation features (FAF). Against Polynormer, which reports Polynormer-r on the same ten splits with ten runs (Deng et al., 2024), GraphPack is 1.4 points higher on amazon-ratings. It is 0.6 lower on questions, a tie under the tie rule, 6.9 lower on roman-empire, and 5.2 lower on minesweeper (Deng et al., 2024). The retuned baselines of Luo et al. (2024) are stronger still on these two graphs. Over five runs, their Polynormer rerun reaches 92.66 on roman-empire and their tuned GCN 97.86 on minesweeper. FAF trains a tuned MLP on fixed multi-hop aggregations and also trails the GNN baselines on these two graphs (78.11 and 90.00 over official splits 0–2). Its authors note that the GNN baselines there use 10 and 15 layers (Rubio-Madrigal & Burkholz, 2026). GraphPack is above FAF on both graphs, and directed views appear in 0.79 of its selected members on roman-empire (Appendix E).

On amazon-ratings, Moscat reports $56.36 \pm 0.52$ over ten runs (Deng et al., 2025b), a tie with GraphPack's $56.16 \pm 0.58$.

On ogbn-arxiv under its official split, GraphPack reaches $72.77 \pm 0.15$ over three seeds. This is above the OGB entries of SIGN (71.95), GCN (71.74), and GraphSAGE (71.49), which use no label inputs (Hu et al., 2020; Frasca et al., 2020). Entries that combine label reuse with deep message passing are higher. The leaderboard entry that combines RevGAT with label reuse and self-distillation reaches 74.26, 1.49 points above GraphPack (Hu et al., 2020). Tuned classic GNNs without label inputs also exceed GraphPack (Luo et al., 2024).

### D.6 POKEC-REGIONS

pokec-regions is the largest of our development graphs, with 1.63M nodes and 183 classes. GraphPack runs it with the default configuration. Its bank occupies 21.7 GB in fp32 and is held on the GPU in fp16. The size and class rules activate validation subsampling and skip GBDT candidates (Appendices C.1 and B.6). A complete RH run takes 2.2 hours on average (8089 s over three seeds, about 141 s of it for the bank). It trains for 87 epochs and selects 5.7 members on average (Appendix E).

On RH, GraphPack scores 59.12 against the best published 50.06 (GT), a margin of 9.1 points. Label views supply 7.3 of them. Without label views GraphPack scores $51.83 \pm 1.74$, a tie with GT. Region labels are shared by neighbors in this social network. Edge homophily is 0.43 over 183 classes, and a vote over the training labels of one-hop neighbors alone reaches 51.5 on the RH test nodes, above every published label-free baseline. Label views appear in 0.88 of the selected members, whose mean receptive field is 2.8 hops. The members combine label views with features and multi-hop views to reach 59.12.

This graph is compared with published label-free baselines and the neighbor-label vote; the resource constraints on equal-information baselines are described in Appendix C.3. The margin persists on the temporal split, 37.62 against 25.09 (GT).

### D.7 HELD-OUT WEB GRAPH

The web graph supplies three held-out tasks with 2.9M nodes: web-fraud, web-topics, and web-traffic. Appendices C.1 and C.2 describe the data and evaluation protocol. Table 14 compares GraphPack with GraphLand's best published RH model for each task. GraphLand trains these models with hidden size 200 and reports five runs (Bazhenov et al., 2025).

Table 14 lists our runs. We report the mean of two runs of the frozen configuration per task. GraphPack reaches $19.85 \pm 0.28$ AP on web-fraud against $20.47 \pm 0.17$ for GraphLand's tuned GraphSAGE (a loss), $86.67 \pm 0.01$ $R^2$ on web-traffic against $85.70 \pm 0.08$ for GAT (a win), and $51.14 \pm 0.12$ accuracy on web-topics against $50.58 \pm 0.04$ for GT (a win). The default size rules place the banks in host memory, skip GBDT candidates, and score selection on $10^5$ validation nodes (Appendix B.6). Thus all three tasks evaluate the MLP pack alone. The web-fraud bank has 16,710 columns and occupies about 193 GB in fp32. The other tasks have similar bank widths. Table 14 reports bank size and run time.

Table 14: Held-out web tasks (RH split). The table reports split sizes, bank width, and bank construction and total run times (means over runs). GBDT is the maximum number of LightGBM members in the selected ensemble across runs. The table also lists the number of runs (seeds 0 and 1), GraphPack test scores (mean±std over its runs), and GraphLand's best published model (mean±std over five runs).

| Task | train / val / test | bank width | bank (s) | run (h) | GBDT | runs | GraphPack | best published |
|---|---|---|---|---|---|---|---|---|
| web-fraud (AP) | 1,444,671 / 722,585 / 723,075 | 16,710 | 417 | 4.1 | 0 | 2 | 19.85±0.28 | 20.47±0.17 SAGE |
| web-traffic ($R^2$) | 1,439,984 / 720,583 / 721,480 | 16,726 | 373 | 5.0 | 0 | 2 | 86.67±0.01 | 85.70±0.08 GAT |
| web-topics (Acc) | 1,067,615 / 534,291 / 534,140 | 14,080 | 310 | 3.0 | 0 | 2 | 51.14±0.12 | 50.58±0.04 GT |

## E ADDITIONAL ANALYSES AND RUN STATISTICS

This appendix completes the mechanism analysis of Section 5 (Appendices E.1 to E.3). It then reports the controls, sensitivity studies, and run statistics behind the fixed defaults (Appendices E.4 to E.6).

### E.1 PAIRWISE ERROR CORRELATION

An ensemble gains from members that make errors on different nodes. We measure which differences between two members predict this pattern. On each of the six datasets of Table 4, we train the *Both* arm with GBDT candidates off and offline selection (Appendix E.4). Offline selection keeps each member's test predictions from the epoch with its best validation score. We use three seeds per dataset, 18 runs in total. For each of the 2016 pairs of the 64 members of a run, we compute the Pearson correlation of their test residuals $f_i(v) - t(v)$, flattened over nodes and classes.

We relate this residual correlation to two distances between the members. The *structural distance* is the Jaccard distance of the two view sets. The *hyperparameter distance* is the mean absolute difference of five coordinates, each scaled to $[0, 1]$ over its sampling range: log learning rate, log Muon learning rate, log weight decay, dropout$/0.5$, and $(\text{depth} - 1)/2$. For every dataset and seed, we correlate residual correlation with each distance across the 2016 pairs and then average over seeds.

The structural distance correlates negatively with residual correlation on all six datasets. The Pearson $r$ is $-0.12$ on amazon-ratings, $-0.13$ on artnet-exp, $-0.14$ on city-roads-M, $-0.16$ on hm-categories, $-0.26$ on roman-empire, and $-0.38$ on tolokers-2. The hyperparameter distance gives $|r| \leq 0.07$ on five datasets. The exception is city-roads-M, where it reaches $r = -0.15$, against $-0.14$ for the structural distance. Mean residual correlation over the pairs of a run lies in 0.57–0.91 across the 18 runs. Averaged over seeds, it ranges from 0.59 on roman-empire to 0.90 on artnet-exp and tolokers-2. We summarize these associations descriptively because member pairs overlap within a run. Figure 3(b) shows the same relation in quintile bins, pooled over datasets and seeds.

To account for other differences between members, we fit an ordinary least-squares model for each run. The response is the residual correlation of a pair, and the model uses four standardized pre-

dictors. These are the structural distance, the hyperparameter distance, an indicator of whether exactly one member uses label views, and the absolute difference between the members' mean squared residuals. The structural coefficient is negative in 18 of 18 runs (mean $-0.011$). The hyperparameter coefficient is negative in 11 runs and averages $-0.0004$. The label-mismatch coefficient is negative in 9 runs (mean $-0.007$). The loss-gap coefficient has the largest mean magnitude ($-0.039$), so pairs whose losses differ more have less correlated residuals. With label views removed from the structural distance, its coefficient stays negative in 18 runs.

### E.2 Selection adapts label use

Greedy selection uses only validation scores and no dataset-specific rule. We test whether it keeps label views where they help, on the eleven GraphLand graphs. For each graph $d$, let $u_d$ be the fraction of selected MLP members whose view set contains a label view, pooled over the three seeds. Let $g_d$ be the gain of GraphPack over GraphPack with label views disabled (the no-lab column of Table 2). The Spearman correlation between $u_d$ and $g_d$ is $\rho = 0.84$ (one-sided permutation test with $10^5$ permutations, $p = 0.001$). GBDT members can also use label views. Including them in the count gives $\rho = 0.86$ ($p < 0.001$). On the eight graphs with at least five selected MLP members per run, $\rho = 0.81$.

The GraphLand graphs have no directed views, so a given seed draws the same 64 configurations on every graph. Label views are sampled with probability 0.5 (label budget 1 with probability 0.3, 2 with probability 0.2), and 51% of the sampled configurations contain one. Selection raises this share to 1.00 on city-roads-M and 1.00 on city-roads-L, where labels add 5.5 and 10.7 points. It reaches 0.91 on twitch-views and 0.88 on pokec-regions. On avazu-ctr, where labels add nothing, the share stays near the prior at 0.48, and on tolokers-2 (gain 0.1) it is 0.58. Three graphs have fewer than five selected MLP members per run, because GBDT members take a large share of their ensembles. For hm-categories (1.00), hm-prices (1.00), and artnet-views (0.83), the MLP shares are based on 3, 9, and 12 selected members over three seeds. The all-member analysis includes their selected GBDT members as well.

Selection also moves the other view families away from the prior. On graphs with at least five selected MLP members per run, we call a family enriched or depleted if its selected share differs from its sampled share by at least 0.1. Directed views are sampled with probability 0.6 when the edge set is asymmetric. On such graphs, direction-separated label views are added with probability 0.7 whenever directed views are drawn, so 70% of the sampled configurations on the heterophily graphs with directed views contain some label view. The sampled share of directed views on these graphs is 0.59. Selection raises it to 0.89 on roman-empire and keeps it at 0.56 on questions. It lowers it to 0.34 on amazon-ratings and 0.42 on minesweeper. High-pass and min/max views are each sampled with probability 0.4. Min/max views are enriched on artnet-exp, avazu-ctr, and roman-empire (0.54–0.56 against 0.38) and depleted on tolokers-2, city-reviews, pokec-regions, city-roads-L, minesweeper, and ogbn-arxiv (0.09–0.28). High-pass views are depleted on amazon-ratings (0.22 against 0.41), minesweeper (0.24 against 0.41), and ogbn-arxiv (0.14 against 0.43).

### E.3 Composition of the selected ensembles

Table 15 extends Figure 4 in Section 5 to all sixteen development datasets. It reports the fraction of selected members whose views include each family. These fractions count GBDT members, which train on random view subsets of the bank, so they differ from the MLP-only shares of Appendix E.2. Direction-separated label views count toward both *label* and *directed*. The table also gives the GBDT share of the selected members and the mean hop budget $k$ of the selected MLP members. The mix differs by dataset. GBDT members form 70% of the ensemble on hm-categories and 3% on minesweeper. Directed views appear in 0.79 of the members on roman-empire. The mean hop budget ranges from 1.67 on hm-categories to 4.50 on twitch-views and is 2.82 on pokec-regions.

### E.4 Online versus offline selection

Table 16 replaces online selection with one greedy pass after training. The pass selects from the best-validation checkpoint of every member and the finished GBDT candidates. The two procedures are close in accuracy. They differ by at most 0.47 points, and the gap exceeds the summed standard

Table 15: Selected-ensemble composition per dataset (headline runs on the RH and official splits, pooled over seeds or splits). View-family columns: fraction of selected members, MLP and GBDT, whose views include the family. GBDT%: share of GBDT members. Mean hop: mean hop budget $k$ of the selected MLP members.

| Dataset | sym | rw | high-pass | min/max | label | directed | GBDT% | mean hop |
|---|---|---|---|---|---|---|---|---|
| tolokers-2 | 0.64 | 0.62 | 0.43 | 0.21 | 0.52 | 0.00 | 21 | 2.97 |
| city-reviews | 0.54 | 0.57 | 0.36 | 0.27 | 0.62 | 0.00 | 16 | 2.95 |
| artnet-exp | 0.44 | 0.65 | 0.44 | 0.47 | 0.63 | 0.00 | 19 | 3.51 |
| hm-categories | 0.40 | 0.70 | 0.10 | 0.10 | 0.70 | 0.00 | 70 | 1.67 |
| pokec-regions | 0.65 | 0.88 | 0.41 | 0.24 | 0.88 | 0.00 | 0 | 2.82 |
| hm-prices | 0.65 | 0.76 | 0.35 | 0.12 | 0.71 | 0.00 | 47 | 2.67 |
| avazu-ctr | 0.67 | 0.85 | 0.39 | 0.48 | 0.42 | 0.00 | 24 | 3.64 |
| city-roads-M | 0.59 | 0.68 | 0.44 | 0.32 | 0.85 | 0.00 | 24 | 3.00 |
| city-roads-L | 0.72 | 0.59 | 0.41 | 0.26 | 0.92 | 0.00 | 15 | 4.00 |
| twitch-views | 0.89 | 0.86 | 0.49 | 0.32 | 0.81 | 0.00 | 14 | 4.50 |
| artnet-views | 0.67 | 0.57 | 0.33 | 0.24 | 0.62 | 0.00 | 43 | 3.17 |
| roman-empire | 0.59 | 0.53 | 0.41 | 0.57 | 0.76 | 0.79 | 11 | 3.51 |
| amazon-ratings | 0.48 | 0.67 | 0.24 | 0.42 | 0.59 | 0.24 | 28 | 3.62 |
| minesweeper | 0.52 | 0.60 | 0.24 | 0.28 | 0.83 | 0.40 | 3 | 2.77 |
| questions | 0.62 | 0.56 | 0.40 | 0.29 | 0.68 | 0.52 | 7 | 3.18 |
| ogbn-arxiv | 0.62 | 0.50 | 0.15 | 0.08 | 0.62 | 0.00 | 15 | 3.00 |

deviations only on city-roads-M ($0.26$ in favor of online selection). The wall-clock comparison gives mixed results. Online selection is faster on four datasets and slower on the other two. We keep online selection as the default because it stores test predictions only for the selected snapshots (Section 3.4), while the offline pass stores them for every member. The pairwise analysis in Appendix E.1 needs this full pool and therefore uses offline runs.

Table 16: Online (default) versus offline selection: test score (mean±std over seeds) and training wall-clock in seconds. For amazon-ratings and roman-empire, the online column averages the ten official splits. The offline column averages splits 0–2. On splits 0–2, online selection scores $56.43$ and $85.66$.

| Dataset | Online (default) | Time (s) | Offline | Time (s) |
|---|---|---|---|---|
| amazon-ratings | 56.16±0.58 | 394 | 56.23±0.20 | 309 |
| artnet-exp | 53.62±0.28 | 112 | 53.60±0.19 | 91 |
| city-roads-M | 78.00±0.09 | 122 | 77.74±0.07 | 143 |
| hm-categories | 82.67±0.59 | 257 | 82.38±0.68 | 421 |
| roman-empire | 85.69±0.51 | 413 | 85.51±0.35 | 643 |
| tolokers-2 | 64.42±0.57 | 29 | 64.89±0.62 | 33 |

### E.5 Budget sensitivity and slot refilling

Figure 5 and Table 17(a)–(b) show the effects of varying the pack size $m$ and hop budget $K$ on two datasets, city-roads-M and tolokers-2. Each non-default setting uses two seeds. Across $m \in \{16, 32, 64, 128\}$, the mean score spans $0.54$ points on city-roads-M, where it grows with $m$. It spans $0.97$ points on tolokers-2, where $m$=128 is lowest. Across $K \in \{2, 4, 6, 8\}$, it spans $0.29$ and $0.36$ points. No tested setting moves the mean score by more than $0.68$ points from the default on these two datasets.

Table 17(c) evaluates slot refilling. Slot refilling replaces each early-stopped member with a newly sampled configuration, up to $m$ refills per run. The new member joins the candidate pool in the next epoch. Refilled members enter the selected ensemble in all eight refill runs, yet accuracy does not improve. We compare refilling with the default runs on the same seeds (on amazon-ratings, the same official splits). Refilling scores $0.27$–$0.45$ points lower on amazon-ratings, artnet-exp, and tolokers-2 and higher by $0.07$ on city-roads-M. Refilling is therefore disabled by default.

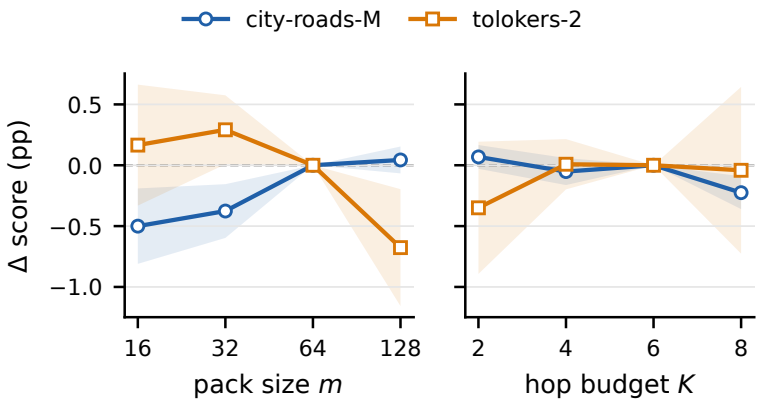


Figure 5: Sensitivity to pack size $m$ and hop budget $K$ on city-roads-M and tolokers-2: mean score change relative to the defaults $m$=64 and $K$=6. Shaded bands show one standard deviation over the two seeds of each non-default setting.

Table 17: Budget sensitivity on two datasets and slot refilling on four (test score, mean±std over seeds). Non-default settings use two seeds and refill runs two. Default columns use all headline runs, which on amazon-ratings are the ten official splits. On the seeds of the refill runs, 0 and 1 (official splits on amazon-ratings), the default scores 56.76 on amazon-ratings, 53.47 on artnet-exp, 77.95 on city-roads-M, and 64.30 on tolokers-2.

**(a) Pack size $m$**

| Dataset | $m$=16 | $m$=32 | $m$=64 (default) | $m$=128 |
|---|---|---|---|---|
| city-roads-M | 77.50±0.31 | 77.62±0.22 | 78.00±0.09 | 78.04±0.11 |
| tolokers-2 | 64.58±0.50 | 64.71±0.28 | 64.42±0.57 | 63.74±0.48 |

**(b) Hop budget $K$**

| Dataset | $K$=2 | $K$=4 | $K$=6 (default) | $K$=8 |
|---|---|---|---|---|
| city-roads-M | 78.07±0.10 | 77.95±0.11 | 78.00±0.09 | 77.77±0.13 |
| tolokers-2 | 64.07±0.54 | 64.42±0.21 | 64.42±0.57 | 64.37±0.69 |

**(c) Slot refilling**

| Dataset | No refill (default) | With refill |
|---|---|---|
| amazon-ratings | 56.16±0.58 | 56.33±0.28 |
| artnet-exp | 53.62±0.28 | 53.20±0.09 |
| city-roads-M | 78.00±0.09 | 78.02±0.17 |
| tolokers-2 | 64.42±0.57 | 63.85±0.34 |

### E.6 Complete run statistics

Table 18 gives the bank width, stopping epoch, selected ensemble size, number of GBDT members, and wall-clock time of the headline runs on the RH and official splits of all sixteen development datasets. Total time includes the precomputation of the propagation bank. The selected ensemble ranges from 3.3 members on hm-categories to 31.1 on questions, close to the cap $m_{\text{ens}}$=32. Among these datasets, runs take longest on pokec-regions, with 8089 s (2.2 hours) on average. Table 14 gives the run times of the three web tasks.

Table 18: Run statistics of the headline runs on the RH and official splits (means over seeds or splits). $|\mathcal{M}|$: selected ensemble size, including GBDT members. †GBDT candidates are skipped on pokec-regions because its 183 classes exceed the cap of 32 classes.

| Dataset | Bank dim | Epochs | $\lvert\mathcal{M}\rvert$ | GBDT | Train (s) | Total (s) |
|---|---|---|---|---|---|---|
| tolokers-2 | 492 | 59 | 14.0 | 3.0 | 29 | 30 |
| city-reviews | 3732 | 53 | 24.7 | 4.0 | 540 | 550 |
| artnet-exp | 2328 | 51 | 14.3 | 2.7 | 112 | 115 |
| hm-categories | 2276 | 69 | 3.3 | 2.3 | 257 | 279 |
| pokec-regions | 3320 | 87 | 5.7 | 0.0† | 7930 | 8089 |
| hm-prices | 4954 | 123 | 5.7 | 2.7 | 325 | 368 |
| avazu-ctr | 4684 | 57 | 11.0 | 2.7 | 239 | 299 |
| city-roads-M | 1732 | 70 | 13.7 | 3.3 | 122 | 123 |
| city-roads-L | 2596 | 70 | 13.0 | 2.0 | 313 | 316 |
| twitch-views | 562 | 73 | 12.3 | 1.7 | 236 | 247 |
| artnet-views | 1876 | 53 | 7.0 | 3.0 | 80 | 82 |
| roman-empire | 7876 | 60 | 9.8 | 1.1 | 413 | 416 |
| amazon-ratings | 7824 | 62 | 9.0 | 2.5 | 394 | 398 |
| minesweeper | 194 | 78 | 14.1 | 0.4 | 34 | 34 |
| questions | 7838 | 54 | 31.1 | 2.1 | 351 | 362 |
| ogbn-arxiv | 2386 | 52 | 8.7 | 1.3 | 717 | 730 |

# F Complete Per-Run Results

Table 19 lists every individual run behind each headline GraphPack mean: per-seed scores for GraphLand (RH/RL/TH), ogbn-arxiv, and the three held-out web tasks, and per-official-split scores for the heterophily suite.

Table 19: Individual run scores behind each headline GraphPack mean. $n$: number of runs.

| Dataset (split) | Individual runs | Mean±std | $n$ |
|---|---|---|---|
| tolokers-2 (RH) | 63.77, 64.64, 64.83 | 64.42±0.57 | 3 |
| city-reviews (RH) | 82.51, 82.52, 82.61 | 82.55±0.06 | 3 |
| artnet-exp (RH) | 53.40, 53.53, 53.93 | 53.62±0.28 | 3 |
| hm-categories (RH) | 82.29, 82.37, 83.35 | 82.67±0.59 | 3 |
| pokec-regions (RH) | 59.11, 59.13, 59.13 | 59.12±0.01 | 3 |
| hm-prices (RH) | 82.81, 83.03, 83.46 | 83.10±0.33 | 3 |
| avazu-ctr (RH) | 36.25, 36.57, 36.61 | 36.48±0.20 | 3 |
| city-roads-M (RH) | 77.92, 77.98, 78.10 | 78.00±0.09 | 3 |
| city-roads-L (RH) | 76.80, 76.97, 76.98 | 76.91±0.10 | 3 |
| twitch-views (RH) | 77.85, 78.06, 78.51 | 78.14±0.34 | 3 |
| artnet-views (RH) | 63.59, 63.69, 63.95 | 63.74±0.19 | 3 |
| roman-empire (official) | 85.14, 85.19, 85.21, 85.47, 85.51, 85.53, 85.70, 86.25, 86.30, 86.55 | 85.69±0.51 | 10 |
| amazon-ratings (official) | 55.19, 55.50, 55.77, 56.03, 56.18, 56.30, 56.31, 56.43, 56.67, 57.23 | 56.16±0.58 | 10 |
| minesweeper (official) | 91.42, 91.76, 91.83, 91.87, 92.07, 92.44, 92.71, 92.80, 92.97, 93.15 | 92.30±0.59 | 10 |
| questions (official) | 77.53, 77.54, 77.57, 77.80, 78.03, 78.17, 78.43, 78.53, 79.65, 79.71 | 78.30±0.81 | 10 |
| ogbn-arxiv (official) | 72.61, 72.79, 72.91 | 72.77±0.15 | 3 |
| tolokers-2 (RL) | 58.27, 58.85, 59.24 | 58.79±0.48 | 3 |
| city-reviews (RL) | 79.09, 79.24, 79.41 | 79.25±0.16 | 3 |
| artnet-exp (RL) | 47.91, 48.46, 48.97 | 48.45±0.53 | 3 |
| hm-categories (RL) | 66.62, 67.75, 70.14 | 68.17±1.80 | 3 |
| pokec-regions (RL) | 53.09, 53.50, 53.54 | 53.37±0.25 | 3 |
| hm-prices (RL) | 71.67, 72.15, 73.05 | 72.29±0.70 | 3 |
| avazu-ctr (RL) | 33.69, 34.13, 34.15 | 33.99±0.26 | 3 |
| city-roads-M (RL) | 63.90, 64.25, 64.27 | 64.14±0.21 | 3 |
| city-roads-L (RL) | 59.24, 59.51, 59.69 | 59.48±0.23 | 3 |
| twitch-views (RL) | 75.21, 75.34, 75.79 | 75.45±0.30 | 3 |
| artnet-views (RL) | 60.77, 60.83, 61.50 | 61.03±0.40 | 3 |
| tolokers-2 (TH) | 45.86, 46.91, 50.69 | 47.82±2.54 | 3 |
| artnet-exp (TH) | 42.85, 43.60, 43.93 | 43.46±0.55 | 3 |
| artnet-views (TH) | 56.20, 56.59, 56.62 | 56.47±0.24 | 3 |
| hm-categories (TH) | 58.67, 62.03, 63.16 | 61.29±2.34 | 3 |
| hm-prices (TH) | 67.63, 68.54, 69.31 | 68.49±0.84 | 3 |
| avazu-ctr (TH) | 40.01, 40.03, 40.16 | 40.07±0.08 | 3 |
| twitch-views (TH) | 69.47, 69.54, 70.19 | 69.73±0.39 | 3 |
| pokec-regions (TH) | 37.52, 37.64, 37.70 | 37.62±0.09 | 3 |
| web-fraud (held-out) | 19.65, 20.04 | 19.85±0.28 | 2 |
| web-traffic (held-out) | 86.67, 86.68 | 86.67±0.01 | 2 |
| web-topics (held-out) | 51.05, 51.22 | 51.14±0.12 | 2 |

# G Evaluation Scope and Limitations

**Development and held-out evaluation.** The configuration space, sampling distributions, and defaults were developed on the eleven GraphLand graphs, the heterophily suite, and ogbn-arxiv. The web evaluation applies the same default configuration to one new graph with three tasks from the same benchmark. Transfer across a broader range of data sources remains to be evaluated. The sensitivity study varies the pack size and hop budget on city-roads-M and tolokers-2. Other sampling probabilities retain their defaults.

**Inputs and baseline coverage.** The equal-information comparison gives all pipelines the features, graph, and training labels and evaluates each workflow with the model space and tuning budget specified in Appendix C.3. That appendix also documents memory-driven substitutions and the daggered published results. The mechanism study holds the bank, trainer, and selection fixed to isolate the sampled axes within the MLP pack.

Equal-information baselines cover the ten GraphLand RH datasets of Table 2. The no-label variant additionally separates label reuse on all eleven RH datasets and on seven of the eight TH datasets. On RL, the heterophily suite, and the web tasks, comparisons use the published model inputs; no equal-information or no-label reruns are available for those settings. The margins over published label-free models in these comparisons combine label reuse with the other components of GraphPack.

**Statistical interpretation.** Appendix C.6 specifies the run counts and how results are aggregated. Sample standard deviations describe variation over runs or official splits. Their precision is limited by the three runs on GraphLand and two runs per web task. GraphLand's summed-standard-deviation tie rule is a descriptive convention, not a test with a controlled error rate. Accordingly, the paper reports per-dataset means, win/tie/loss counts, and gaps to the best pipeline as descriptive comparisons. For the six-dataset mechanism study, paired-run counts describe consistency across seeds. The label-use correlation is an exploratory association across the eleven GraphLand datasets, two pairs of which share a graph. Every run behind a headline GraphPack mean is listed in Appendix F.

**Comparisons across implementations.** Published results use their source preprocessing, model configurations, and hardware. GraphLand trains full-batch GNNs on A100 80 GB GPUs and reduces hidden width on its largest graphs. Our training GPUs have 24 GB. The fidelity study in Appendix C.3 compares our runs of GraphLand's code with its published scores. The tables preserve each source's run counts and split protocol. For example, FAF uses three official heterophily splits, whereas the main heterophily comparison uses ten.

**Cost accounting.** The primary cost comparison uses the same bank construction, trainer, and GPU for GraphPack and the SIGN search. A GraphPack run includes bank construction and uses one GPU plus 16 CPU threads for concurrent LightGBM training when candidates are enabled. Search time sums 50 sequential trial training times and excludes bank construction and reruns. Thus the ratios compare wall-clock time, while CPU use differs between workflows. Figure 1(b) reports performance throughout the 50-trial search. Run times include the contention described in Appendix C.7 and vary with stopping epochs. The message-passing search times from the external study in Appendix D.2 provide context. That study uses different hardware and a different split protocol. Warm-inference timings measure only forward passes through the specified deployed architectures, excluding preprocessing and training. Appendix C.8 gives the timing protocol.

**Validation under temporal shift.** Member early stopping and ensemble selection both use validation scores. Under TH, this period precedes the test period, and label views use only training-period labels. Appendix D.3 reports the observed validation–test gaps and changes in label coverage that characterize this shift.

**Task scope.** The experiments cover transductive node classification and regression on graphs with node features. The bank is computed on the full graph from the training labels. New nodes or labels require recomputing the affected views.