COMPILED TASK MEMORY FOR EFFICIENT INFERENCE WITH FROZEN TABULAR FOUNDATION MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Context optimization offers a way to reduce the inference cost of tabular foundation models by replacing large training contexts with compact learned representations. Building on this approach, we study *compiled task memory*: task-specific context vectors optimized offline and reused through a frozen predictor. Slots are initialized from stratified frozen row encodings and optimized using supervised loss and out-of-fold teacher guidance, with all backbone weights held fixed. Our study evaluates accuracy, retained state, and cached inference cost on TabDPT and TabICLv2 across 35 classification and 13 regression tasks, using complete training and test splits and three seeds. Compared with the same unoptimized 256-slot initializations, optimization improves mean classification accuracy by 2.14 and 2.55 percentage points and reduces geometric-mean RMSE by 19.2% and 17.6%, respectively, with essentially unchanged compact inference cost. On classification tasks, cached 256-slot inference achieves geometric-mean speedups of $3.30\times$ and $1.17\times$ over full cached context, with mean accuracy losses of 0.76 and 0.78 percentage points. One-time preparation costs are reported separately. Construction and teacher-weight ablations characterize how memory learning affects quality at fixed capacity. Across $K = 64$–1024, larger memories improve regression at higher serving and storage cost, while classification gains beyond 256 slots remain uncertain. These findings characterize the practical accuracy–cost tradeoffs of context optimization.

1 INTRODUCTION

Tabular foundation models (TFMs) predict from labeled context rows through a shared pretrained network (Hollmann et al., 2023; Ma et al., 2025; Qu et al., 2026). We aim to compile a task's training examples once into K learned context vectors, then reuse this fixed-size memory for new queries. The shared predictor remains frozen. We study whether this preparation reduces inference cost and retained task state while preserving predictive accuracy.

Contexts may contain selected rows or optimized representations (Feuer et al., 2024; Ma et al., 2024). Teacher supervision is established: TuneTables explores Kullback–Leibler (KL) divergence from PFN outputs, and out-of-fold (OOF) distillation excludes a teacher query's context label (Tanna et al., 2026). The practical question is how much accuracy task-loss optimization recovers at a fixed number of prediction slots, beyond choosing or pooling frozen encoded rows, and what it costs relative to serving the complete training context.

We call the optimized context a *compiled task memory*: task-specific representations served through a shared frozen predictor. We compare construction rules at K256 across all 48 tasks, evaluate teacher weights at K256/512/1024 on a common eligible cohort, and measure quality and serving cost across slot budgets. Our empirical results are:

- **Accuracy at a fixed slot budget.** Optimized256 improves over its exact Unoptimized256 initialization on two backbones. That initialization is already a stratified frozen-encoding baseline. At the same K256 budget, cluster/PCA representatives and fixed-anchor Hopfield pooling provide additional construction controls on all 48 tasks; the K1024 construction follow-up is in Appendix B.6.

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.

- **Quality and serving across complete datasets.** Uncapped comparisons on 35 classification and 13 regression tasks measure the recovered quality and remaining gaps to full context. Every test row enters inference measurements, and all arms use key–value (KV) caches. At fixed K, optimization changes accuracy with essentially unchanged serving cost.
- **Capacity and preparation tradeoffs.** Slot counts of 64/256/512/1024 expose the accuracy, state and latency tradeoff. Larger memories improve regression but retain a mean gap to full context. Offline preparation is accounted for separately from repeated query serving.

The comparisons assess fixed training recipes and compact serving, without establishing a universally optimal slot count or unique distillation mechanism.

2 Related work

Context optimization and distillation. TuneTables optimizes embedding prompts through TabPFN, evaluates prompt-only and prompt-plus-row serving, and explores KL supervision from PFN predictions (Feuer et al., 2024). ICD optimizes feature-space contexts through a fixed TabPFN (Ma et al., 2024). OOF teacher labeling and mixed supervised/distillation losses also appear in TFM-to-student distillation (Tanna et al., 2026). Our study builds on these learning principles: it measures optimization gains at matched capacity on TabDPT/TabICLv2 and compares compact serving with uncapped, KV-cached full context. Table 10 distinguishes these antecedents from our evaluated controls. The TuneTables entries use adapted single prompts on TabPFN-v1, without its full search/ensembles; ICD is related work, not a reproduced baseline. The paired $\lambda = 0$ arm instead provides supervised context optimization on each evaluated backbone (Section 5.4).

Memory construction and compression. Context selection and retrieval construct prediction contexts from retained task data (Thomas et al., 2024). Our geometric controls instead select encoded rows once and retain only the compact serving artifact and its required fitted state. Modern Hopfield networks provide an attention-like associative update (Ramsauer et al., 2021); our baseline uses fixed stratified queries for a single offline pooling step. It does not train a condenser. TACO jointly trains a compressor and predictor (Zabërgja et al., 2026); we freeze the predictor. Dataset distillation supplies the synthetic-data perspective (Wang et al., 2018; Cazenavette et al., 2022), while hypernetworks and distilled tree students offer alternatives to retaining the TFM interface (Müller et al., 2025; Bonet et al., 2024; Tanna et al., 2026).

3 Method

Figure 1 summarizes the two stages: offline optimization of task-specific slots through a frozen predictor, followed by cached inference with the compiled memory. The same backbone supplies out-of-fold teacher targets during preparation and predictions during serving. Sample code for compiling task memories and generating predictions is provided in the supplementary material.

3.1 Frozen context optimization

Let f_θ be a frozen predictor and $p_S(\cdot \mid x)$ its prediction with context $S \in \mathbb{R}^{K \times d}$. The K context vectors are the task-specific learnable parameters. TabDPT supplies fixed slot labels ℓ separately; TabICLv2 starts from vectors that already contain label encodings. Preprocessing and architecture-specific caches remain part of the serving artifact (Appendix A.2). No backbone parameter changes.

We initialize S from K distinct training rows, sampled within class strata for classification or target-quantile strata for regression. The selected full-pool frozen encodings and labels form Unoptimized$K$; Optimized$K$ starts from exactly those values. For teacher guidance, partition the training pool into two folds. Each query row i receives a probability vector

$$q_i = p_\theta(\cdot \mid x_i, \mathcal{D}_{\text{other fold}}). \tag{1}$$

The teacher's context excludes the queried row's label, and each fold fits preprocessing only on its context rows. Stored log probabilities are exponentiated when evaluating the divergence. For classification, we minimize

$$\mathcal{L}(S) = \frac{1}{|B|} \sum_{i \in B} \left[-\log p_S(y_i \mid x_i) + \lambda \, \mathrm{KL}(q_i \| p_S(\cdot \mid x_i)) \right], \qquad \lambda = 1 \text{ by default.} \tag{2}$$

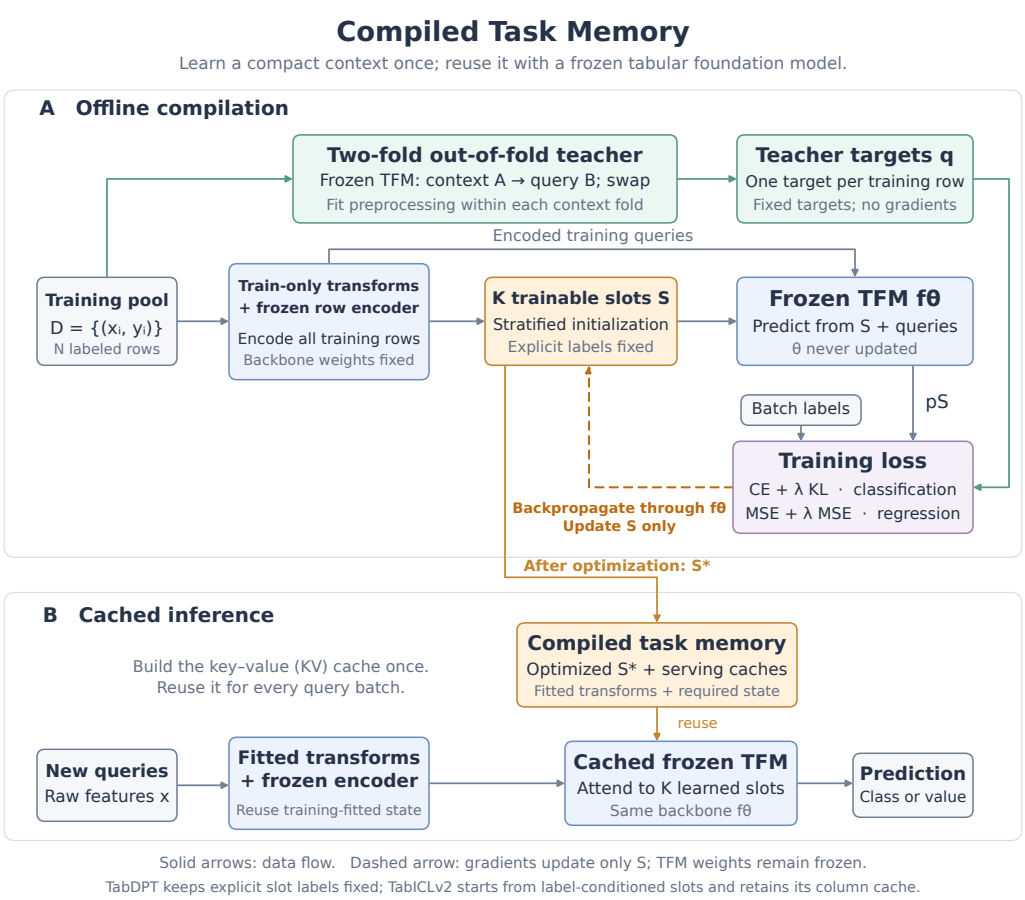


Figure 1: **Compiled task memory: offline learning and cached inference.** (A) Frozen training-row encodings initialize K stratified slots. Labels and two-fold out-of-fold teacher targets guide updates to S; gradients pass through the frozen predictor while its weights remain unchanged. (B) Optimized slots, fitted preprocessing and required caches form a reusable task artifact. New queries use the frozen backbone and cached K-slot context. Explicit TabDPT slot labels remain fixed; TabICLv2 retains its column cache.

CE denotes cross-entropy, the supervised first term; CE+KL adds teacher matching. Regression instead uses MSE to standardized training labels plus λ times MSE to OOF teacher native-mean predictions. Slot labels remain fixed, and gradients update only S.

All primary optimized memories use 100 Adam steps, initial learning rate 0.03 with cosine decay to 0.0015, norm-one gradient clipping and minibatches of at most 512 training queries sampled with replacement. The complete training pool is available to teacher construction and sampling; 100 updates need not visit every row in a large dataset. The schedule is fixed across K and does not establish convergence. Full-context accuracy is an empirical reference, not a ceiling. Alternative constructors receive neither task-loss updates nor teacher targets (Section 5.3).

3.2 Cached serving and cost accounting

For each task, preparation produces fitted preprocessing, compact vectors, fixed labels where required, and the backbone's serving caches. Queries pass through raw-feature preprocessing and the frozen row encoder, then attend to the retained context. Both compact and full-context references cache transformer keys and values. TabICLv2 also retains its required column-embedding cache. The frozen checkpoint is shared across tasks and excluded from retained task state; all task-specific serving components are included.

Reducing K decreases the context cache and query-to-context attention cost. Row encoding, feed-forward work and preprocessing still contribute to inference time, so state savings need not translate proportionally to latency. At a fixed K, optimized and unoptimized memories have identical cache tensor shapes. Any quality gain from optimizing the vectors is distinct from the efficiency gain of using fewer context positions.

We report preparation, retained state and raw-query inference separately. Preparation can use the full training data and OOF teachers; the compact serving artifact discards construction-only row banks. Its encodings still derive from real training rows, so compactness does not establish privacy.

4 EXPERIMENTAL PROTOCOLS

Data and metrics. We use official fold-0 train/test assignments for 35 classification tasks from TabArena v0.1 (Erickson et al., 2025), restricted to 2–10 classes and at least 600 training rows, and all 13 official regression tasks. The primary studies use every training and test row, with no 16,000-row training cap or fixed 1,000-query workload. TFM category vocabularies, numerical medians and PCA above 100 encoded features are fitted only on the permitted training pool. Unseen categories have an explicit unknown code. TFM ordinal/PCA preprocessing differs from native benchmark pipelines; the broader baselines use method-specific transforms (Section 5.5). These are within-study comparisons, not leaderboard claims.

ROC-AUC is binary AUC or macro one-versus-rest AUC. Accuracy is the fraction of correct argmax predictions. Regression reports R^2 and RMSE normalized by the training target scale (nRMSE). We average seeds within each dataset and then weight datasets equally. Unless stated otherwise, paired 95% intervals use 10,000 dataset-bootstrap resamples with seed 2027 and are unadjusted across comparisons. Speed and RMSE ratios use geometric means. Datasets are the independent units; three seeds do not triple the sample size. An interval spanning zero does not establish equivalence.

Slot budgets and cohorts. We compare $K = 64, 256, 512, 1024$, the corresponding unoptimized initializations and full KV-cached context on TabDPT and TabICLv2, with seeds 41–43. K1024 requires 1,024 distinct training rows, leaving 30 classification and nine regression tasks. The smaller budgets cover all 48 tasks. Four-K aggregate comparisons use the same eligible tasks at every K; all-task K256 summaries remain separate. Verified K256 memories, teachers and full-training encodings are reused exactly. Appendix B.2 records eligibility and paired differences.

Construction and teacher ablations. The main K256 construction study covers all 35 classification and 13 regression tasks: three fixed constructors produce 864 memories across 288 dataset–backbone–seed conditions. Exact unoptimized, optimized and full-context controls are reused; all six arms are timed together. The K1024 construction follow-up covers 30/9 tasks (Appendix B.6). Teacher-weight sweeps at K256/512/1024 use that same 30/9-task cohort, adding 3,510 memories while reusing each $\lambda = 1$ arm. No backbone or teacher is refitted. All settings precede test evaluation (Appendices B.3 and B.5).

Timing and verification. TFMs run on an NVIDIA GB10 with ten CPU threads: TabDPT in FP32 and TabICLv2 in native FP16. Each timed call processes the complete official test split once, in chunks of at most 1,024 rows including the actual final remainder. Three warmups precede ten synchronized calls with rotating arm order, including raw transforms and returned CPU predictions. No competing benchmark jobs run during these TFM timing measurements. All full-context speedups and slot-budget results use the joint four-budget timing run; construction tables use their separate matched runs. Reused preparation components are identified as reconstructed costs; artifact restoration is excluded. Independent scoring and exact saved-memory prediction replays cover every condition. Training-only numerical checks and the retained FP16 discrepancy are documented in Appendix B.3; source and checkpoint identities are in Appendix A.1.

5 RESULTS

5.1 QUALITY AND SERVING WITH COMPLETE TRAINING AND TEST SPLITS

Optimization improves the compact baseline. Across all 35 classification tasks, optimized256 gains 3.80 AUC points ($[2.72, 5.10]$) on TabDPT and 3.35 ($[2.01, 4.81]$) on TabICLv2 over unopti-

Table 1: Slot budget, quality and inference cost on fixed cohorts: 30 classification and nine regression tasks, three seeds. Each optimized memory uses 100 updates; unoptimized256 retains 256 stratified frozen encodings without slot updates. Full context uses all training rows. Time is milliseconds per actual test row from complete-test calls. Scores and times average seeds within dataset, then weight datasets equally. nRMSE uses the training target scale.

	TabDPT			TabICLv2		
Context	AUC (%)	Acc. (%)	Time (ms)	AUC (%)	Acc. (%)	Time (ms)
Classification: same 30 tasks at every budget						
Unoptimized256	78.94	86.77	0.0315	79.95	86.29	0.0600
Optimized64	82.77	88.42	0.0302	83.62	88.80	0.0600
Optimized256	83.09	88.64	0.0315	83.75	88.87	0.0604
Optimized512	83.21	88.66	0.0338	84.02	88.83	0.0605
Optimized1024	83.22	88.61	0.0367	84.07	88.85	0.0606
Full context	84.67	89.24	0.1471	84.84	89.45	0.0672
Context	R^2	nRMSE	Time (ms)	R^2	nRMSE	Time (ms)
Regression: same nine tasks at every budget						
Unoptimized256	0.610	0.590	0.0218	0.626	0.569	0.0383
Optimized64	0.686	0.521	0.0205	0.705	0.496	0.0383
Optimized256	0.716	0.486	0.0219	0.729	0.472	0.0383
Optimized512	0.725	0.477	0.0235	0.739	0.461	0.0384
Optimized1024	0.731	0.470	0.0269	0.746	0.456	0.0388
Full context	0.777	0.428	0.1232	0.775	0.428	0.0446

mized256; accuracy gains are 2.14 and 2.55 points. Each backbone wins on 32/35 tasks for both metrics. On 13 regression tasks, mean R^2 increases by 0.081 and 0.079, with geometric-mean RMSE reductions of 19.2% and 17.6%. The compact arms have identical KV tensor sizes and essentially identical inference time. Appendix B.1 retains the complete K256 controls.

Full context retains higher mean quality. On all 35 tasks, optimized256-minus-full AUC is -1.57 points ($[-2.66, -0.81]$) for TabDPT and -1.16 ($[-1.91, -0.52]$) for TabICLv2; accuracy gaps are -0.76 and -0.78 points. On 13 regression tasks, R^2 gaps are -0.046/-0.035, with RMSE 13.4%/10.7% higher. Table 1 and Figure 2 compare all four budgets on the same 30 classification and nine regression tasks eligible for K1024; these cohort means are distinct from the all-task K256 results.

Serving savings across context sizes and slot budgets. Figure 3 plots all four optimized budgets against complete training size, using fresh matched timings on every actual test row. At K256, all-task geometric speedups are $3.30\times$/$1.17\times$ for classification and $3.20\times$/$1.19\times$ for regression (TabDPT/TabICLv2). Larger K reduces the serving speedup, especially for TabDPT. Preprocessing and required caches remain included; one-time preparation is separate. Cross-dataset trends do not isolate a within-task context-size effect.

5.2 Effect of the number of slots

Number of slots on fixed cohorts. Relative to K256, K1024 raises mean AUC by $+0.12$/$+0.32$ points (TabDPT/TabICLv2), but both 95% intervals include zero. Regression R^2 improves by $+0.015$/$+0.016$, with positive intervals for both K512 and K1024; K64 lowers regression quality. K1024 still trails full context in mean quality. Appendix B.2 gives paired intervals and state costs.

Implications for choosing K. Under this 100-update schedule, K256 is a reasonable starting point for classification: larger memories have no resolved mean AUC or accuracy gain. For regression, K1024 offers additional quality if the serving budget permits; relative to K256, retained state grows $3.40\times$/$2.73\times$ and latency $1.26\times$/$1.02\times$ (TabDPT/TabICLv2). Choose K using training-side validation and deployment constraints. These aggregate findings do not establish convergence, equivalence between classification budgets, or a universally optimal K; no default is reselected from test scores.

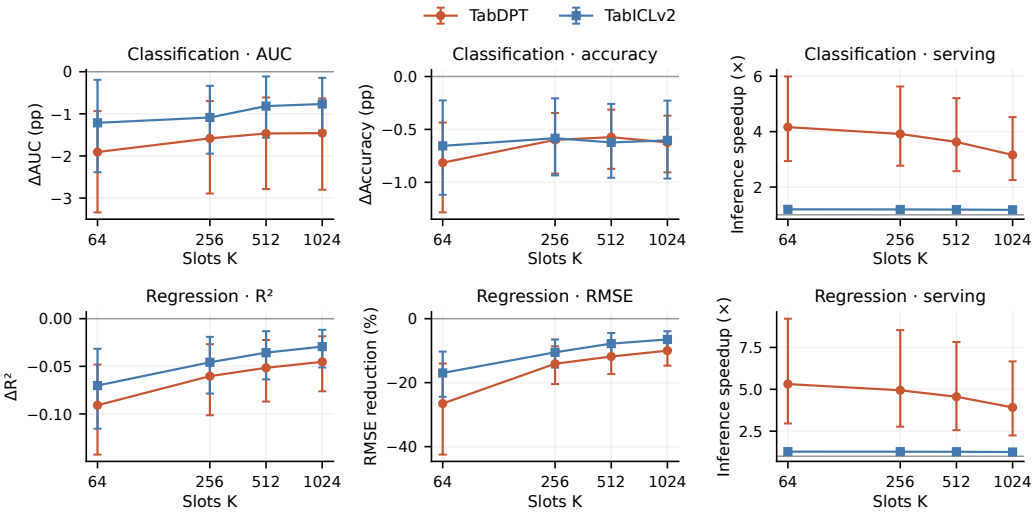


Figure 2: **Optimized K-slot memory versus full context, K=64/256/512/1024.** All four budgets use the same 30 classification and nine regression tasks, three seeds and 100 updates. Bars are paired 95% dataset-bootstrap intervals. Quality differences are optimized minus full; RMSE reduction is $100(1 - \mathrm{RMSE}_K/\mathrm{RMSE}_{\mathrm{full}})$. Speedup is full/optimized complete-test time; speed and error ratios use geometric means. Full context retains every training row.

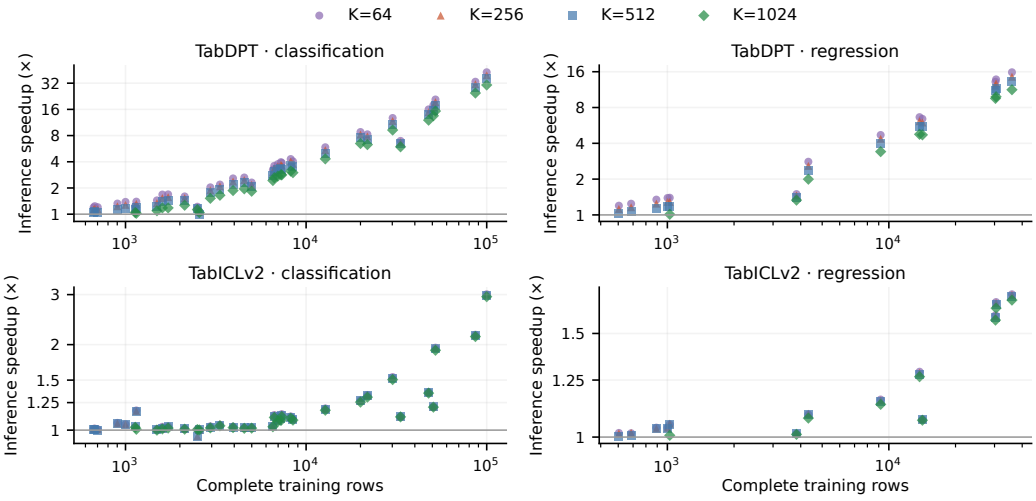


Figure 3: **Inference scaling at four slot budgets.** Each point is a dataset's geometric mean of three paired full/optimized complete-test time ratios. K64/256/512 cover all 35 classification and 13 regression tasks; K1024 covers the eligible 30/nine. Every actual test row is processed once per call. Colors and markers identify K; the $1\times$ line marks equal time. All arms are measured together with raw-query preprocessing and serving caches. Axes are logarithmic; these are cross-dataset comparisons.

5.3 Memory-construction ablation at 256 slots

Matched construction at K256. Table 2 compares the exact stratified frozen-encoding initialization, cluster representatives, PCA-guided representatives and fixed-anchor Hopfield pooling with Optimized256 and full context on all 35 classification and 13 regression tasks, with three seeds. Each constructor uses complete training encodings and the same class or target-quantile quotas, without

task-loss updates or teacher targets. Cluster/PCA methods select distinct real rows; Hopfield pooling uses one fixed-query lookup. Appendix B.3 gives settings; Appendix B.6 reports K1024 on its smaller eligible cohort.

Table 2: **Memory-construction ablation at $K = 256$.** All 35 classification and 13 regression tasks, three seeds. Unoptimized256 is the existing stratified frozen-encoding baseline, reused exactly. Other compact arms retain 256 slots; full context retains all training rows. Scores and times average seeds within dataset, then datasets equally. nRMSE uses the training target scale. Inference time is ms per actual test row, measured afresh together for all six arms; C/R denote classification/regression. This all-task cohort differs from the 30/9-task cohort in Table 1. Construction timings use a separate six-arm run. Bold identifies Optimized256.

Construction	AUC (%)	ACC (%)	R^2	nRMSE	C ms/row	R ms/row
TabDPT						
Unoptimized256	79.95	85.60	0.674	0.534	0.0308	0.0226
Cluster representatives	80.83	85.90	0.676	0.531	0.0308	0.0225
PCA-guided representatives	80.51	85.89	0.672	0.534	0.0309	0.0225
Fixed-anchor Hopfield	76.95	70.91	0.502	0.664	0.0306	0.0225
Optimized256	**83.75**	**87.74**	**0.755**	**0.449**	**0.0306**	**0.0225**
Full context	85.32	88.50	0.801	0.401	0.1301	0.0941
TabICLv2						
Unoptimized256	81.00	85.45	0.687	0.517	0.0560	0.0360
Cluster representatives	82.10	86.45	0.690	0.510	0.0566	0.0361
PCA-guided representatives	82.03	86.11	0.694	0.509	0.0562	0.0359
Fixed-anchor Hopfield	80.72	71.40	0.577	0.603	0.0567	0.0359
Optimized256	**84.35**	**88.00**	**0.766**	**0.438**	**0.0560**	**0.0361**
Full context	85.51	88.78	0.801	0.399	0.0623	0.0402

Quality at fixed serving capacity. Optimized256 improves over Unoptimized256 by 3.80/3.35 AUC points and 0.081/0.079 in R^2 (TabDPT/TabICLv2). Clustering partly closes the gap, but optimization remains ahead by 2.92 $[2.02, 3.86]$/2.25 $[1.26, 3.35]$ AUC points and 1.84/1.54 accuracy points. PCA-guided selection has similar mean AUC to clustering. On regression, optimization exceeds the best mean-R^2 alternative constructor by 0.079 $[0.045, 0.117]$ (TabDPT, clustering) and 0.072 $[0.037, 0.112]$ (TabICLv2, PCA-guided selection). Intervals are paired 95% dataset-bootstrap intervals. Figure 6 gives the paired construction effects in the appendix.

Pooling and serving costs. Fixed-anchor Hopfield pooling yields mean accuracy of 70.91%/71.40% and R^2 of 0.502/0.577; learned queries and projections were not evaluated. All compact arms have identical KV tensor sizes and similar measured inference times. Full context retains higher mean quality. Table 2 uses its own matched six-arm timing run and all 48 tasks; Table 1 uses the common K1024-eligible cohort and joint four-budget timings. These results assess the specified construction recipes, without tuning each method or refining every initializer.

5.4 Teacher guidance across memory capacities

We vary teacher weight at each fixed budget $K \in \{256, 512, 1024\}$, using $\lambda \in \{0, 0.1, 0.3, 1, 3, 10\}$. Every curve uses the same 30 classification and nine regression tasks and seeds 41–43. Within each K, initialization, fixed labels, the 100-update schedule, LR 0.03 and sampled loss-query sequence are paired. $\lambda = 0$ is supervised context optimization through the same frozen backbone, retaining the labels, initialization and update budget. This paired control measures the added value of guidance within our recipe; it is not a reproduction of TuneTables or ICD. Each $\lambda = 1$ arm reuses its verified primary memory. No new teacher is fitted.

At $\lambda = 1$, TabDPT's accuracy changes by +0.90/+1.09/+1.41 points and nRMSE decreases by +0.012/+0.013/+0.016 at K256/512/1024, respectively.

At $\lambda = 1$, TabICLv2's accuracy changes by +1.06/+1.21/+1.32 points and nRMSE decreases by +0.015/+0.017/+0.017 at K256/512/1024, respectively.

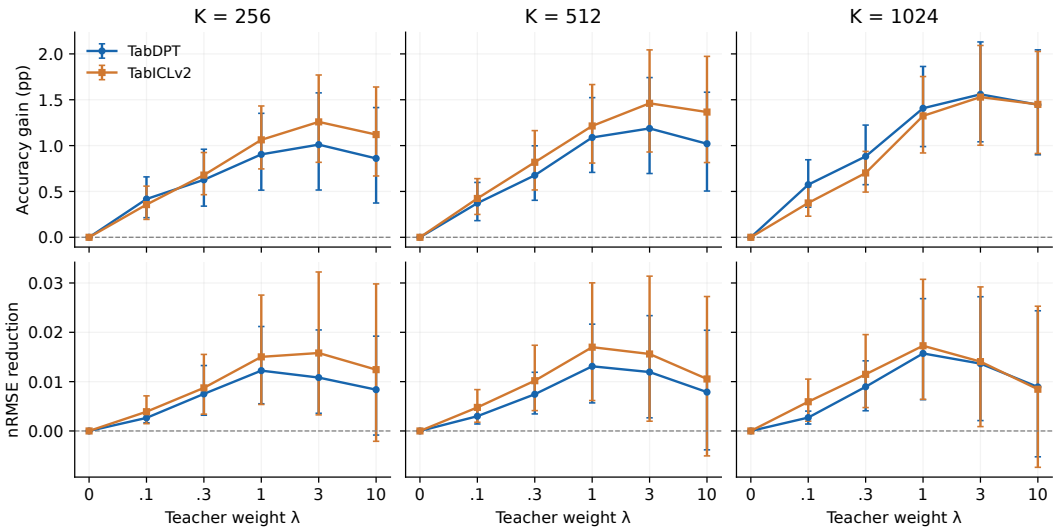


Figure 4: **Teacher guidance at K256/512/1024.** Paired changes from supervised-only slot optimization ($\lambda = 0$) in classification accuracy (top) and training-normalized RMSE (bottom). Every column uses the same tasks and three seeds. Positive values favor guidance; bars are 95% dataset-bootstrap intervals. Vertical scales match across K within each row. nRMSE reduction is absolute. Weight values occupy equally spaced positions. The default remains $\lambda = 1$.

The sweep assesses sensitivity rather than selecting a weight from test scores. CE+λKL and label-MSE+λteacher-MSE are not divided by $1 + \lambda$; changing λ changes both teacher weighting and gradient scale under Adam/clipping. The sweep therefore cannot isolate teacher information from scale effects. AUC, R^2 and paired intervals are in Appendix B.5.

5.5 Comparison with broader tabular baselines

We compare optimized memories at K64/256/512/1024 and full-context TFMs with XGBoost, CatBoost, LightGBM, Random Forest, RealMLP (Holzmüller et al., 2024) and TabM (Gorishniy et al., 2025). Classification also includes adapted single-prompt TuneTables components at 256/1,000 slots on frozen TabPFN-v1 (Feuer et al., 2024), without the full search/ensemble pipeline. Their different backbone and adaptation prevent attributing rank differences to the context-learning recipe alone. All baselines use fixed presets, without a new hyperparameter search. Figure 5 uses the same 30 classification and nine regression tasks as Table 1: every method is restricted to the K1024-eligible cohort and reranked. All test rows and seeds 41–43 are retained. We rank seed-mean accuracy or raw-unit RMSE within each dataset, then average dataset ranks. Appendix B.8 retains the original 35/13-task K256 comparison and baseline protocols.

Regression improves with memory capacity. Across K64/256/512/1024, TabICLv2's mean RMSE ranks are 12.33/9.78/6.89/5.44; TabDPT's are 14.11/12.22/10.78/8.89. Full-context TabICLv2/TabDPT remain ahead at 1.56/1.89. Optimized1024 with TabICLv2 ranks third among the 16 regression methods. Classification ranks are not monotonic in K: the four accuracy ranks are 8.48/8.48/9.22/9.72 for TabICLv2 and 10.23/9.65/9.62/9.77 for TabDPT. Mean accuracy ranges only from 88.80% to 88.87% for TabICLv2 and from 88.42% to 88.66% for TabDPT. Ranks discard the magnitude of within-task gaps. The results strengthen the regression quality–serving tradeoff in Table 1, without identifying a universal best K or a matched latency advantage over the newly fitted baselines.

6 Limitations

Scope and tuning. Evaluation uses one official fold, two backbones and one GPU; pretraining overlap remains possible. The primary schedule and constructor settings are fixed; the teacher-weight

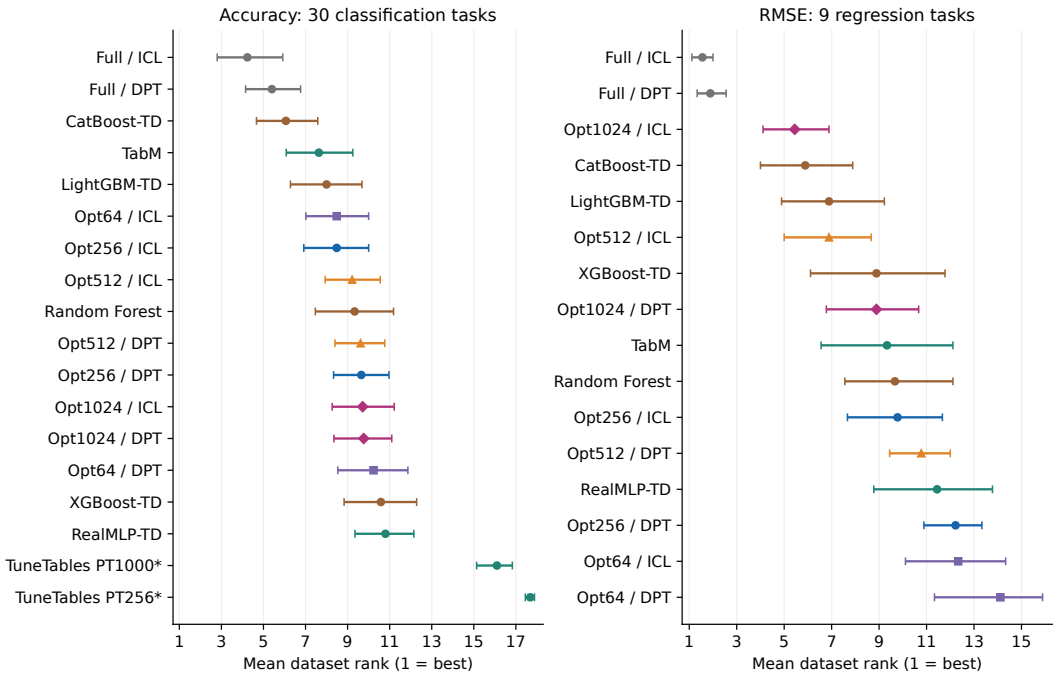


Figure 5: **Accuracy and RMSE ranks across four slot budgets.** All 18 classification and 16 regression methods share the same 30/9 tasks. Lower is better. Each task score averages three seeds; bars are 95% dataset-bootstrap intervals, not significance tests. Each optimized budget is labeled separately for both backbones. DPT/ICL denote TabDPT/TabICLv2. Asterisks mark adapted TuneTables single prompts on TabPFN-v1, without full search/ensembles; TD denotes tuned defaults.

sweep retains one as the default. The broader comparison uses preset baselines, without equal tuning budgets or reproduction of full TuneTables search/ensembles. The construction ablation tests one recipe per method at K256/K1024, without tuning them or refining each initializer. It does not establish superiority over learned condensers, all regularizers or fully tuned selection methods. Stronger validation-tuned CE512 controls on earlier capped TabDPT cohorts narrow the optimized256 advantage to 0.03/0.88 AUC points. Teacher/smoothing controls qualify the objective without identifying a unique mechanism.

Serving and transfer. Full context retains higher mean quality. Preparation uses training data and can outweigh query savings at small workloads. Warm timings exclude backbone loading and durable commits; TabICLv2 retains a column cache in addition to K slots. Broad size plots compare different datasets and do not establish causal within-task scaling. Earlier combined-memory and prospective coverage experiments did not support transfer of the single-context gain; their separate protocols and negative results remain in the archived ledgers. Compact state establishes neither privacy nor row-level unlearning.

7 Conclusion

Context optimization improves accuracy over the same unoptimized 256-slot initialization with essentially unchanged compact serving state and cost. The all-task K256 construction comparison and teacher-weight ablations test how the memory values are learned at fixed capacity. Complete-dataset comparisons expose a remaining mean quality gap to full context and a preparation cost. Under the tested schedule, K256 is a reasonable classification starting point; K1024 offers additional regression quality at higher state and inference cost. These findings guide validation under a serving budget, without establishing an optimal K. Compiled task memory therefore provides an empirical accuracy–cost tradeoff whose preparation, storage and query costs must be evaluated together.

AI USE STATEMENT

Generative AI tools assisted literature and manuscript critique, experimental-design suggestions and figure generation. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REFERENCES

David Bonet, Daniel Mas Montserrat, Xavier Giró-i Nieto, and Alexander G. Ioannidis. HyperFast: Instant classification for tabular data. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 11114–11123, 2024. doi: 10.1609/aaai.v38i10.28988.

George Cazenavette, Tongzhou Wang, Antonio Torralba, Alexei A. Efros, and Jun-Yan Zhu. Dataset distillation by matching training trajectories. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

Tianqi Chen and Carlos Guestrin. XGBoost: A scalable tree boosting system. In *ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*, 2016.

Nick Erickson, Lennart Purucker, Andrej Tschalzev, David Holzmüller, Prateek Mutalik Desai, David Salinas, and Frank Hutter. TabArena: A living benchmark for machine learning on tabular data. In *Advances in Neural Information Processing Systems (NeurIPS), Datasets and Benchmarks Track*, volume 38, 2025. arXiv:2506.16791.

Benjamin Feuer, Robin Tibor Schirrmeister, Valeriia Cherepanova, Chinmay Hegde, Frank Hutter, Micah Goldblum, Niv Cohen, and Colin White. TuneTables: Context optimization for scalable prior-data fitted networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

Yury Gorishniy, Akim Kotelnikov, and Artem Babenko. Tabm: Advancing tabular deep learning with parameter-efficient ensembling. In *International Conference on Learning Representations (ICLR)*, 2025. arXiv:2410.24210.

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

David Holzmüller, Léo Grinsztajn, and Ingo Steinwart. Better by default: Strong pre-tuned MLPs and boosted trees on tabular data. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024. arXiv:2407.04491.

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 *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

Junwei Ma, Valentin Thomas, Guangwei Yu, and Anthony Caterini. In-context data distillation with tabPFN. *arXiv preprint arXiv:2402.06971*, 2024.

Junwei Ma, Valentin Thomas, Rasa Hosseinzadeh, Alex Labach, Hamidreza Kamkari, Jesse C. Cresswell, Keyvan Golestan, Guangwei Yu, Anthony L. Caterini, and Maksims Volkovs. TabDPT: Scaling tabular foundation models on real data. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 38, 2025. arXiv:2410.18164.

Andreas Müller, Carlo Curino, and Raghu Ramakrishnan. MotherNet: Fast training and inference via hyper-network transformers. In *International Conference on Learning Representations (ICLR)*, 2025. arXiv:2312.08598.

Liudmila Prokhorenkova, Gleb Gusev, Aleksandr Vorobev, Anna Veronika Dorogush, and Andrey Gulin. CatBoost: Unbiased boosting with categorical features. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

Jingang Qu, David Holzmüller, Gaël Varoquaux, and Marine Le Morvan. TabICLv2: A better, faster, scalable, and open tabular foundation model. In *International Conference on Machine Learning (ICML)*, 2026. arXiv:2602.11139.

Hubert Ramsauer, Bernhard Schäfl, Johannes Lehner, Philipp Seidl, Michael Widrich, Thomas Adler, Lukas Gruber, Markus Holzleitner, Milena Pavlović, Geir Kjetil Sandve, Victor Greiff, David Kreil, Michael Kopp, Günter Klambauer, Johannes Brandstetter, and Sepp Hochreiter. Hopfield networks is all you need. *arXiv preprint arXiv:2008.02217*, 2021. URL https://arxiv.org/abs/2008.02217.

Aditya Tanna, Nassim Bouarour, Mohamed Bouadi, Vinay Kumar Sankarapu, and Pratinav Seth. Pocket foundation models: Distilling TFMs into CPU-ready gradient-boosted trees. *arXiv preprint arXiv:2605.18654*, 2026.

Valentin Thomas, Junwei Ma, Rasa Hosseinzadeh, Keyvan Golestan, Guangwei Yu, Maksims Volkovs, and Anthony Caterini. Retrieval & fine-tuning for in-context tabular models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

Tongzhou Wang, Jun-Yan Zhu, Antonio Torralba, and Alexei A. Efros. Dataset distillation. *arXiv preprint arXiv:1811.10959*, 2018.

Guri Zabërgja, Rafiq Kamel, Arlind Kadra, Christian M. M. Frey, and Josif Grabocka. End-to-end compression for tabular foundation models. In *International Conference on Machine Learning (ICML)*, 2026. arXiv:2602.05649.

A PROTOCOL AND MODEL INTERFACES

A.1 DATA, STATISTICS AND PROVENANCE

Official TabArena repeat0/fold0/sample0 assignments define the 35 classification and 13 regression tasks. Classification uses 2–10 classes and at least 600 training rows. Training/test sizes range from 666/334 to 100,000/50,000 for classification and 604/303 to 35,960/17,980 for regression. There are 345,389 distinct test rows across the 48 tasks. Every eligible training and test row is retained; predictions use actual final chunks without repeated queries or padding.

Categorical vocabularies, finite numeric medians and any PCA to the 100-feature TFM limit fit only permitted training rows. Unknown and missing categories receive distinct codes. Teachers refit preprocessing within their own context fold. The supplied `ctm.py` implements these transforms and teacher folds; broader baselines use the method-specific transforms in Appendix B.8.

Quality averages seeds 41–43 within dataset before equally weighting datasets. Paired effects bootstrap datasets with 10,000 percentile draws, unless the rank experiment's separate seed is specified. Three seeds do not multiply the number of independent datasets. Speed and RMSE ratios use geometric means; rank and absolute quality summaries use arithmetic means. Intervals are unadjusted across comparisons. Raw RMSE is used within tasks and is not averaged across incompatible units. Training-normalized RMSE uses each backbone's fitted training target scale.

Materials in the supplementary archive. `data/table2_scores.csv` contains all 48 task names, train/test sizes, seeds, construction-arm scores and matched timings for Table 2. `data/figure5_scores.csv` contains the per-seed scores for Figure 5 on its common 30/9-task cohort. `reproduce.py` verifies both input checksums and implements their aggregation, rank ties and the figure's 10,000-draw bootstrap. `ctm.py` pins the TabDPT/TabICLv2 checkpoint filenames, download revisions; `requirements.txt` and `requirements-model.txt` pin the corresponding dependencies. `README.md` maps these files to Appendices A.1 and A.2. This compact release provides sample method code and saved measurements for these two exhibits; the original split-index, prediction and full benchmark audit bundles are not included.

Dataset metadata. Tables 3 and 4 list the complete benchmark. Feature counts exclude the target and precede the TFM's PCA reduction. All 48 tasks support K64/256/512; the K1024 column identifies the 30 classification and nine regression tasks used in the common-cohort comparisons. Dataset identifiers match the supplementary score CSVs. Complete metadata, including OpenML IDs and target-column names, is provided in the supplementary `README.md`.

Table 3: Metadata for the 35 classification datasets. D: raw input features before PCA; C: classes. K1024: at least 1,024 training rows.

Dataset	Train	Test	D	C	K1024
Amazon_employee_access	21,846	10,923	9	2	Yes
APSFailure	50,666	25,334	170	2	Yes
bank-marketing	30,140	15,071	13	2	Yes
Bank_Customer_Churn	6,666	3,334	10	2	Yes
Bioresponse	2,500	1,251	1776	2	Yes
churn	3,333	1,667	19	2	Yes
coil2000_insurance_policies	6,548	3,274	85	2	Yes
credit-g	666	334	20	2	No
credit_card_clients_default	20,000	10,000	23	2	Yes
customer_satisfaction_in_airline	86,586	43,294	21	2	Yes
Diabetes130US	47,678	23,840	47	2	Yes
E-CommereShippingData	7,332	3,667	10	2	Yes
Fitness_Club	1,000	500	6	2	No
GiveMeSomeCredit	100,000	50,000	10	2	Yes
hazelnut-spread-contaminant-detection	1,600	800	30	2	Yes
heloc	6,972	3,487	23	2	Yes
hiva_agnostic	2,563	1,282	1617	3	Yes
HR_Analytics_Job_Change_of_Data_Scientis	12,772	6,386	12	2	Yes
in_vehicle_coupon_recommendation	8,456	4,228	24	2	Yes
Is-this-a-good-customer	1,148	575	13	2	Yes
jm1	7,256	3,629	21	2	Yes

Continued on next page

Table 3 (continued)

Dataset	Train	Test	D	C	K1024
kddcup09_appetency	33,333	16,667	212	2	Yes
Marketing_Campaign	1,493	747	25	2	Yes
maternal_health_risk	676	338	6	3	No
MIC	1,132	567	111	8	Yes
NATICUSdroid	4,994	2,497	86	2	Yes
online_shoppers_intention	8,220	4,110	17	2	Yes
polish_companies_bankruptcy	3,940	1,970	64	2	Yes
qsar-biodeg	702	352	41	2	No
SDSS17	52,035	26,018	11	3	Yes
seismic-bumps	1,722	862	15	2	Yes
splice	2,126	1,064	60	3	Yes
students_dropout_and_academic_success	2,949	1,475	36	3	Yes
taiwanese_bankruptcy_prediction	4,546	2,273	94	2	Yes
website_phishing	902	451	9	3	No

Table 4: Metadata for the 13 regression datasets. D: raw input features before PCA. K1024: at least 1,024 training rows.

Dataset	Train	Test	D	K1024
airfoil_self_noise	1,002	501	5	No
Another-Dataset-on-used-Fiat-500	1,025	513	7	Yes
concrete_compressive_strength	686	344	8	No
diamonds	35,960	17,980	9	Yes
Food_Delivery_Time	30,300	15,151	9	Yes
healthcare_insurance_expenses	892	446	6	No
houses	13,760	6,880	8	Yes
miami_housing	9,184	4,592	15	Yes
physiochemical_protein	30,486	15,244	9	Yes
QSAR-TID-11	3,828	1,914	1024	Yes
QSAR_fish_toxicity	604	303	6	No
superconductivity	14,175	7,088	81	Yes
wine_quality	4,331	2,166	12	Yes

A.2 Frozen representations and serving

TabDPT stores feature vectors and supplies fixed slot labels separately to its label encoders. Classification uses native logits; regression uses the native bin expectation, transformed back to target units. TabICLv2 stores label-conditioned row representations and retains the fitted column cache; it uses one estimator and a seeded feature permutation. Regression predictions use the native quantile mean. Both backbones update only FP32 slot parameters. TabDPT executes in FP32; TabICLv2 uses native FP16 with gradient scaling during optimization.

Full and compact memories use KV caching at the same model interface. Query chunks are at most 1,024 rows, while attention keeps every retained context key/value. Required preprocessing, labels, compact vectors and architecture-specific caches enter task-state accounting; shared backbone weights do not. Construction row banks and teacher targets are excluded from compact serving artifacts. Full-context speedups in the abstract, main results and Table 5 use the joint four-budget timing run; the supplementary Table 2 CSV instead contains the separate matched construction timings. Preparation accounts for encoding, teachers where used, fitting/construction and final caches, separately from warm query serving. Reused component times are reconstructed costs, not fresh end-to-end preparation timings.

Executable interface. In the supplementary `ctm.py`, `compile` fits and saves memory from a labeled training CSV; `predict` reloads it and predicts every supplied query row. `memory_payload` defines the persisted task state, and `MemoryPredictor` builds the serving cache. `README.md` provides installation, input formats and commands for both backbones and task types. Checkpoints download separately; a compiled memory can be reloaded without its training CSV.

Table 5: Complete training and test splits, $K = 256$, three seeds. Quality is averaged within each dataset and then equally across datasets. Classification cells give AUC / accuracy (%); regression cells give R^2 / training-normalized RMSE. Speedup is the geometric mean of paired full/optimized complete-test times from the joint four-budget timing run, matching the main text. All arms use KV caches; these all-task means differ from the 30/9-task cohort in Table 1.

Tasks	Backbone	Unoptimized256	Optimized256	Full context	Speedup
35 classification	TabDPT	79.95 / 85.60	83.75 / 87.74	85.32 / 88.50	3.30
35 classification	TabICLv2	81.00 / 85.45	84.35 / 88.00	85.51 / 88.78	1.17
13 regression	TabDPT	0.674 / 0.534	0.755 / 0.449	0.801 / 0.401	3.20
13 regression	TabICLv2	0.687 / 0.517	0.766 / 0.438	0.801 / 0.399	1.19

The original benchmark's cached/native probes used training rows only. TabDPT adopted FP32 after reduced-precision probes exceeded a 0.005 probability-error screen. TabICLv2 retains native FP16; some newly constructed memories exceed that screen. Separate direct/cache checks used identical representations promoted to FP32 at a 10^{-4} tolerance. The supplied runtime was checked against saved optimized memories on one classification and one regression task for both backbones, with exact complete-test prediction agreement, followed by CSV compilation/save/reload checks. These focused checks are described in `README.md`; they do not establish bitwise repeatability of GPU optimization trajectories.

B Additional complete-split results

B.1 K256 on all 48 tasks

Table 5 retains the uncapped 35-classification/13-regression comparison. Unoptimized256 samples distinct rows uniformly within class or target-quantile strata and stores their full-pool frozen encodings. Optimized256 starts from exactly those vectors and labels. Uniform sampling over the whole dataset is a different rule and is not a separately measured arm. All 288 dataset–backbone–seed conditions and 864 quality cells pass independent scoring and exact saved-memory replay. Table 5 uses the joint four-budget timing run below, matching the main-text speedups; its K256/full-context predictions reproduce the original study exactly. The original three-arm timing run remains in its source ledger and is not pooled with these timings.

B.2 Paired slot-count effects

K1024 needs 1,024 distinct training rows, leaving 30 classification and nine regression tasks. All four budgets use that same cohort in Table 1; smaller-budget all-task records remain in the ledger. Initial rows are nested and stratified, with unchanged K256 anchors. Each new K uses the same 100-step schedule, loss-query sequence and verified OOF targets. Existing K256 memories are reused exactly; all eligible arms are timed together afresh.

At K1024, geometric mean RMSE is 3.6% lower than at K256 for both backbones on the common regression cohort. AUC and accuracy intervals for K512/K1024 versus K256 include zero. K1024 increases retained state by 3.40 times for TabDPT and 2.38/2.73 times for TabICLv2 (classification/regression). Whole-test latency increases by 1.24/1.26 and 1.01/1.02 times, respectively. This supports a stronger regression tradeoff, without establishing universal optimality or convergence at 100 updates. All 288 conditions, 2,484 serving arms and 24,840 timing calls pass the complete-grid audit. Per-task scores, eligibility, preparation and state records remain under `results/review_revision/slot_count_full_context`.

B.3 Construction settings and numerical scope

All constructors preserve the exact anchor stratum quotas and use complete frozen training encodings. Cluster representatives standardize encodings and choose distinct real rows nearest mini-batch k-means centers within each stratum: three initializations, at most 100 iterations, batch size 2,048 and ten non-improving steps. PCA-guided selection first uses randomized PCA of rank

Table 6: Paired changes from optimized256 on the fixed four-budget cohort (30 classification and nine regression tasks). Entries are estimates [95% dataset-bootstrap intervals], averaging three seeds per dataset. Positive differences favor the new K; RMSE ratios below one favor the new K. Intervals are unadjusted across comparisons.

Backbone	K	ΔAUC (percentage points)	ΔAccuracy (percentage points)
TabDPT	64	-0.32 $[-0.58, -0.06]$	-0.22 $[-0.45, +0.01]$
TabDPT	512	$+0.12$ $[-0.05, +0.28]$	$+0.03$ $[-0.07, +0.12]$
TabDPT	1024	$+0.12$ $[-0.13, +0.40]$	-0.02 $[-0.22, +0.16]$
TabICLv2	64	-0.13 $[-0.62, +0.50]$	-0.07 $[-0.33, +0.14]$
TabICLv2	512	$+0.27$ $[-0.01, +0.63]$	-0.04 $[-0.16, +0.07]$
TabICLv2	1024	$+0.32$ $[-0.04, +0.74]$	-0.02 $[-0.21, +0.19]$
Backbone	K	ΔR^2	RMSE ratio ($K/256$)
TabDPT	64	-0.031 $[-0.045, -0.017]$	1.109 $[1.038, 1.224]$
TabDPT	512	$+0.009$ $[+0.003, +0.015]$	0.980 $[0.969, 0.993]$
TabDPT	1024	$+0.015$ $[+0.005, +0.027]$	0.964 $[0.948, 0.983]$
TabICLv2	64	-0.024 $[-0.043, -0.009]$	1.058 $[1.028, 1.088]$
TabICLv2	512	$+0.010$ $[+0.005, +0.016]$	0.975 $[0.966, 0.984]$
TabICLv2	1024	$+0.016$ $[+0.007, +0.028]$	0.964 $[0.948, 0.977]$

$\min(32, D, N-1)$ with four power iterations, then the same selection. Both export the original full-width vectors and labels.

Fixed-anchor Hopfield pooling uses stratified vectors as queries, normalized keys/queries, inverse temperature 10, one softmax lookup within each stratum, identity projections and no residual. It reads the training bank in 2,048-row chunks. Classification keeps anchor labels; regression pools labels with the same weights. No constructor receives teacher targets or task-loss updates. This is a fixed-query adaptation, not trained Hopfield pooling or a learned condenser.

The K256 study covers 288 conditions and 864 new memories, reusing the exact unoptimized, optimized and full-context controls. All 1,728 arm predictions replay exactly. Six arms are timed together with three warmups and ten measured complete-test passes. Native-FP16 discrepancies are retained alongside the separate FP32 structural cache check described above. The recipes were not retuned on test scores. Complete construction costs, state sizes and paired intervals are in `results/review_revision/memory_construction_k256`.

B.4 K1024 CONSTRUCTION AND TEACHER-WEIGHT PROTOCOL

The K1024 follow-up uses the same 30 classification and nine regression tasks as Table 1, two backbones and seeds 41–43: 234 conditions. The three constructors retain the K256 settings above and the exact K1024 anchors' stratum quotas, producing 702 new memories. Unoptimized1024, Optimized1024 and full context reuse their verified vectors and predictions. Six construction-table arms are timed together afresh on complete test splits. Appendix B.6 reports these construction results. Comparison with the main-text K256 table is descriptive because Table 2 includes all 48 tasks and uses a separate timing run.

The teacher-weight sweep fixes K1024 and evaluates $\lambda \in \{0, 0.1, 0.3, 1, 3, 10\}$. Each non-unit weight receives 100 updates from the same initial vectors, with identical fixed labels, loss-query indices, initial LR 0.03 and cosine schedule. This produces 1,170 new optimized memories. The unit-weight arm reuses the exact earlier memory, rather than retraining it, and teachers are not refitted. Classification uses CE plus λ times KL; regression uses standardized-label MSE plus λ times teacher MSE. The sum is not divided by $1 + \lambda$: changing λ changes both the target mixture and gradient scale under Adam/clipping. Thus this sweep does not isolate teacher information from loss-scale effects. $\lambda = 0$ uses no teacher loss. No backbone weights change, and no weight is selected from test scores.

Independent scoring and exact saved-memory replays cover all 2,574 quality cells. The six constructor/control arms contribute 14,040 measured complete-test calls. Training-only smoke checks

cover loss equivalence at $\lambda = 1$ and cache parity. Nondeterministic GPU backward reductions mean optimization trajectories need not be bitwise repeatable, while inference from each saved memory is replayed exactly. Native-FP16/cache differences remain recorded with the separate FP32 structural check. Source hashes, exact diagnostics and all paired intervals are retained under `results/review_revision/k1024_ablation`.

B.5 Teacher-weight sensitivity across slot budgets

The extension applies the same six-weight grid and unchanged optimizer to K256 and K512 on the exact K1024-eligible cohort. It adds 468 dataset–backbone–seed–budget conditions and 2,340 optimized memories. All 2,808 new-study quality cells pass independent scoring and exact saved-memory prediction replay, including the reused unit-weight controls. Combined with the original K1024 sweep, the analysis contains 4,212 quality cells across 702 conditions. The parent construction study's additional unoptimized/full-context and constructor arms are outside this lambda-only count.

Initial source rows and labels are exactly those from the slot-count study. Each weight sees the same sampled query sequence within budget; OOF targets and preprocessing are reused by hash. No test score selects a weight or changes the schedule. Smaller budgets also use the common 30/9-task cohort so capacity effects cannot arise from differing dataset eligibility. No constructor or timing experiment is repeated for this extension. Native-FP16 diagnostic differences are retained separately from the FP32 structural cache check. The extended experiment waited for the parent's matched timing and replay audit before taking the GPU.

Table 7: Default guidance ($\lambda = 1$) minus zero guidance, at each fixed K. Same 30 classification/nine regression tasks, three seeds; estimates [95% paired dataset-bootstrap intervals]. Accuracy and AUC use percentage points. Positive nRMSE reductions favor guidance. No weight is selected using these test scores.

Backbone	K	ΔAUC (pp)	ΔACC (pp)
TabDPT	256	$+1.58$ $[+0.98, +2.26]$	$+0.90$ $[+0.51, +1.35]$
TabDPT	512	$+1.97$ $[+1.22, +2.80]$	$+1.09$ $[+0.71, +1.52]$
TabDPT	1024	$+2.58$ $[+1.66, +3.66]$	$+1.41$ $[+0.99, +1.86]$
TabICLv2	256	$+1.52$ $[+0.75, +2.36]$	$+1.06$ $[+0.75, +1.43]$
TabICLv2	512	$+2.03$ $[+1.13, +3.03]$	$+1.21$ $[+0.81, +1.67]$
TabICLv2	1024	$+2.49$ $[+1.59, +3.47]$	$+1.32$ $[+0.92, +1.75]$
Backbone	**K**	**ΔR^2**	**nRMSE reduction**
TabDPT	256	$+0.013$ $[+0.005, +0.021]$	$+0.012$ $[+0.006, +0.021]$
TabDPT	512	$+0.014$ $[+0.005, +0.026]$	$+0.013$ $[+0.006, +0.022]$
TabDPT	1024	$+0.017$ $[+0.005, +0.031]$	$+0.016$ $[+0.006, +0.027]$
TabICLv2	256	$+0.017$ $[+0.006, +0.030]$	$+0.015$ $[+0.005, +0.028]$
TabICLv2	512	$+0.020$ $[+0.006, +0.037]$	$+0.017$ $[+0.006, +0.030]$
TabICLv2	1024	$+0.020$ $[+0.006, +0.037]$	$+0.017$ $[+0.006, +0.031]$

The main figure shows accuracy and nRMSE; the analogous AUC/R^2 plot and absolute scores for every weight remain under `results/review_revision/lambda_slot_budgets/analysis`. Paired intervals average seeds within task and resample tasks, using 10,000 draws and seed 2027. They are unadjusted across the grid. Default-weight comparisons in Table 7 do not establish that one is the optimal teacher coefficient.

B.6 Memory construction at K1024 on the eligible cohort

Table 8 compares the exact stratified frozen-encoding initialization, cluster representatives, PCA-guided representatives and fixed-anchor Hopfield pooling with Optimized1024 and full context. All arms share the same 30 classification and nine regression tasks and three seeds. The three constructors use complete training encodings with the same stratum quotas, without task-loss updates or teacher targets. Cluster/PCA methods export distinct real rows; Hopfield pooling uses one fixed-query lookup. Settings and numerical checks are in Appendix B.3. The main-text K256 ablation

(Table 2) includes all 35 classification and 13 regression tasks, so differences between the two construction tables do not isolate the effect of K. The paired capacity analysis in Section 5.2 holds the cohort fixed. **Quality at fixed serving capacity.**

Table 8: **Memory-construction ablation at $K = 1024$.** The same 30 classification and nine regression tasks, three seeds. Unoptimized1024 is the stratified frozen-encoding initialization. Compact arms retain 1,024 slots; full context retains all training rows. Scores and times average seeds within dataset, then datasets equally. nRMSE uses the training target scale. Time is ms per actual test row, measured together afresh for all six arms; C/R denote classification/regression. Table 2 reports K256 on all 35/13 tasks; differences between these tables also reflect cohort and timing-run differences.

Construction	AUC (%)	ACC (%)	R^2	nRMSE	C ms/row	R ms/row
TabDPT						
Unoptimized1024	81.81	88.13	0.692	0.512	0.0368	0.0271
Cluster representatives	82.15	88.21	0.695	0.509	0.0368	0.0270
PCA-guided representatives	82.03	88.23	0.694	0.510	0.0368	0.0269
Fixed-anchor Hopfield	76.78	63.43	0.338	0.742	0.0368	0.0269
Optimized1024	**83.22**	**88.61**	**0.731**	**0.470**	**0.0368**	**0.0269**
Full context	84.67	89.24	0.777	0.428	0.1474	0.1232
TabICLv2						
Unoptimized1024	82.13	88.44	0.697	0.505	0.0610	0.0411
Cluster representatives	82.69	88.56	0.703	0.497	0.0609	0.0409
PCA-guided representatives	82.85	88.58	0.701	0.499	0.0609	0.0409
Fixed-anchor Hopfield	79.64	65.03	0.468	0.657	0.0604	0.0409
Optimized1024	**84.07**	**88.85**	**0.746**	**0.456**	**0.0604**	**0.0409**
Full context	84.84	89.45	0.775	0.428	0.0670	0.0464

For TabDPT, optimization changes AUC by $+1.40$ $[+0.65, +2.22]$ points, accuracy by +0.48 points and R^2 by $+0.039$ $[+0.017, +0.063]$ relative to Unoptimized1024. Optimized mean AUC/R^2 are 83.22%/0.731; the highest means among the three alternative constructors are 82.15%/0.695, respectively.

For TabICLv2, optimization changes AUC by $+1.94$ $[+0.87, +3.24]$ points, accuracy by +0.40 points and R^2 by $+0.049$ $[+0.022, +0.080]$ relative to Unoptimized1024. Optimized mean AUC/R^2 are 84.07%/0.746; the highest means among the three alternative constructors are 82.85%/0.703, respectively.

Intervals are paired 95% dataset-bootstrap intervals. All compact arms have the same KV tensor sizes; constructor choice changes stored values and preparation, while K fixes serving capacity. Fresh matched inference times appear in the table. These comparisons assess the specified construction recipes, without tuning each method or refining every initializer.

B.7 Paired construction effects at K256

Figure 6 supplements the all-task K256 construction results in Section 5.3 and Table 2. All 35 classification and 13 regression tasks enter the paired comparisons; the stratified encoded baseline is the exact initialization of Optimized256.

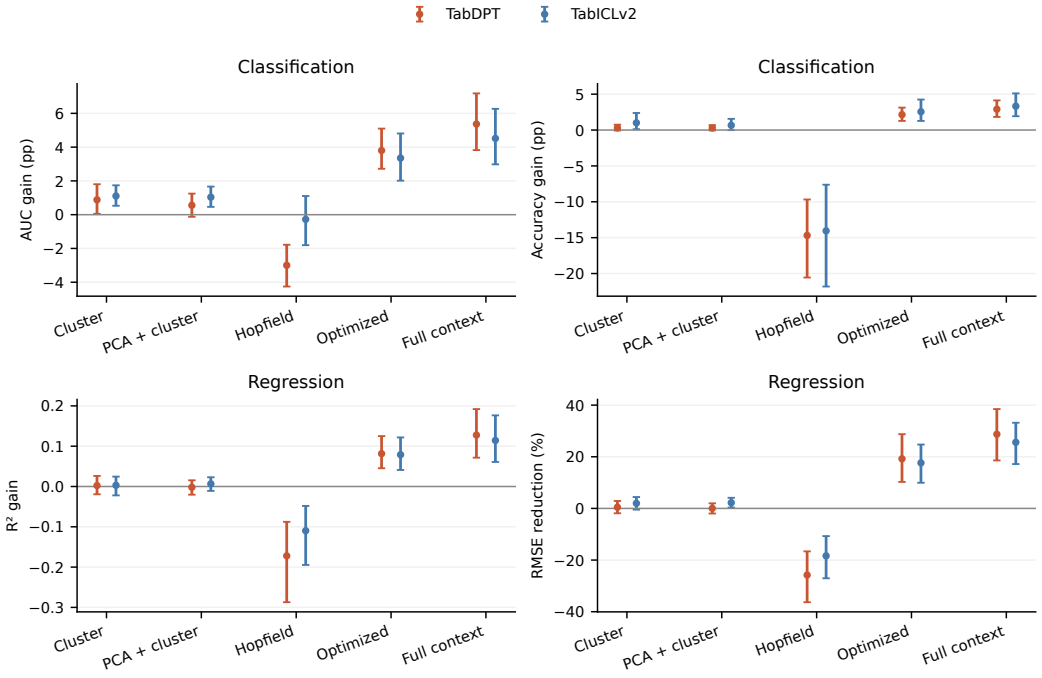


Figure 6: Memory construction at K256: quality relative to Unoptimized256 (stratified frozen encodings), on all 35 classification and 13 regression tasks. Points average paired seed differences within dataset and then datasets; intervals bootstrap datasets. RMSE reductions use geometric paired ratios. Positive values favor the plotted method. The full-context arm uses every training row.

B.8 BROADER TABULAR BASELINES ON THE COMPLETE SPLITS

Coverage and estimands. This experiment uses the same 35 classification and 13 regression tasks, official fold-0 splits and seeds 41, 42 and 43 as the complete K256 study. All original test rows are scored. There are 12 classification methods and 10 regression methods: 1,260 and 390 score cells, respectively. We reuse 576 independently rescored Optimized256/full-context cells and fit 1,074 new method–dataset–seed conditions. No incomplete or failed condition is omitted from the final panels. Optimized256 retains the paper's fixed 100-step, learning-rate-0.03 recipe; the broader baselines have their own fixed presets, not equal training budgets.

Accuracy is averaged across seeds before ranking methods in descending order within each classification task; RMSE is averaged across seeds before ranking in ascending order within each regression task. Exact ties receive average ranks. Mean ranks weight datasets equally. For this figure, 10,000 paired dataset-bootstrap draws use random seed 20260917; classification draws precede regression draws from the same generator. The three fits per task are not treated as independent datasets. Intervals describe mean-rank uncertainty and are not adjusted significance tests. Mean accuracy can order methods differently because ranks discard the magnitude of within-task gaps. We retain raw-unit RMSE per task, without averaging quantities with different units into one aggregate RMSE.

Method	Accuracy (%)	Classification rank	Regression rank
Full context (TabICLv2)	88.78	2.99 [2.20, 3.90]	1.54 [1.23, 1.92]
Full context (TabDPT)	88.50	4.07 [3.26, 4.90]	1.77 [1.46, 2.08]
CatBoost-TD	88.17	4.49 [3.70, 5.29]	5.23 [4.00, 6.54]
TabM	86.87	5.53 [4.67, 6.40]	6.69 [5.46, 7.92]
LightGBM-TD	87.82	5.70 [4.76, 6.63]	5.77 [4.77, 6.85]
Optimized256 (TabICLv2)	88.00	5.83 [4.87, 6.80]	5.85 [4.31, 7.38]
Random Forest	87.83	6.21 [5.21, 7.19]	7.62 [6.62, 8.62]
Optimized256 (TabDPT)	87.74	6.54 [5.74, 7.36]	6.85 [5.46, 8.23]
XGBoost-TD	86.51	6.73 [5.80, 7.64]	6.54 [5.46, 7.62]
RealMLP-TD	87.65	7.44 [6.74, 8.10]	7.15 [5.92, 8.31]
TuneTables PT1000*	80.18	10.66 [10.23, 11.00]	—
TuneTables PT256*	74.94	11.81 [11.66, 11.94]	—

Table 9: Complete three-seed baseline comparison. Accuracy averages datasets after seeds; ranks first compare methods within each dataset. Brackets are 95% dataset-bootstrap intervals on mean ranks. Regression ranks use raw-unit RMSE within task; raw RMSE is not pooled across tasks. TuneTables components (*) support classification only.

Trees and preprocessing. XGBoost (Chen & Guestrin, 2016), CatBoost (Prokhorenkova et al., 2018) and LightGBM (Ke et al., 2017) use the classification/regression tuned-default presets from PyTabKit (Holzmüller et al., 2024), implemented with their native libraries. Each has a maximum of 1,000 trees and patience 300. An 80/20 inner split, stratified for classification, selects the number of trees using accuracy or RMSE, followed by a refit on all outer training rows. Numeric medians and categorical vocabularies fit the inner training data during selection and the complete training data for refitting. XGBoost and LightGBM use native categorical columns, and CatBoost uses native categorical strings; unknown values have a distinct code. The 500-tree Random Forest uses ordinal categories, all outer training rows and the library defaults apart from maximum features ($\sqrt{D}$ for classification, D for regression). These models receive all raw columns, without the TFM's 100-feature cap. Settings are fixed before test evaluation, with no new hyperparameter search.

Neural baselines. RealMLP-TD uses PyTabKit's published defaults, one inner split and one full-data refit. Its external median/category transformer fits the inner training split and remains fixed; the library refits learned transforms and network parameters on all outer training rows. It runs in IEEE FP32, matching its disabling of TF32. TabM uses the official default architecture without numerical feature embeddings, with 32 implicit ensemble members and one-hot categorical features. Its numeric inputs use a training-fitted normal-quantile transformer with the PyTabKit TabR preprocessing defaults. AdamW uses learning rate 0.002 and weight decay 0.0003; BF16 autocast retains FP32 parameters. Validation selects an epoch within 1,000 epochs, with patience 16, then preprocessing and the model refit on the complete training set for the selected epoch count. The refit seed is the run seed plus 10,000. Batch size is 32/64/128/256/512/1,024 for training sizes below

2,800/4,500/6,400/32,000/108,000/otherwise. Classification predictions average member probabilities; regression predictions average member outputs and invert training-target scaling. These are preset implementations, not the methods' full published search protocols.

Table 10: Relationship to prior context optimization and scope of the evaluated controls. The $\lambda = 0$ control shares our backbone and training recipe; it does not reproduce a published method's complete pipeline.

Approach	Context / predictor	Supervision	Relationship to this study
TuneTables (Feuer et al., 2024)	Embedding prompts through TabPFN; frozen-predictor prompt tuning	Supervised loss; PFN-output KL explored	Prior prompt-only and prompt-plus-row serving. Ranked entries are adapted TabPFN-v1 single prompts, excluding full search/ensembles.
ICD (Ma et al., 2024)	Feature-space context through fixed TabPFN	Supervised context optimization	Antecedent for optimizing compact contexts; no separate ICD reproduction is ranked.
Supervised-only control ($\lambda = 0$)	Same K slots and frozen TabDPT/TabICLv2 as the guided arm	Label CE or MSE	Paired initialization, query sequence and 100 updates; tests guidance within the current recipe.
Guided memory ($\lambda = 1$)	Same K slots and frozen TabDPT/TabICLv2	Label loss plus OOF teacher matching	Measures quality at fixed serving capacity and tradeoffs against uncapped KV-cached full context.

TuneTables component scope. TuneTables PT256 and PT1000 use its released single-prompt training recipe through frozen TabPFN-v1, with 256 and 1,000 learned prefix tokens, default learning rate 0.1, at most 31 epochs and native training-loss stopping. Only the prefix is served; no original context rows accompany queries. This comparison does not reproduce TuneTables' full method-selection or ensemble pipeline. Its released prediction wrapper re-splits and reprocesses query inputs, so the adapter instead fits categories, medians, PCA up to 100 features, standardization, clipping to $[-10, 10]$ and $100/D$ scaling on the complete training pool, then calls the native forward on every test row. Native diagnostic validation overlaps training and is not used for selection; actual test inputs and labels never enter fitting. All 153 backbone state tensors remain exactly equal to the checkpoint. Training probes verify native-forward agreement and invariance to dummy query labels. The poor ranks of these components apply to this specified adaptation, not to the full published method. Backbone differences also prevent isolating a context-optimization advantage from these ranks; the same-backbone supervised-only control in Section 5.4 addresses the effect of adding guidance within our fixed recipe.

Repairs, numerical limitations and audit. Fifteen TabM fits (five entirely categorical datasets, all three seeds) initially failed because its API requires `None` when there are zero numerical features. The two forward call sites were corrected without altering learning settings. CatBoost on APS-Failure, seed 43, repeated an internal split-search assertion; disabling its exclusive-feature-bundling speed option resolved the failure. Its full refit then stopped natively at 332 of the selected 542 trees with a numerical-degeneracy warning; we retain the finite returned model and disclose this limitation instead of selecting a new regularization value from test performance. The original and repaired code are separately pinned, and all unaffected fits retain their artifacts. All 1,074 new exported models pass prediction replay (within 10^{-6} tolerance; classification decisions identical). The final audit independently rescores all 1,650 cells and checks source indices and prediction/model hashes. New fitting jobs can overlap, so their recorded runtimes are diagnostics, not a matched latency benchmark against classical or neural baselines. The TFM serving measurements remain those of Section 5.1.

Four-budget extension in the main figure. We independently rescore existing predictions after restricting every method to the same 30 classification and nine regression tasks eligible for K1024,

then include each backbone's verified Optimized64/256/512/1024 predictions. This gives 1,620 classification and 432 regression score cells, with no new fitting. Integer correct-prediction totals identify exact accuracy ties across three seeds; regression ranks use mean raw RMSE. The same 10,000-draw dataset bootstrap and seed apply. Adding methods changes the ranks of every method, so these 18/16-method ranks must not be compared numerically with the original 12/10-method, 35/13-task ranks in Table 9.