ACCELERATING TABULAR FOUNDATION MODELS VIA RESIDUAL-AWARE CONTEXT COMPRESSION

Anonymous authors
Paper under double-blind review

ABSTRACT

Tabular foundation models (TFMs) enable in-context learning over tabular data by conditioning on a labeled training table without task-specific backbone updates. However, their inference cost grows with context size because each query attends to the training rows, leading to substantial latency and memory overhead on large datasets. In this paper, we study context compression for TFM inference and identify three fundamental issues when reducing the context: (1) insufficient preservation of predictive signals for the frozen TFM, (2) the overhead of incorporating supervision from omitted rows, and (3) the lack of a unified compression framework across frozen TFMs and downstream tasks. To address these issues, we propose Residual-Aware Context Compression (RCC), which compiles a large training table into a compact reusable artifact through a one-time offline process. RCC first selects an initial set of context rows, then uses prediction residuals under this context to identify additional rows with complementary predictive information. Afterwards, it fits a closed-form residual readout using the remaining rows to correct predictions under the compact context. Together, RCC preserves key information from the full context and generalizes across different TFMs and downstream tasks. We evaluate RCC on three frozen TFMs and 100 classification and 29 regression datasets. At $128\times$ compression, RCC retains 96.5%–97.5% of full-context classification macro-F1, with regression nRMSE increasing by 8.3–8.6 points, while achieving median single-query speedups of 2.56–$15.61\times$. On 32 large-scale datasets with up to one million rows, our lightweight variant, RCC-lite, maintains comparable predictive performance at $128\times$ compression ratio, achieving single-query speedups of up to $1{,}514\times$ over the full-context baseline.

1 INTRODUCTION

Tabular foundation models (TFMs) (Hollmann et al., 2025) have shifted tabular learning from fitting dataset-specific predictors to solving new tasks through in-context learning (Hollmann et al., 2023). Pretrained on diverse tabular tasks, TFMs condition on a labeled training table to predict unseen queries without updating the backbone. Recent models, e.g., TabPFNv2 (Hollmann et al., 2025), LimiX-2M (Wang et al., 2026), and TabICLv2 (Qu et al., 2026), have demonstrated competitive performance across classification and regression. This paradigm reduces task-specific training but shifts computation to inference, because prediction uses attention between queries and labeled context rows. As tables grow, attention incurs substantial latency and memory overhead, hindering TFM deployment on large datasets and in high-throughput applications (Feuer et al., 2023).

To solve this problem, a natural remedy is to replace the full training table with a compact context. Existing approaches differ primarily in how this context is constructed and reused. Fixed subset-selection methods, including random sampling, clustering, and classical coresets (Feuer et al., 2023), are efficient and produce a single context that can be shared across queries. However, coverage of the data distribution alone does not ensure that the selected rows are useful for prediction. Simple selection criteria are largely data-driven and model-agnostic. By contrast, model-aware methods use validation feedback to search for a fixed context (Chen et al., 2026). To incorporate query-specific prediction, query-conditioned methods retrieve a different context for each query (Thomas et al., 2024). However, constructing a context for each query incurs additional retrieval and selection costs at inference. Per-table context optimization and dataset distillation instead construct contexts by encoding supervision in synthetic examples through iterative optimization (Feuer et al., 2024; Ma et al.,

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.

2024). Although reusable, these contexts require per-table optimization. Thus, query-conditioned selection adds inference cost, whereas per-table optimization increases context construction cost.

An ideal approach would construct a compact context at low cost and reuse it across queries while preserving predictive quality. However, achieving this goal presents three challenges. First, context selection should reflect the predictive power of rows to the frozen backbone. A valuable row is not only distributionally representative, but also sufficiently informative to improve prediction under the context. Directly evaluating candidate insertions requires repeated backbone evaluations. Second, supervision from omitted rows can be incorporated through learned contexts, but per-table gradient optimization adds repeated forward and backward passes. For example, TuneTables optimizes a learned context for each dataset through iterative prompt tuning (Feuer et al., 2024). The challenge is therefore to incorporate supervision from the omitted rows without iterative gradient optimization or backbone updates. Third, a unified context-compression framework that supports different frozen TFMs and both classification and regression is still lacking. Without such a framework, users may need separate compression methods for different backbones or task types, limiting their flexibility to choose models suited to their applications.

In light of these challenges, we propose Residual-Aware Context Compression (RCC). RCC operates in two stages. During *compilation*, given the training table and a frozen backbone, it selects a compact context of real rows and fits a residual readout once. During *inference*, each new query is predicted using the same context and readout, without revisiting the omitted rows. Specifically, residual-guided context selection (RGCS) first selects representative rows, then uses prediction residuals to select additional rows. A closed-form residual readout (CRR) processes the remaining rows as queries under the selected context and uses their representations to fit a residual correction. It is efficiently fitted in closed form without updating the backbone or performing iterative optimization. Together, the selected rows provide explicit in-context evidence, while a fixed readout reuses supervision from the omitted rows across queries. The same compression procedure supports different frozen TFM backbones and both classification and regression.

We evaluate RCC using three frozen backbones—TabPFNv2 (Hollmann et al., 2025), LimiX-2M (Wang et al., 2026), and TabICLv2 (Qu et al., 2026)—on 100 classification and 29 regression datasets. At $128\times$ compression, RCC retains 96.5%–97.5% of full-context classification macro-F1, with regression nRMSE increasing by 8.3–8.6 points, while achieving median single-query speedups of 2.56–$15.61\times$. On 32 additional datasets with up to one million training rows, our lightweight variant RCC-lite maintains comparable predictive performance to the full-context reference at $128\times$ compression, with single-query speedups of up to $1{,}514\times$ over the local full-context baseline.

Our main contributions are as follows:

- We formulate reusable context compression as a general approach to efficient TFM inference. To our knowledge, RCC is the first query-independent context compression framework supporting both classification and regression across multiple existing frozen TFMs.
- We propose RCC, an efficient method that preserves training information through two complementary components. Residual-guided context selection constructs a compact real-row context aligned with the predictive behavior of the TFM, while a closed-form residual readout captures supervision from omitted rows without updating the backbone. Moreover, we evaluate residual alignment as a selection signal and characterize when the residual readout can improve predictions under a fixed context.
- We conduct extensive experiments on three frozen TFMs and 129 classification and regression datasets, plus 32 large-scale datasets containing up to one million rows. RCC and RCC-lite achieve state-of-the-art efficiency–performance trade-offs, substantially reducing inference time and memory while preserving predictive performance.

2 Related Work

Tabular foundation models. TabPFN introduced tabular in-context learning through pretraining on synthetic priors (Hollmann et al., 2023). Subsequent models extend its architecture, scale, and task coverage, including TabPFNv2 (Hollmann et al., 2025), TabICL (Qu et al., 2025; 2026), and LimiX (Zhang et al., 2025), while TabDPT explores pretraining on real data (Ma et al., 2025). These models predict without task-specific backbone updates, but large contexts increase inference costs.

Context compression with fixed artifacts. Fixed-context methods construct a compact context once and reuse it across queries. Selection quality depends on both context size and row composition (Thomas et al., 2024; Abdullah, 2026). Common approaches include sampling, clustering, and coreset selection (Feuer et al., 2023); model-aware alternatives use validation-risk or Shapley-style row valuation (Rundel et al., 2024) and validation-guided search (Chen et al., 2026). For subtable-only predictors, unselected labels may affect selection, but predictions rely on the retained context.

Beyond row selection, TuneTables and in-context data distillation optimize embeddings or synthetic examples through per-table gradient descent (Feuer et al., 2024; Ma et al., 2024). TACO instead jointly meta-trains a compressor and a companion predictor to amortize compression across new tables (Zabërgja et al., 2026). RCC retains observed rows and incorporates supervision from the remaining rows through a closed-form residual readout, without per-table gradient optimization or changes to the existing backbone.

Query-conditioned and streaming access. TabPFN-kNN retrieves query-local contexts, while LoCalPFN additionally fine-tunes the model (Thomas et al., 2024). MixturePFN routes queries to a prompter pool (Xu et al., 2025), and CRUMB selects distribution-matched subtables for each test batch (Heredge et al., 2026). These methods construct contexts from a retained source pool, index, or prompter bank; RCC reuses a fixed context and readout across queries. CURE addresses streaming data with a bounded context (Lee et al., 2026), while GOTabPFN reduces feature-axis costs through reordering and compact tokenization (Habib et al., 2026).

3 Residual-Aware Context Compression

As shown in Figure 1, RCC combines residual-guided context selection (RGCS) with a closed-form residual readout (CRR). In the following sections, we first formulate the compression problem, then describe how RGCS constructs the context and how CRR uses supervision from the omitted rows.

3.1 Problem Formulation

We denote a pretrained tabular foundation model with frozen parameters θ by F_θ. The labeled training table is $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$. Each row contains a feature vector $\mathbf{x}_i \in \mathbb{R}^d$ and its label or regression target y_i. We write $[n] = \{1, \dots, n\}$ for the training indices. An index set $I \subseteq [n]$ specifies the context $\mathcal{D}_I = \{(\mathbf{x}_i, y_i) : i \in I\}$ supplied to the model. Conditioned on $\mathcal{D}_I$, the frozen backbone produces a query representation $\mathbf{e}_I(\mathbf{x})$ and maps it to a prediction through its native task-specific output mapping g_I. We denote the complete native predictor by

$$f_I(\mathbf{x}) = g_I\big(\mathbf{e}_I(\mathbf{x})\big). \tag{1}$$

For classification, $f_I(\mathbf{x})$ is a probability vector over C classes; for regression, it is a scalar response. Both the representation computation and g_I remain frozen. Our residual readout uses $\mathbf{e}_I(\mathbf{x})$ to correct the native prediction, while leaving g_I unchanged.

Given a training table $\mathcal{D}$ and a frozen backbone F_θ, let B denote the context budget. Our goal is to compile a reusable predictor without iterative gradient optimization. The predictor should use at most B real training rows as context, reduce inference cost, and preserve predictive quality relative to the full-context reference $f_{[n]}$. The compiled artifact is

$$\mathcal{A}_{\mathrm{RCC}} = \big(\mathcal{D}_{\widehat{I}}, \widehat{\mathbf{W}}\big), \qquad \widehat{I} \subseteq [n], \quad |\widehat{I}| \le B,$$

where $\mathcal{D}_{\widehat{I}}$ contains the retained labeled training rows and $\widehat{\mathbf{W}}$ the fitted residual-readout parameters.

3.2 Residual-Guided Context Selection

A direct way to measure a candidate row's contribution to a frozen TFM is to add it to a fixed context and evaluate its impact on prediction loss over held-out queries. However, this requires repeated backbone inference for every candidate, making it computationally expensive. To avoid this overhead, we use prediction residuals defined in Eq. (4) as a proxy for predictive utility. Residual alignment captures the potential of a candidate row to improve predictions for other rows with similar residual directions, providing an efficient criterion for row selection. Our single-row intervention

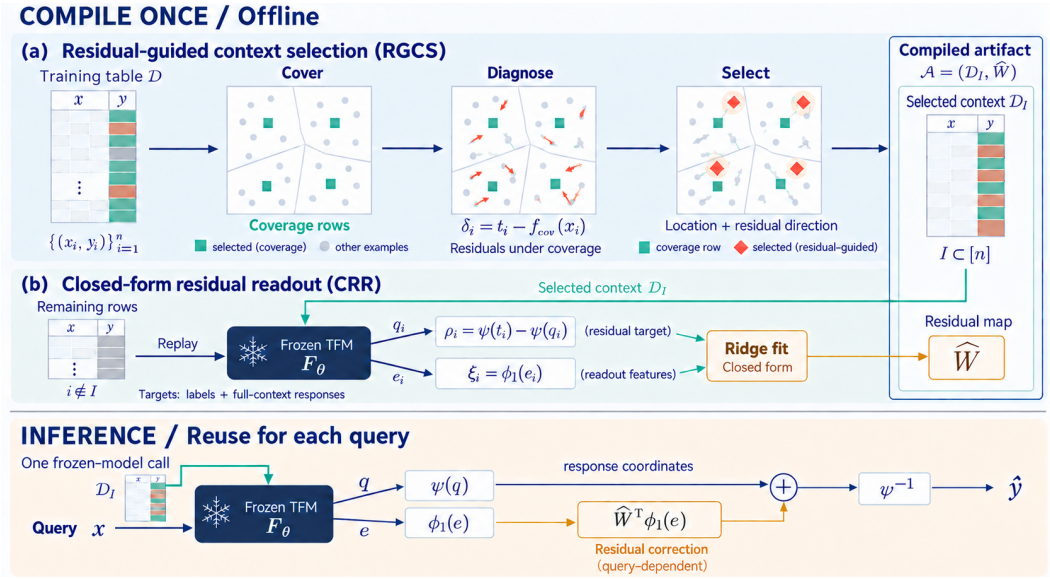

Figure 1: **RCC overview.** (a) RGCS selects coverage rows (green squares) and residual-guided rows (coral diamonds). (b) CRR fits a residual readout using the remaining rows under the selected context. At inference, the readout corrects the backbone prediction using the query representation.

study in Section 4.4 further demonstrates that stronger residual alignment leads to greater reductions in prediction loss.

To compute these residuals, we use a supervision target that combines observed labels with full-context predictions. The labels provide task supervision, while the full-context predictions serve as a reference for the frozen backbone's behavior before compression. Inspired by knowledge distillation (Hinton et al., 2015), we combine label supervision with the backbone's predictions. We use their equal-weight average as the supervision target:

$$\mathbf{t}_i = \tfrac{1}{2}\mathbf{y}_i + \tfrac{1}{2} f_{[n]}(\mathbf{x}_i), \tag{2}$$

where $\mathbf{y}_i$ is the one-hot label for classification or the scalar target for regression.

Coverage initialization. By default, we divide the row budget equally between coverage initialization and residual-guided selection, with budgets B_{cov} and $B_{\text{sel}} = B - B_{\text{cov}}$, respectively. We define the coverage representation as

$$\mathbf{c}_i = 2^{-1/2}[\mathcal{N}(\mathbf{Z})_i; \mathcal{N}(\mathbf{L})_i], \tag{3}$$

where $\mathbf{Z}$ contains normalized features and $\mathbf{L}$ contains one-hot labels for classification or standardized targets for regression. The operator $\mathcal{N}$ scales each nonzero block to unit mean squared row norm, giving the feature and label blocks comparable overall scales before concatenation. Appendix C.1 gives the normalization details.

For classification and regression, we select coverage rows via k-means in the coverage representation. For classification, we additionally reserve the row nearest each class's mean representation as an anchor, then cluster all training rows to fill the remaining coverage budget without per-class quotas.

The selected rows are indexed by I_{cov}. We assign each training row to its nearest coverage row using $\mathbf{c}_i$; the rows assigned to coverage row $j \in I_{\text{cov}}$ form its cell $\mathcal{C}_j$. Appendix C.1 gives the construction details.

Residual-guided nomination. Given selected I_{cov}, we evaluate all training rows under the fixed coverage context $\mathcal{D}_{I_{\text{cov}}}$ and compute

$$\boldsymbol{\delta}_i^{\text{cov}} = \mathbf{t}_i - f_{I_{\text{cov}}}(\mathbf{x}_i), \qquad u_i = \text{RankNorm}\big(\|\boldsymbol{\delta}_i^{\text{cov}}\|_2\big), \qquad \mathbf{v}_i = \text{Normalize}\big(\boldsymbol{\delta}_i^{\text{cov}}\big). \tag{4}$$

Here, u_i measures residual severity through normalized empirical ranks, and $\mathbf{v}_i$ is the unit residual direction for nonzero residuals.

We combine the coverage representation and residual direction as

$$\mathbf{g}_i = 2^{-1/2}[\mathcal{N}(\mathbf{C})_i; \mathcal{N}(\mathbf{V})_i], \tag{5}$$

where $\mathbf{C}$ and $\mathbf{V}$ stack the coverage representations and residual directions, respectively. Each cell with unselected rows nominates one candidate:

$$b_j \in \operatorname*{arg\,min}_{b \in \mathcal{C}_j \setminus I_{\text{cov}}} \sum_{i \in \mathcal{C}_j} u_i \|\mathbf{g}_i - \mathbf{g}_b\|_2^2. \tag{6}$$

Each nominee is the unselected row nearest its cell's severity-weighted centroid. In the equal-budget case, we combine one nominee from each eligible cell with the initial coverage rows to form the final context $\widehat{I}$. Appendix C.1 specifies adjustments for unequal budgets and boundary cases and Appendix A.2 gives the centroid derivation.

3.3 Closed-Form Residual Readout

The selected context provides direct evidence to the backbone. We additionally fit a residual readout using omitted-row supervision to correct predictions without updating the backbone. We denote the fitting indices by $I_{\text{fit}} = [n] \setminus \widehat{I}$. For each fitting row, we obtain its native prediction and query representation under the selected context in the same forward pass:

$$\mathbf{q}_i = f_{\widehat{I}}(\mathbf{x}_i), \qquad \mathbf{e}_i = \mathbf{e}_{\widehat{I}}(\mathbf{x}_i), \qquad i \in I_{\text{fit}}. \tag{7}$$

To preserve supervision from omitted rows, we fit a lightweight readout that predicts the correction needed for each query. Specifically, we use the difference between the transformed supervision target and the transformed backbone prediction as the residual target, and learn to predict this residual from the query representation. We define the residual targets and readout features as

$$\boldsymbol{\rho}_i = \psi(\mathbf{t}_i) - \psi(\mathbf{q}_i), \qquad \boldsymbol{\xi}_i = \phi_{\widehat{I}}(\mathbf{e}_i). \tag{8}$$

Here, ψ transforms predictions and targets into a suitable space for residual correction. For classification, it uses centered log-probabilities so that corrected outputs can be mapped back to valid probabilities (Aitchison, 2005), while for regression, it standardizes responses to account for target scale. To capture nonlinear relationships between query representations and prediction errors while retaining efficient linear fitting, $\phi_{\widehat{I}}$ combines normalized query representations with random Fourier features (Rahimi & Recht, 2007) and a bias term. This allows the omitted labels to complement the compact context without updating the backbone or requiring iterative optimization of the readout. Both transformations remain fixed after compilation, with details provided in Appendices C.2 and C.3.

Afterwards, we fit the residual readout through weighted ridge regression:

$$\widehat{\mathbf{W}} = \operatorname*{arg\,min}_{\mathbf{W}} \sum_{i \in I_{\text{fit}}} w_i \|\mathbf{W}^\top \boldsymbol{\xi}_i - \boldsymbol{\rho}_i\|_2^2 + c\|\mathbf{W}\|_{\text{F}}^2.$$

Here $\mathbf{W}$ is the coefficient matrix and w_i is a positive sample weight. Classification weights use inverse class frequency to account for unequal label counts across classes. Regression weights are uniform and we set $c = 10^{-2}$ for all datasets. Here $\|\cdot\|_{\text{F}}$ denotes the Frobenius norm.

We obtain the residual readout in closed form as

$$\widehat{\mathbf{W}} = (\mathbf{A}\mathbf{X} + c\mathbf{I}_p)^{-1}\mathbf{A}\mathbf{R}, \qquad \mathbf{A} = \mathbf{X}^\top \mathbf{D}. \tag{9}$$

Here, p is the number of readout features and $\mathbf{I}_p$ is the p-dimensional identity matrix. The matrices $\mathbf{X}$ and $\mathbf{R}$ stack the feature vectors $\boldsymbol{\xi}_i$ and residual targets $\boldsymbol{\rho}_i$ row-wise in the same order, while the diagonal matrix $\mathbf{D}$ contains the corresponding weights. The derivation and implementation details are provided in Appendix A.3.

For classification, ψ is the centered log-ratio transform, whose inverse on the zero-sum subspace is softmax; the residual correction is also constrained to this subspace. For regression, ψ standardizes the response using the training-label mean and standard deviation, and ψ^{-1} reverses this transformation. Therefore the corrected predictor is

$$h_{I,\mathbf{W}}(\mathbf{x}) = \psi^{-1}\big(\psi\big(f_I(\mathbf{x})\big) + \mathbf{W}^\top \phi_I\big(\mathbf{e}_I(\mathbf{x})\big)\big). \tag{10}$$

At inference, we deploy $\widehat{h} = h_{\widehat{I},\widehat{\mathbf{W}}}$, requiring only one compact-context backbone call per query. Algorithm 1 in Appendix C summarizes the complete RCC procedure.

Lightweight Variant: RCC-lite. On large tables, generating full-context targets requires evaluating all training rows under the full training context and can become a compilation bottleneck, as detailed in Appendix D. To accelerate compilation, we introduce RCC-lite, which uses label-only targets $\mathbf{t}_i = \mathbf{y}_i$ to eliminate this costly stage. Context selection, residual readout fitting, and inference otherwise remain unchanged from RCC.

Corrective capacity of CRR. Finally, we analyze when our residual readout can improve the backbone's predictions under a fixed context. Let $\boldsymbol{\xi}_I(\mathbf{x}) = \phi_I(\mathbf{e}_I(\mathbf{x}))$ denote the fixed readout features, and let $(X, \mathbf{Y})$ follow the deployment distribution, where $\mathbf{Y}$ is a one-hot label or scalar response. We define $\mathcal{R}(h)$ as the expected log loss for classification or half squared error for regression.

Proposition 1 (Risk reduction condition) *Suppose* $\mathbb{E}\|\boldsymbol{\xi}_I(X)\|_2^2 < \infty$ *and* $\mathcal{R}(f_I) < \infty$*, with strictly positive native probabilities for classification. Then*

$$\inf_{\mathbf{W}} \mathcal{R}(h_{I,\mathbf{W}}) < \mathcal{R}(f_I) \quad \Longleftrightarrow \quad \mathbb{E}\big[\boldsymbol{\xi}_I(X)(\mathbf{Y} - f_I(X))^\top\big] \neq \mathbf{0}. \tag{11}$$

The detailed proof is given in Appendix A.1. The proposition shows that there exists a residual readout that strictly reduces population risk when the backbone's prediction residual is not orthogonal to the readout feature space. A nonzero cross-moment identifies a direction of correction that decreases risk; if it vanishes, the uncorrected predictor already minimizes risk within this readout family. Thus, the corrective capacity depends on whether the selected-context representations retain features aligned with the remaining prediction errors.

4 Experiments

We evaluate predictive quality across context compression ratios, the deployment-cost savings achieved by RCC, and the contributions of RGCS and CRR to predictive quality. Code, reproduction instructions, and experimental results are available at `https://anonymous.4open.science/r/rcc-anonymous-submission-29E7/`.

4.1 Experimental setup

Datasets. We use TALENT (Liu et al., 2025) for classification, OpenML (Feurer et al., 2021) for regression, and BeyondArena (Purucker et al., 2026) for large-scale evaluation. Table 1 summarizes the dataset groups and the numbers of training instances available for RCC. Other details are provided in Appendix B.

Table 1: Dataset statistics. Cla./Reg.: classification/regression.

Task	Group (#datasets)	#Instances	#Features
Cla.	TALENT (100)	2.5k–56k	3–970
	BeyondArena (22)	50.7k–1.0M	3–711
Reg.	OpenML (29)	2.8k–48.5k	5–1,024
	BeyondArena (10)	60.8k–1.0M	6–988

Backbones. We evaluate three pretrained TFMs that support both classification and regression. TabPFNv2 (Hollmann et al., 2025) is a pretrained model for in-context tabular classification and regression. LimiX-2M (Wang et al., 2026) unifies structured-data tasks through conditional prediction. TabICLv2 (Qu et al., 2026) is a scalable tabular foundation model supporting both task types. We refer to these models as TabPFN, LimiX, and TabICL below.

Baselines. We consider three groups of baselines: (1) *conventional ML methods*: Random Forest (Breiman, 2001), XGBoost (Chen & Guestrin, 2016), LightGBM (Ke et al., 2017), and CatBoost (Prokhorenkova et al., 2018); (2) *coreset methods*: Random sampling and feature-space KMeans, which select real rows as context for each frozen backbone (Feuer et al., 2023); and (3) a *learned context-compression method*: TACO (Zabërgja et al., 2026), using its released compressor and companion predictor, with its predictor-only Transformer (POT) as the full-context reference. We also report each backbone with full context. Results for linear models, Extra Trees (Geurts et al., 2006), and HistGradientBoosting are included in Appendix E.1.

Settings. All TFM backbones remain frozen. For context budget B and original context size n, the nominal compression ratio is n/B. We evaluate seven ratios from $8\times$ to $512\times$, emphasizing $16\times$ and $128\times$. Quality is averaged equally across datasets: macro-F1 (%) for classification and nRMSE ($\times 100$) for regression. Experiments use one RTX 4090 D GPU and an Intel Xeon Platinum 8481C CPU. We time batches of $Q = 1$ and $Q = 128$ test examples and report median speedups across datasets, dividing full context inference time by the compressed predictor's time on each dataset. Appendix B provides configurations, thread settings, and measurement details.

4.2 Predictive quality across compression ratios

Comparison with baselines. Table 2 compares predictive quality on TALENT and OpenML. RCC remains competitive after compressing the context. At $16\times$, RCC with TabICL achieves higher mean macro-F1 than the best conventional learner under the evaluated fixed configurations. At $128\times$, its advantage over TACO reaches 6.57 percentage points. Across all three backbones, RCC retains 96.5%–97.5% of the mean full-context macro-F1, while regression nRMSE increases by 8.3–8.6 points on the reported scale.

Table 2: Predictive quality on TALENT and OpenML. Parentheses indicate context compression ratios. TabPFN, LimiX, and TabICL use the full training context. Blue shading highlights our methods.

Method	F1 (%) ↑	nRMSE ($\times 100$) ↓
Baselines		
Random Forest	77.56	49.73
XGBoost	77.61	53.05
LightGBM	78.27	57.00
CatBoost	79.03	47.29
TACO ($128\times$)	72.27	–
TabPFN	79.21	45.69
+ Random ($128\times$)	61.61	65.10
+ KMeans ($128\times$)	62.43	69.07
+ RCC ($128\times$)	77.24	53.95
+ RCC ($16\times$)	79.08	48.79
LimiX	78.25	47.66
+ Random ($128\times$)	60.50	65.70
+ KMeans ($128\times$)	61.26	67.22
+ RCC ($128\times$)	75.54	56.24
+ RCC ($16\times$)	78.17	49.23
TabICL	80.87	44.64
+ Random ($128\times$)	63.43	65.36
+ KMeans ($128\times$)	64.86	67.96
+ RCC ($128\times$)	78.83	53.23
+ RCC ($16\times$)	79.72	48.96

Table 3: Predictive quality on the large-scale BeyondArena benchmark. TabICL uses full context. RCC-lite uses $128\times$ compression.

Method	F1 (%) ↑	nRMSE ($\times 100$) ↓
TabICL	63.76	54.86
+ RCC-lite ($128\times$)	64.17	57.06

Table 4: TabPFN ablation on TALENT and OpenML. Compressed methods use $128\times$; "w/o residual" removes the selection signal. Bold marks the best compressed method.

Method	F1 (%) ↑	nRMSE ($\times 100$) ↓
TabPFN (full context)	79.21	45.69
Random	61.61	65.10
KMeans	62.43	69.07
RGCS w/o residual	68.23	69.57
RGCS	69.14	64.81
Random + CRR	76.47	56.44
KMeans + CRR	76.24	55.84
Ours (RGCS + CRR)	**77.24**	**53.95**

Varying the compression ratio. Figure 2 compares KMeans and RGCS with native backbone predictions, and our proposed RCC (CRR+RGCS). RGCS improves mean classification scores at every evaluated ratio on all three backbones. It also improves regression scores in all but one backbone-ratio setting. CRR consistently improves RGCS on both tasks, mitigating degradation under stronger compression.

Large-scale datasets. We evaluate RCC-lite with TabICL to study compression without computing full-context targets. Table 3 shows that, at $128\times$ compression, RCC-lite achieves mean macro-F1 64.17, compared with 63.76 for the full-context reference, while regression quality remains close. Appendices E.5 and B.7 report per-dataset results and benchmark splits, respectively.

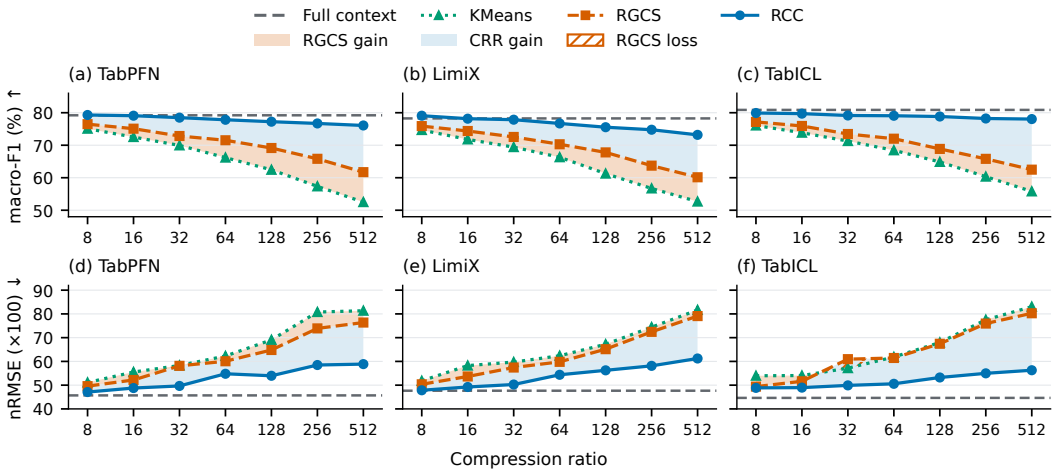

Figure 2: Predictive quality across compression ratios. Curves are dataset means; KMeans and RGCS use native backbone predictions.

4.3 Computational and Storage Efficiency

At $128\times$ compression, we compare inference time and peak GPU memory against each backbone's full context predictor (Full). Table 5 reports inference time, peak GPU memory, compilation time, and storage size; Figure 3 shows speedups for each dataset. Across TALENT/OpenML groups defined by backbone and task, RCC achieves median inference speedups of up to $15.6\times$ for individual queries, with median peak GPU memory as low as 2.6% of Full. On BeyondArena's 21 paired datasets, RCC-lite achieves a median speedup of $119.61\times$ and a maximum of $1{,}514\times$. RCC-lite completes inference on all 32 datasets, whereas Full completes 21.

The compact context and readout are compiled once and reused. Including the residual head, RCC's median artifact-to-full-context size ratio ranges from 5.90% to 51.11% across backbone–task groups and falls to 0.93% for RCC-lite on large-scale datasets. In comparison, TACO's eight-member deployment artifact has a median size of $15.24\times$ the full numerical context. Appendix B.3 details the storage measurements.

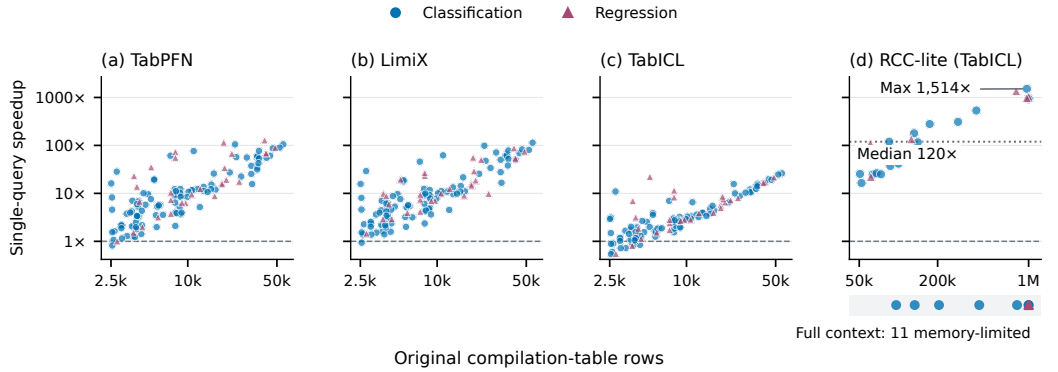

Figure 3: Single-query speedup at $128\times$ compression. Panels (a–c): RCC; (d): RCC-lite on 21 paired tasks. The bottom strip marks 11 tasks on which full context exceeds available memory.

4.4 Ablation Study

Table 4 isolates the contributions of RGCS and CRR using TabPFN at $128\times$ compression, with full context as a reference. RGCS yields better native predictions than Random, KMeans, and RGCS w/o residual signal. Adding CRR improves macro-F1 by a further 8.10 percentage points, and the complete RCC predictor achieves the best result among the compressed configurations on both task types. Appendix E.1 reports the other backbones.

Table 5: Costs at 128× compression (dataset medians). Parentheses give paired speedups or GPU memory and storage size percentages of Full. GPU memory is measured at $Q = 1$.

Method	Compile (s) ↓	Query (ms) $Q = 1$ ↓	Query (ms) $Q = 128$ ↓	Peak GPU (MB) ↓	Storage size ↓
Classification (100 datasets)					
POT (full context)	–	1196.5	1227.5	877.5	212.0 KB
TACO	–	162.6 (6.7×)	165.3 (6.4×)	86.7 (9.9%)	3.4 MB (1524.3%)
TabPFN (full context)	–	1282.4	1305.4	1014.1	212.0 KB
RCC (TabPFN)	24.5	162.3 (7.9×)	171.2 (7.5×)	54.4 (5.4%)	36.5 KB (25.3%)
LimiX (full context)	–	664.2	673.9	857.1	212.0 KB
RCC (LimiX)	46.9	89.8 (7.7×)	94.2 (6.8×)	25.3 (3.0%)	24.8 KB (14.9%)
TabICL (full context)	–	501.9	512.2	2640.4	212.0 KB
RCC (TabICL)	14.7	203.2 (2.6×)	219.6 (2.4×)	205.4 (8.0%)	88.9 KB (51.1%)
Regression (29 datasets)					
TabPFN (full context)	–	2648.6	2667.4	1283.6	377.6 KB
RCC (TabPFN)	66.9	171.4 (15.6×)	178.4 (15.5×)	73.1 (5.9%)	28.0 KB (7.5%)
LimiX (full context)	–	3054.0	3063.5	1078.0	377.6 KB
RCC (LimiX)	321.0	222.1 (10.0×)	225.4 (14.5×)	27.2 (2.6%)	22.1 KB (5.9%)
TabICL (full context)	–	1079.7	1108.5	3523.5	377.6 KB
RCC (TabICL)	42.1	213.1 (5.0×)	227.8 (4.9×)	218.4 (5.7%)	47.9 KB (12.7%)
Large-scale (32 datasets)					
TabICL (full context)	–	90022.7	85914.1	15383.5	53.2 MB
RCC-lite (TabICL)	591.6	405.3 (119.6×)	449.6 (124.4×)	994.8 (7.7%)	481.2 KB (0.9%)

Residual selection signal. On 37 multiclass datasets, we test whether inserting rows one at a time into a fixed 64-row context helps queries with similar prediction errors. We measure changes in each query's true-label log probability and compare six scoring rules by their top-ranked candidates' gains relative to the candidate-pool average. Current-context residual alignment achieves the highest mean top-5% lift across all three backbones (Figure 4). Further details are provided in Appendix B.5.

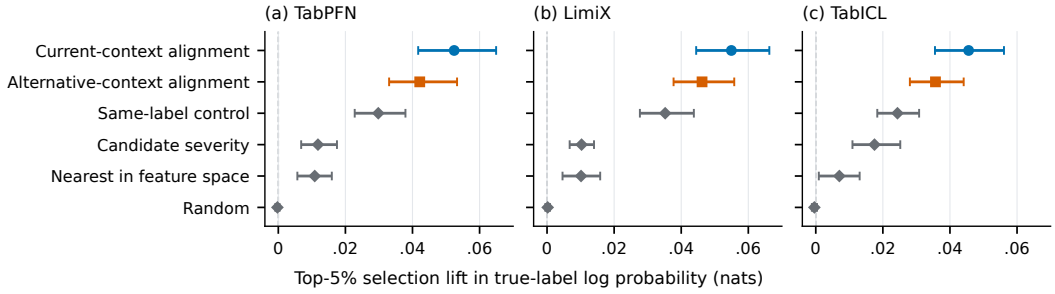

Figure 4: Mean top-5% lift (nats) and dataset-bootstrap 95% intervals (bars) over 37 datasets.

5 Conclusion and Limitations

RCC compiles a compact, reusable predictor through residual-guided context selection and a closed-form readout that incorporates supervision from omitted rows without iterative gradient optimization. Evaluations on classification and regression across three frozen backbones, plus additional tables with up to one million rows, show a favorable balance between predictive quality and inference cost.

Although RCC substantially reduces inference costs, it preserves predictive quality less effectively for regression than for classification at high compression ratios. Regression requires precise numerical estimates over a continuous target space, making target ordering and sparsely sampled regions important for generalization (Yang et al., 2021; Zha et al., 2023). Aggressive context compression may make these requirements harder to satisfy. Improving regression accuracy under limited context budgets remains an important direction for future work.

AI USE STATEMENT

Generative AI tools were used solely to improve the manuscript's grammar, clarity, and readability. The authors reviewed all suggested edits and take full responsibility for the final content.

REFERENCES

Mohammed Abdullah. Understanding context sampling in TabPFN on small tabular datasets. *arXiv preprint arXiv:2607.26628*, 2026.

John Aitchison. The statistical analysis of compositional data. *Journal of the Royal Statistical Society: Series B (Methodological)*, 44(2):139–160, 1982. doi: 10.1111/j.2517-6161.1982.tb01195.x.

John Aitchison. A concise guide to compositional data analysis, 2005. URL https://imae.udg.edu/Activitats/CoDaWork05/A_concise_guide_to_compositional_data_analysis.pdf. CoDaWork'05 course notes.

Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001. doi: 10.1023/A:1010933404324.

Tianqi Chen and Carlos Guestrin. XGBoost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 785–794, 2016. doi: 10.1145/2939672.2939785.

Yilong Chen, Xueying Ding, and Leman Akoglu. VIP-COP: Context optimization for tabular foundation models. *arXiv preprint arXiv:2605.12904*, 2026.

Benjamin Feuer, Chinmay Hegde, and Niv Cohen. Scaling TabPFN: Sketching and feature selection for tabular prior-data fitted networks. *NeurIPS Workshop on Table Representation Learning*, 2023.

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*, volume 37, pp. 83430–83464. Curran Associates, Inc., 2024. doi: 10.52202/079017-2654. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/97dc07f1253ab33ee514f395a82fa7cc-Paper-Conference.pdf.

Matthias Feurer, Jan N. van Rijn, Arlind Kadra, Pieter Gijsbers, Neeratyoy Mallik, Sahithya Ravi, Andreas Müller, Joaquin Vanschoren, and Frank Hutter. OpenML-Python: An extensible python API for OpenML. *Journal of Machine Learning Research*, 22(100):1–5, 2021.

Pierre Geurts, Damien Ernst, and Louis Wehenkel. Extremely randomized trees. *Machine Learning*, 63(1):3–42, 2006. doi: 10.1007/s10994-006-6226-1.

Al Zadid Sultan Bin Habib, Md Younus Ahamed, Prashnna Kumar Gyawali, Gianfranco Doretto, and Donald A. Adjeroh. GOTabPFN: From feature ordering to compact tokenization for tabular foundation models on high-dimensional data. In *International Conference on Machine Learning*, 2026. arXiv:2606.05441.

Jamie Heredge, Mattia J. Villani, Pranav Deshpande, Akshay Seshadri, and Niraj Kumar. CRUMB: Efficient prior fitted network inference via distributionally matched context batching. *arXiv preprint arXiv:2606.11473*, 2026.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

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

Noah Hollmann, Samuel Müller, Lennart Purucker, Arjun Krishnakumar, Max Körfer, Shi Bin Hoo, Robin Tibor Schirrmeister, and Frank Hutter. Accurate predictions on small data with a tabular foundation model. *Nature*, 637(8045):319–326, 2025. doi: 10.1038/s41586-024-08328-6.

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*, volume 30, 2017. URL https://proceedings.nips.cc/paper/2017/hash/6449f44a102fde848669bdd9eb6b76fa-Abstract.html.

Jinmo Lee, Doyun Choi, Moongi Choi, and Jaemin Yoo. Bounded context management for tabular foundation models on stream learning. In *ICML Workshop on Foundation Models for Structured Data*, 2026. URL https://arxiv.org/abs/2606.18677.

Si-Yang Liu, Hao-Run Cai, Qi-Le Zhou, Huai-Hong Yin, Tao Zhou, Jun-Peng Jiang, and Han-Jia Ye. TALENT: A tabular analytics and learning toolbox. *Journal of Machine Learning Research*, 26 (226):1–16, 2025. URL https://www.jmlr.org/papers/v26/25-0512.html.

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*, volume 38, 2025. URL https://openreview.net/forum?id=pIZxEOZCId.

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*, volume 31, 2018. URL https://proceedings.neurips.cc/paper/2018/hash/14491b756b3a51daac41c24863285549-Abstract.html.

Lennart Purucker, Andrej Tschalzev, Nick Erickson, Gioia Blayer, David Holzmüller, Alan Arazi, Alexander Pfefferle, Mustafa Tajjar, Gaël Varoquaux, and Frank Hutter. Beyond IID: How general are tabular foundation models, really? *arXiv preprint arXiv:2606.30410*, 2026. URL https://arxiv.org/abs/2606.30410.

Jingang Qu, David Holzmüller, Gaël Varoquaux, and Marine Le Morvan. TabICL: A tabular foundation model for in-context learning on large data. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 50817–50847. PMLR, 2025. URL https://proceedings.mlr.press/v267/qu25d.html.

Jingang Qu, David Holzmüller, Gaël Varoquaux, and Marine Le Morvan. TabICLv2: A better, faster, scalable, and open tabular foundation model. In *International Conference on Machine Learning*, 2026. URL https://arxiv.org/abs/2602.11139.

Ali Rahimi and Benjamin Recht. Random features for large-scale kernel machines. In *Advances in Neural Information Processing Systems*, volume 20, 2007. URL https://papers.nips.cc/paper_files/paper/2007/hash/013a006f03dbc5392effeb8f18fda755-Abstract.html.

David Rundel, Julius Kobialka, Constantin von Crailsheim, Matthias Feurer, Thomas Nagler, and David Rügamer. Interpretable machine learning for TabPFN. In Luca Longo, Sebastian Lapuschkin, and Christin Seifert (eds.), *Explainable Artificial Intelligence*, volume 2154 of *Communications in Computer and Information Science*, pp. 465–476, Cham, 2024. Springer Nature Switzerland. doi: 10.1007/978-3-031-63797-1_23.

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*, volume 37, pp. 108439–108467. Curran Associates, Inc., 2024. doi: 10.52202/079017-3442. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/c40daf14d7a6469e65116507c21faeb7-Paper-Conference.pdf.

Yuanrui Wang, Xingxuan Zhang, Han Yu, Mingchao Hao, Gang Ren, Hao Yuan, Li Mao, Yunjia Zhang, Chun Yuan, and Peng Cui. LimiX-2M: Mitigating low-rank collapse and attention bottlenecks in tabular foundation models, 2026. URL https://arxiv.org/abs/2606.04485.

Derek Qiang Xu, F. Olcay Cirit, Reza Asadi, Yizhou Sun, and Wei Wang. Mixture of in-context prompters for tabular PFNs. In *International Conference on Learning Representations*, 2025.

Yuzhe Yang, Kaiwen Zha, Yingcong Chen, Hao Wang, and Dina Katabi. Delving into deep imbalanced regression. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 11842–11851. PMLR, 2021. URL https://proceedings.mlr.press/v139/yang21m.html.

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*, 2026. URL https://arxiv.org/abs/2602.05649.

Kaiwen Zha, Peng Cao, Jeany Son, Yuzhe Yang, and Dina Katabi. Rank-N-Contrast: Learning continuous representations for regression. In *Advances in Neural Information Processing Systems*, volume 36, pp. 17882–17903. Curran Associates, Inc., 2023. doi: 10.52202/075280-0786. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/39e9c5913c970e3e49c2df629daff636-Paper-Conference.pdf.

Xingxuan Zhang, Gang Ren, Han Yu, Hao Yuan, Hui Wang, Jiansheng Li, Jiayun Wu, Lang Mo, Li Mao, Mingchao Hao, Ningbo Dai, Renzhe Xu, Shuyang Li, Tianyang Zhang, Yue He, Yuanrui Wang, Yunjia Zhang, Zijing Xu, Dongzhe Li, Fang Gao, Hao Zou, Jiandong Liu, Jiashuo Liu, Jiawei Xu, Kaijie Cheng, Kehan Li, Linjun Zhou, Qing Li, Shaohua Fan, Xiaoyu Lin, Xinyan Han, Xuanyue Li, Yan Lu, Yuan Xue, Yuanyuan Jiang, Zimu Wang, Zhenlei Wang, and Peng Cui. LimiX: Unleashing structured-data modeling capability for generalist intelligence. *arXiv preprint arXiv:2509.03505*, 2025.

Appendix

A Theoretical Results and Proofs **14**

A.1 Proof of the readout improvement criterion 14

A.2 Weighted nomination as a centroid problem 14

A.3 Weighted ridge solution 15

B Experimental Setup **15**

B.1 Datasets and splits 15

B.2 Backbones and baselines 15

B.3 Metrics and measurement 16

B.4 Ablation controls 16

B.5 Single-row intervention protocol 17

B.6 TACO configuration 18

B.7 Large-scale evaluation 18

B.8 Dataset inventories 19

C Algorithm and Method Settings **22**

C.1 Budget and coverage construction 22

C.2 Readout instantiation 23

C.3 Representation features 23

D Complexity **24**

E Additional Experimental Results **24**

E.1 Ranks and component ablations 24

E.2 Amortizing compilation cost 25

E.3 Single-row intervention results 26

E.4 TACO task-level comparison 26

E.5 Large-scale task results 27

A Theoretical Results and Proofs

A.1 Proof of the readout improvement criterion

We prove Proposition 1, which characterizes when some residual readout can improve population risk. We condition on the training table and compilation randomness and hold the context, feature map, and prediction procedure fixed as $\mathbf{W}$ varies. Let (X, Y) follow the deployment distribution, where Y is the class index or scalar response; $\mathbf{Y}$ denotes its one-hot encoding for classification and equals Y for regression. X includes all inputs used to predict the focal query.

Proof. Write $J(\mathbf{W}) = \mathcal{R}(h_{I,\mathbf{W}})$ and $\mathbf{M} = \mathbb{E}[\boldsymbol{\xi}_I(X)(\mathbf{Y} - f_I(X))^\top]$. For regression, $h_{I,\mathbf{W}} = f_I + s_y \mathbf{W}^\top \boldsymbol{\xi}_I$ with a fixed $s_y > 0$. Finite native risk makes $Y - f_I(X)$ square-integrable, so the assumed feature moment ensures that $\mathbf{M}$ is finite. Expanding half squared loss gives

$$J(\mathbf{W}) - J(\mathbf{0}) = -s_y \langle \mathbf{W}, \mathbf{M} \rangle_{\mathrm{F}} + \tfrac{1}{2} s_y^2 \mathbb{E} \|\mathbf{W}^\top \boldsymbol{\xi}_I(X)\|_2^2.$$

Hence J is finite, convex, and differentiable, with $\nabla J(\mathbf{0}) = -s_y \mathbf{M}$.

For classification, adding $\mathbf{W}^\top \boldsymbol{\xi}_I$ in centered log-probability coordinates gives

$$h_{I,\mathbf{W}}(X)_k = \frac{f_I(X)_k \exp([\mathbf{W}^\top \boldsymbol{\xi}_I(X)]_k)}{\sum_l f_I(X)_l \exp([\mathbf{W}^\top \boldsymbol{\xi}_I(X)]_l)}.$$

The log-loss change for fixed $\mathbf{W}$ is bounded in absolute value by $2\|\mathbf{W}\|_{\mathrm{F}} \|\boldsymbol{\xi}_I(X)\|_2$, so finite native risk and the feature moment imply finite $J(\mathbf{W})$. The sample gradient is bounded by $2\|\boldsymbol{\xi}_I(X)\|_2$, which is integrable. Differentiation under expectation therefore yields

$$\nabla J(\mathbf{0}) = \mathbb{E}[\boldsymbol{\xi}_I(X)(f_I(X) - \mathbf{Y})^\top] = -\mathbf{M}.$$

Log-sum-exp convexity makes J convex in $\mathbf{W}$. The class residual sums to zero, so $\mathbf{M}\mathbf{1}_C = \mathbf{0}$; a nonzero moment cannot lie solely in the unidentifiable common-logit direction.

In either task, a nonzero gradient at zero gives a strict descent direction, whereas a zero gradient makes zero a global minimizer. This proves the equivalence without requiring the infimum to be attained by a finite parameter. □

A.2 Weighted nomination as a centroid problem

Reduction to a weighted centroid. For a coverage cell $\mathcal{C}_j$, define its total severity and weighted centroid by

$$s_j = \sum_{i \in \mathcal{C}_j} u_i, \qquad \overline{\mathbf{g}}_j = \frac{1}{s_j} \sum_{i \in \mathcal{C}_j} u_i \mathbf{g}_i.$$

When nomination is performed, the rank weights in Appendix C.1 satisfy $u_i \in (0, 1)$, so $s_j > 0$. The expansion around the centroid together with $\sum_{i \in \mathcal{C}_j} u_i(\mathbf{g}_i - \overline{\mathbf{g}}_j) = \mathbf{0}$ yields

$$\sum_{i \in \mathcal{C}_j} u_i \|\mathbf{g}_i - \mathbf{g}_b\|_2^2 = \sum_{i \in \mathcal{C}_j} u_i \|\mathbf{g}_i - \overline{\mathbf{g}}_j\|_2^2 + s_j \|\mathbf{g}_b - \overline{\mathbf{g}}_j\|_2^2. \tag{A.1}$$

The first term does not depend on b. Equation (6) therefore selects the row in $\mathcal{C}_j \setminus I_{\mathrm{cov}}$ nearest $\overline{\mathbf{g}}_j$. The centroid computation and candidate scan cost $O(|\mathcal{C}_j| \dim(\mathbf{g}_i))$. No pairwise distance matrix is required.

Relationship to directional alignment. The concatenated feature gives the distance decomposition

$$\|\mathbf{g}_i - \mathbf{g}_b\|_2^2 = \|\widetilde{\mathbf{c}}_i - \widetilde{\mathbf{c}}_b\|_2^2 + \|\widetilde{\mathbf{v}}_i - \widetilde{\mathbf{v}}_b\|_2^2.$$

Write $\widetilde{\mathbf{v}}_i = \mathbf{v}_i / s_v$, where $s_v = \sqrt{2}\, s_2(\mathbf{V})$ for a nonzero direction block and $s_v = 1$ for an all-zero block, using s_2 from Appendix C.1. The resulting squared distance is

$$\|\widetilde{\mathbf{v}}_i - \widetilde{\mathbf{v}}_b\|_2^2 = \frac{\|\mathbf{v}_i\|_2^2 + \|\mathbf{v}_b\|_2^2 - 2\langle \mathbf{v}_i, \mathbf{v}_b \rangle}{s_v^2}.$$

When both directions are nonzero, they have unit norm and this reduces to $2(1 - \langle \mathbf{v}_i, \mathbf{v}_b \rangle)/s_v^2$, rewarding severity-weighted alignment in the nomination objective. The coverage representation block penalizes separation between rows. The general expression also covers zero residuals.

A.3 WEIGHTED RIDGE SOLUTION

Let $m = |I_{\text{fit}}|$ and let $K = C$ for classification and $K = 1$ for regression. The design matrix $\mathbf{X} \in \mathbb{R}^{m \times p}$ stacks readout features $\boldsymbol{\xi}_i^\top$, and $\mathbf{R} \in \mathbb{R}^{m \times K}$ stacks transformed residual targets $\boldsymbol{\rho}_i^\top$ in the same row order. The coefficient matrix is $\mathbf{W} \in \mathbb{R}^{p \times K}$. With the positive sample weights on the diagonal of $\mathbf{D}$, the fitting objective is

$$J(\mathbf{W}) = \|\mathbf{D}^{1/2}(\mathbf{X}\mathbf{W} - \mathbf{R})\|_{\mathrm{F}}^2 + c\|\mathbf{W}\|_{\mathrm{F}}^2.$$

Setting its gradient to zero gives

$$(\mathbf{X}^\top \mathbf{D}\mathbf{X} + c\mathbf{I}_p)\mathbf{W} = \mathbf{X}^\top \mathbf{D}\mathbf{R}.$$

For any nonzero $\mathbf{a} \in \mathbb{R}^p$,

$$\mathbf{a}^\top(\mathbf{X}^\top \mathbf{D}\mathbf{X} + c\mathbf{I}_p)\mathbf{a} = \|\mathbf{D}^{1/2}\mathbf{X}\mathbf{a}\|_2^2 + c\|\mathbf{a}\|_2^2 > 0.$$

Thus $c > 0$ makes the system positive definite, and Eq. (9) is the unique minimizer.

B EXPERIMENTAL SETUP

B.1 DATASETS AND SPLITS

Classification datasets come from TALENT. We retain tasks with 2,000–50,000 official training rows and at most ten classes, excluding Volkert because of out-of-memory failures under the available computational resources, yielding the 100 datasets in Table 8. We use their official partitions. For regression, we combine the PFN-REG, TabArena-REG, and CTR23 lists distributed with LimiX, deduplicate by dataset name, and retain datasets with at least 2,000 training rows, yielding the 29 tasks in Table 9.[1] Regression uses an 80/20 training/hold-out split with seed 42; the hold-out partition is divided equally into validation and test sets. Compilation uses the union of training and validation rows, with test rows held out.

B.2 BACKBONES AND BASELINES

The TALENT/OpenML evaluation uses eight ensemble members for TabPFN and TabICL, and four for LimiX classification or eight for LimiX regression. Table 6 specifies the model versions and representation layers. The native ensemble prediction and member representations come from the same forward pass.

Table 6: Backbone versions and representation layers for TALENT and OpenML. h is the concatenated representation dimension.

Backbone	Query representation	Members × dimension
TabPFNv2	Query-row embedding before the prediction decoder	8×192; $h = 1536$
LimiX-2M	Normalized label token at the encoder output; no retrieval	4×384 (cls), 8×384 (reg); $h = 1536/3072$
TabICLv2	Query representation before the prediction decoder	8×512; $h = 4096$

TabPFN uses package version 7.1.1 with `tabpfn-v2-classifier.ckpt` and `tabpfn-v2-regressor.ckpt`. TabICL uses version 2.1.1 with `tabicl-classifier-v2-20260212.ckpt` and `tabicl-regressor-v2-20260212.ckpt`. LimiX uses `LimiX-2M.ckpt` from repository revision `61ede97`, with the official no-retrieval configurations.

Conventional learners use the fixed configurations in Table 7 on every dataset, with no per-dataset search. Standard RCC uses the shared method settings in Table 10.

[1] Source lists: LimiX benchmark lists.

Table 7: Conventional learner configurations. These settings are shared across the TALENT/OpenML datasets. Random seeds are 0 for classification and 42 for regression. Unlisted estimator parameters use library defaults.

Learner	Fixed settings
Linear (classification)	Median imputation, standardization, logistic regression; $C = 1$, maximum 1,000 iterations.
Linear (regression)	Median imputation, standardization, ridge regression; $\alpha = 1$.
Random Forest / Extra Trees	Median imputation; `n_jobs=2`.
HistGradientBoosting	Median imputation; task-specific seed.
CatBoost	Default estimator settings; two CPU threads.
LightGBM	Default estimator settings; two CPU threads.
XGBoost	100 trees, depth 6, learning rate 0.05, minimum child weight 1, row and column subsampling 0.9, L_2 penalty 1, histogram tree construction, `n_jobs=2`. Objectives: logistic (binary), soft-probability (multiclass), and squared error (regression).

B.3 Metrics and measurement

Quality and budgets. Quality is evaluated on the complete held-out test set. We report dataset-equal mean macro-F1 for classification and nRMSE for regression; nRMSE divides RMSE by the population standard deviation of the test targets. Classification uses seed 0 and regression seed 42. The nominal compression ratios are $\{8, 16, 32, 64, 128, 256, 512\}\times$; efficiency and ablation use $128\times$. Appendix C.1 defines the row cap and its rounding rules. Realized compression is $n/|\widehat{I}|$.

Query costs. Each predictor prepares its context once and receives three warm-up calls followed by ten timed calls on fixed test prefixes. A call contains $Q = 1$ or $Q = 128$ queries. We measure wall-clock query latency for the backbone prediction and residual readout with the context already prepared; fitting, loading, serialization, and context preparation are excluded. We use the median of the ten timed calls for each dataset. The full-context backbone (Full) and RCC use the same task, backbone configuration, and thread settings. We disable key–value (KV) caching for all methods when measuring inference costs. GPU memory is peak allocated memory in decimal MB. We aggregate absolute costs and per-dataset ratios separately, using medians. Speedup is $t_{\text{Full}}/t_{\text{RCC}}$; GPU memory as a percentage of Full is $100M_{\text{RCC}}/M_{\text{Full}}$.

Hardware and compilation. All experiments use one NVIDIA RTX 4090 D GPU and an Intel Xeon Platinum 8481C CPU, with 16 PyTorch intra-op and 96 inter-op threads. Compilation time includes full-context target generation, context selection, and readout fitting. In the large-scale evaluation, runs are marked memory-limited if they exceed the 76 GiB working-host-memory threshold or exhaust GPU memory.

Persistent storage. We measure losslessly compressed deployment files, excluding test data, runtime caches, and shared backbone weights. RCC stores its selected context, readout, and preprocessing and feature-map state; TACO stores its eight members' compressed contexts and prediction state. Both are compared with the full numeric context (X, y) for the same dataset. We report median storage sizes in decimal KB or MB and median per-task ratios $100S_{\text{artifact}}/S_{\text{Full}}$.

B.4 Ablation controls

Selection without residuals. The RGCS w/o residual signal control retains the coverage rows, cell assignments, budget split, representative-row objective, and tie-breaking rules. It skips residual diagnosis, sets severity weights to one, removes the residual-direction coordinates, and nominates the eligible row nearest each cell's unweighted centroid.

Selector and readout combinations. Random+CRR and KMeans+CRR use the Random and KMeans contexts at $128\times$ with the task's seed. They use the same frozen backbone and readout

settings as RCC. The two selectors are evaluated on all 129 tasks and three backbones. Baseline selectors fill the shared row cap; RCC may return fewer rows under the rules in Appendix C.1. Results are reported in Table 4 for TabPFN and Table 13 for LimiX and TabICL.

B.5 SINGLE-ROW INTERVENTION PROTOCOL

We test whether residual alignment identifies candidates that improve query predictions when added to a context. We measure each candidate's effect through a separate insertion, then ask two questions: do candidates with higher alignment scores yield larger reductions in query prediction loss, and does alignment remain associated with gains after accounting for labels, feature distance, and individual query and candidate effects? This is an offline diagnostic using known query labels.

Setup. On 37 multiclass datasets, we split compilation rows by class approximately 60/20/20 into three disjoint groups: a construction group H, candidates to insert, and queries to evaluate. Each group contains every class. Stratified sampling retains at most 64 candidates and 128 queries. Using the coverage construction in Appendix C.1 with different seeds, we construct two contexts $I, J \subset H$, each with 64 rows and all classes. These contexts may overlap. We use I for all insertion tests and J only to compute alternative residual scores; candidate and query groups remain fixed.

Measuring insertion gains. Each candidate u is added separately to the same base context I, producing a 65-row context. Insertions do not accumulate. For query q, its insertion gain is the change in the frozen backbone's true-label log probability:

$$G(q, u) = \log f_{I\cup\{u\}}(\mathbf{x}_q)_{y_q} - \log f_I(\mathbf{x}_q)_{y_q}.$$

Positive gain means a reduction in the query's log loss. No residual readout is fitted in these tests.

Residuals and candidate scores. For each candidate or query row i, the reference target is $\mathbf{t}_i = \frac{1}{2}\mathbf{y}_i + \frac{1}{2}f_H(\mathbf{x}_i)$, where $\mathbf{y}_i$ is its one-hot label and f_H conditions on the entire construction group. The same target is used under both contexts. We compute

$$\boldsymbol{\delta}_i^S = \mathbf{t}_i - f_S(\mathbf{x}_i), \qquad \mathbf{v}_i^S = \boldsymbol{\delta}_i^S / \|\boldsymbol{\delta}_i^S\|_2, \qquad S \in \{I, J\}.$$

Rows with residual norm below 10^{-8} under either context are excluded from both analyses. All six scoring rules use the remaining candidate–query pairs:

- *Current-context alignment:* $(\mathbf{v}_q^I)^\top \mathbf{v}_u^I$. This tests our hypothesis that similar residual directions identify useful candidates.
- *Alternative-context alignment:* $(\mathbf{v}_q^I)^\top \mathbf{v}_u^J$. Replacing only the candidate's residual context tests whether its alignment with the query depends on using the same context I.
- *Same-label control:* $\mathbf{1}[y_q = y_u]$. Using the query's true label tests whether label agreement alone explains gains.
- *Candidate severity:* $\mathrm{RankNorm}(\|\boldsymbol{\delta}_u^I\|_2)$, using the definition in Appendix C.1 over the combined candidate and query rows. This tests error magnitude rather than direction.
- *Nearest in feature space:* negative Euclidean distance in the preprocessed feature space from Appendix C.1, fitted on the construction group, testing proximity alone.
- *Random:* independent uniform scores for candidate–query pairs.

Candidate ranking (Figure 4). For query q, let U_q contain all eligible candidates and $T_s(q)$ the highest-scoring $\max(1, \lceil 0.05|U_q| \rceil)$ under score s (the top 5%). We compare their mean insertion gain with the mean over all candidates:

$$L_s(q) = \frac{1}{|T_s(q)|} \sum_{u\in T_s(q)} G(q, u) - \frac{1}{|U_q|} \sum_{u\in U_q} G(q, u).$$

A positive top-5% ranking lift means that the selected candidates have higher mean gain than the candidate-pool average. Gains and lifts are in nats.

Adjusted association (Figure 5). We next examine current-context alignment across all eligible candidate–query pairs, rather than only the top-ranked candidates. Within each dataset, backbone, and seed, we fit separate linear models for alignment and gain using indicators for query identity, candidate identity, their ordered label pair, and five feature-distance quantile groups. These controls account for differences such as some queries being easier to improve or some candidates helping many queries. Subtracting the fitted values leaves the variation not explained by this additive model. With indicator matrix $\mathbf{Z}$, projection $\Pi_{\mathbf{Z}}$ onto its column space, alignment vector $\mathbf{a}$, and gain vector $\mathbf{g}$, this gives

$$\widetilde{\mathbf{a}} = (\mathbf{I} - \Pi_{\mathbf{Z}})\mathbf{a}, \qquad \widetilde{\mathbf{g}} = (\mathbf{I} - \Pi_{\mathbf{Z}})\mathbf{g}.$$

We divide pairs into ten nearly equal-sized groups ordered by adjusted alignment and plot each group's mean adjusted gain. A rising curve indicates a positive association after these adjustments; a negative adjusted gain means gain below the fitted value. Separately, we report the highest-alignment group's mean original gain G minus the lowest-alignment group's, using the same groups.

Aggregation. We use seeds 0, 1, and 2 for each dataset and backbone. For each run, ranking lifts are averaged over queries, including for the random control. Lifts and group means are then averaged over seeds within each dataset and equally over datasets for each backbone. For confidence intervals, we resample datasets with replacement 20,000 times, retaining all observations from each sampled dataset and recomputing the aggregate.

B.6 TACO CONFIGURATION

TACO uses the authors' released classifier with eight ensemble members on the same 100 classification datasets, compilation rows, test partitions, and macro-F1 metric as RCC. Compilation chunk sizes are 1,000, 5,000, or 10,000 rows depending on the dataset, with a retained fraction of $1/128$ per ensemble member; chunk-wise rounding gives a median realized compression of $124.58\times$ per member. Table 5 compares TACO with the official full-context predictor-only Transformer (POT), using their respective released checkpoints.

B.7 LARGE-SCALE EVALUATION

The large-scale datasets are drawn from BeyondArena (Purucker et al., 2026). For RCC-lite, we use its official training and test partitions with the official TabArena preprocessing, evaluating split 0 for each selected dataset. RCC-lite uses TabICL with the benchmark's task-specific ensemble sizes: eight members on 20 datasets and one on the 12 largest datasets. Quality is compared with the published full-context benchmark scores on the same split-0 test partition; inference costs are compared with local full-context runs with the matching ensemble size.

B.8 DATASET INVENTORIES

Table 8: Classification datasets from TALENT. #Instances counts training and validation rows used for compilation; test rows are held out.

Dataset	#Instances	#Test instances	#Features	#Classes
madeline	2,512	628	259	2
dna	2,548	638	180	3
splice	2,552	638	60	3
led24	2,560	640	24	10
led7	2,560	640	7	10
Water_Quality_and_Potability	2,620	656	8	2
Telecom_Churn_Dataset	2,666	667	17	2
gina_agnostic	2,774	694	970	2
allbp	3,017	755	29	3
allrep	3,017	755	29	4
dis	3,017	755	29	2
thyroid-ann	3,017	755	21	3
rice_cammeo_and_osmancik	3,048	762	7	2
ada	3,317	830	48	2
abalone	3,341	836	8	3
predict_students_dropout_and_academic_success	3,539	885	34	3
ada_agnostic	3,649	913	48	2
ada_prior	3,649	913	14	2
spambase	3,680	921	57	2
Employee	3,722	931	8	2
Gender_Gap_in_Spanish_WP	3,796	950	13	3
Wilt	3,856	965	5	2
wine-quality-white	3,918	980	11	7
national-longitudinal-survey-binary	3,926	982	16	2
rl	3,976	994	12	2
waveform_database_generator	3,999	1,000	21	3
churn	4,000	1,000	20	2
waveform-5000	4,000	1,000	40	3
waveform_database_generator_version_1	4,000	1,000	21	3
Satellite	4,080	1,020	36	2
sylvine	4,099	1,025	20	2
KDDCup09_upselling	4,102	1,026	49	2
kdd_ipums_la_97-small	4,150	1,038	20	2
phoneme	4,323	1,081	5	2
wall-robot-navigation	4,364	1,092	24	4
page-blocks	4,378	1,095	10	5
optdigits	4,496	1,124	64	10
turiye_student_evaluation	4,656	1,164	32	5
first-order-theorem-proving	4,894	1,224	51	6
satimage	5,144	1,286	36	6
company_bankruptcy_prediction	5,455	1,364	95	2
taiwanese_bankruptcy_prediction	5,455	1,364	95	2
telco-customer-churn	5,634	1,409	18	2
delta_ailerons	5,703	1,426	5	2

Continued on next page

Table 8 (continued)

Dataset	#Instances	#Test instances	#Features	#Classes
thyroid	5,760	1,440	21	3
ringnorm	5,920	1,480	20	2
twonorm	5,920	1,480	20	2
eye_movements_bin	6,086	1,522	20	2
water_quality	6,396	1,600	20	2
Indian_pines	7,315	1,829	220	8
FICO-HELOC-cleaned	7,896	1,975	23	2
GesturePhaseSegmentationProcessed	7,898	1,975	32	5
JapaneseVowels	7,968	1,993	14	9
BLE_RSSI_dataset_for_Indoor_localization	7,987	1,997	3	3
Bank_Customer_Churn_Dataset	8,000	2,000	10	2
heloc	8,000	2,000	22	2
pol	8,065	2,017	26	2
artificial-characters	8,174	2,044	7	10
Firm-Teacher_Clave-Direction_Classification	8,640	2,160	16	4
jm1	8,708	2,177	21	2
eye_movements	8,748	2,188	27	3
pendigits	8,793	2,199	16	10
E-CommereShippingData	8,799	2,200	10	2
PhishingWebsites	8,844	2,211	30	2
mammography	8,946	2,237	6	2
online_shoppers	9,864	2,466	14	2
in_vehicle_coupon_recommendation	10,147	2,537	21	2
house_16H	10,790	2,698	16	2
dry_bean_dataset	10,888	2,723	16	7
gas-drift	11,128	2,782	128	6
eeg-eye-state	11,984	2,996	14	2
mozilla4	12,436	3,109	4	2
compass	13,315	3,329	17	2
credit	13,371	3,343	10	2
htru	14,318	3,580	8	2
MagicTelescope	15,216	3,804	9	2
HR_Analytics_Job_Change_of_Data_ Scientists	15,326	3,832	13	2
microaggregation2	16,000	4,000	20	5
California-Housing-Classification	16,512	4,128	8	2
okcupid_stem	21,341	5,336	13	3
naticusdroid+android+permissions+dataset	23,465	5,867	86	2
default_of_credit_card_clients	24,000	6,000	23	2
Amazon_employee_access	26,215	6,554	7	2
INNHotelsGroup	29,020	7,255	17	2
BNG(breast-w)	31,492	7,874	9	2
BNG(tic-tac-toe)	31,492	7,874	9	2
Click_prediction_small	31,958	7,990	3	2
FOREX_audcad-hour-High	35,060	8,765	10	2
FOREX_audjpy-hour-High	35,060	8,765	10	2
FOREX_audsgd-hour-High	35,060	8,765	10	2
FOREX_audusd-hour-High	35,060	8,765	10	2

Continued on next page

Table 8 (continued)

Dataset	#Instances	#Test instances	#Features	#Classes
FOREX_cadjpy-hour-High	35,060	8,765	10	2
jungle_chess_2pcs_raw_endgame_complete	35,855	8,964	6	3
bank	36,168	9,043	16	2
electricity	36,249	9,063	8	2
mobile_c36_oversampling	41,408	10,352	6	2
BNG(cmc)	44,236	11,060	9	3
shuttle	46,400	11,600	9	7
internet_firewall	52,425	13,107	7	4
Cardiovascular-Disease-dataset	56,000	14,000	11	2

Table 9: Regression datasets from OpenML. #Instances counts training and validation rows used for compilation; test rows are held out.

Dataset	#Instances	#Test instances	#Features
space_ga	2,796	311	6
abalone	3,759	418	8
Mercedes_Benz_Greener_Manufacturing	3,788	421	376
SAT11-HAND-runtime-regression	3,996	444	116
colleges	4,189	465	49
white_wine	4,408	490	11
QSAR-TID-11	5,168	574	1,024
wine_quality	5,847	650	11
cpu_activity	7,373	819	21
kin8nm	7,373	819	8
pumadyn32nh	7,373	819	32
topo_2_1	7,997	888	266
yprop_4_1	7,997	888	251
grid_stability	9,000	1,000	12
Brazilian_houses	9,623	1,069	12
naval_propulsion_plant	10,741	1,193	14
miami_housing	12,539	1,393	15
fifa	16,257	1,806	5
california_housing	18,576	2,064	9
houses	18,576	2,064	8
superconductivity	19,137	2,126	81
kings_county	19,452	2,161	21
health_insurance	20,045	2,227	11
fps_benchmark	22,162	2,462	43
cps88wages	25,340	2,815	6
sarcos	40,036	4,448	21
Food_Delivery_Time	40,906	4,545	9
physiochemical_protein	41,157	4,573	9
diamonds	48,546	5,394	9

C Algorithm and Method Settings

Algorithm 1 Residual-Aware Context Compression

Input: Training table $\mathcal{D}$; frozen backbone F_θ; row budget B; ridge constant c.

1. Form targets $\mathbf{t}_i$ by Eq. (2); for RCC-lite, use $\mathbf{y}_i$.
2. Split B into coverage and selection budgets (Appendix C.1). Construct coverage rows I_{cov} and assign each row to its nearest coverage row to form cells.
3. Evaluate all rows under $\mathcal{D}_{I_{\text{cov}}}$; compute residual severity u_i and direction $\mathbf{v}_i$ by Eq. (4).
4. If all diagnostic residual norms are at most 10^{-12}, set $\widehat{I} = I_{\text{cov}}$ and skip nomination; otherwise proceed as follows. For each cell with an unselected row, nominate a row by Eq. (6). Accept up to B_{sel} nominees in decreasing total cell severity; combine them with coverage rows to obtain $\widehat{I}$.
5. Query the backbone on $i \notin \widehat{I}$ under $\mathcal{D}_{\widehat{I}}$ to obtain native predictions $\mathbf{q}_i$ and representations $\mathbf{e}_i$.
6. Form features $\boldsymbol{\xi}_i$ and residual targets $\boldsymbol{\rho}_i$. Solve for $\widehat{\mathbf{W}}$ by Eq. (9).

Output: $\mathcal{A}_{\text{RCC}} = (\mathcal{D}_{\widehat{I}}, \widehat{\mathbf{W}})$.
Predict: Apply Eq. (10) using one compact-context backbone call.

C.1 Budget and coverage construction

Context row cap. The effective row cap for nominal compression tier r is

$$B = \min\{n,\ \max(C,\ \text{round}(n/r))\}, \tag{C.1}$$

where C is the number of observed classes; regression uses $B = \min\{n, \max(1, \text{round}(n/r))\}$. Rounding is to the nearest integer. The class floor reserves enough budget for the class anchors, which enforce coverage of every observed class. Random, KMeans, and RCC share this row cap for each dataset and compression tier. Random and KMeans retain exactly B rows; RCC may return fewer, with $|\widehat{I}| \le B$. Realized compression is $n/|\widehat{I}|$.

Half–half split.

$$B_{\text{cov}} = \max(C, \lceil B/2 \rceil)\ (\lceil B/2 \rceil \text{ for regression}), \qquad B_{\text{sel}} = B - B_{\text{cov}}. \tag{C.2}$$

Coverage representation. Features are column-mean imputed, column standardized (scale floor 10^{-6}), and row ℓ_2-normalized; classification labels enter as raw one-hot vectors (no per-column standardization) and regression labels as targets standardized using the compilation-table mean and scale from Appendix C.2. For a block $\mathbf{H}$ with n rows, define $s_2(\mathbf{H}) = \sqrt{\text{tr}(\mathbf{H}^\top \mathbf{H})/n}$. The normalization operator in the main text is $\mathcal{N}(\mathbf{H}) = \mathbf{H}/s_2(\mathbf{H})$ for a nonzero block, with $\mathcal{N}(\mathbf{0}) = \mathbf{0}$. Thus each nonzero block has unit mean squared row norm. The coverage representation $\mathbf{c}_i$ combines the normalized feature and label blocks; $\mathbf{g}_i$ combines the normalized coverage representation and residual-direction blocks. Both concatenations weight each block by $2^{-1/2}$. For the derivation in Appendix A.2, write $\widetilde{\mathbf{c}}_i = 2^{-1/2}\mathcal{N}(\mathbf{C})_i$ and $\widetilde{\mathbf{v}}_i = 2^{-1/2}\mathcal{N}(\mathbf{V})_i$, so that $\mathbf{g}_i = [\widetilde{\mathbf{c}}_i; \widetilde{\mathbf{v}}_i]$.

Coverage construction. For classification, we first select one class anchor per observed class: the real row nearest its class mean in the coverage representation. If $B_{\text{cov}} > C$, MiniBatchKMeans on all rows supplies the remaining $B_{\text{cov}} - C$ representatives; otherwise, the anchors complete the coverage set. Regression uses B_{cov} clusters directly. Each centroid is mapped to the nearest unused row within its cluster, or globally if the cluster has no unused row. Training rows are assigned to their nearest retained representative, with each representative assigned to its own cell.

Residual selection at the budget boundary. Each cell containing unselected rows nominates one of them. If the number of nominees exceeds B_{sel}, we keep those from cells with the largest total severity $s_j = \sum_{i \in \mathcal{C}_j} u_i$. If fewer are available, the context remains below budget. If all diagnostic residual norms are at most 10^{-12}, only the coverage context is retained.

RankNorm. For residual norms $a_1, \ldots, a_n$, we use empirical mid-distribution ranks:

$$\mathrm{RankNorm}(a_i) = \frac{\#\{j : a_j < a_i\} + \frac{1}{2}\#\{j : a_j = a_i\}}{n} \in (0, 1).$$

In the implementation, $a_i \le 10^{-12}$ gives $\mathbf{v}_i = \mathbf{0}$; otherwise $\mathbf{v}_i = \boldsymbol{\delta}_i^{\mathrm{cov}}/a_i$.

C.2 Readout instantiation

Response coordinates ψ. At the backbone output, classification probabilities are aligned to the task's observed C classes, floored at 10^{-12}, and renormalized once. The resulting positive probability vector is f_I. We apply centered log-ratio coordinates directly to this vector (Aitchison, 1982):

$$\psi(\mathbf{p}) = \log \mathbf{p} - C^{-1}(\mathbf{1}^\top \log \mathbf{p})\mathbf{1}, \qquad \psi^{-1}(\mathbf{z}) = \mathrm{softmax}(\mathbf{z}). \tag{C.3}$$

Regression standardizes with compilation-table target statistics (μ_y, s_y) and the scale convention $s_y \leftarrow 1$ whenever the population standard deviation is at most 10^{-8}:

$$\psi(y) = (y - \mu_y)/s_y, \qquad \psi^{-1}(z) = \mu_y + s_y z. \tag{C.4}$$

For classification, the logarithm acts componentwise; softmax is the inverse on centered coordinates and is unchanged by a common coordinate shift. Both maps return the native prediction under zero correction. Mixed targets inherit positivity from the full-context probabilities; label-only targets are floored and renormalized before applying the log-ratio map. The regression location cancels from the fitting residual: $\rho_i = (t_i - q_i)/s_y$.

Weights and regularization. For classification, let N_k be the number of fitting rows with label k. We use $w_i \propto 1/N_{y_i}$ and normalize the weights to have mean one over I_{fit}. Regression uses $w_i = 1$. Both RCC and RCC-lite use a fixed $c = 10^{-2}$ across datasets in the unnormalized weighted-sum objective and penalize all coefficients, including the intercept. The solver uses the primal system when $p \le |I_{\mathrm{fit}}|$ and the equivalent dual system otherwise. Table 10 collects the shared settings.

Table 10: Shared settings for standard RCC.

Parameter	Value
Label/full-context target mixture	$1/2$ each
Coverage/residual budget	Approximately $1/2$ each (Appendix C.1)
Ridge constant c	10^{-2}, including intercept penalty
Random-feature dimension D	1024
Bandwidth multiplier	4

C.3 Representation features

Let h be the concatenated representation dimension in Table 6 and $D = 1024$. We normalize each representation as $\bar{\mathbf{e}} = \mathbf{e}/\max(\|\mathbf{e}\|_2, 10^{-12})$ and use

$$\phi_I(\mathbf{e}) = \left[\bar{\mathbf{e}}; \sqrt{2/D}\cos(\boldsymbol{\Omega}^\top \bar{\mathbf{e}} + \boldsymbol{\beta}); 1\right] \in \mathbb{R}^{h+D+1}.$$

Each entry of $\boldsymbol{\Omega} \in \mathbb{R}^{h \times D}$ is an independent Student-t_3 draw divided by bandwidth σ; each phase in $\boldsymbol{\beta}$ is uniform on $[0, 2\pi)$. The linear block retains the normalized representation coordinates, while the random features allow a nonlinear correction with a closed-form linear fit (Rahimi & Recht, 2007).

The bandwidth is four times the lower median Euclidean distance exceeding 10^{-8} between normalized fitting representations, estimated from at most 200,000 row pairs. It is floored at 10^{-6} and set to one if no qualifying distance exists.

The feature map is fixed after compilation and reused for every query.

D COMPLEXITY

For a table with n rows, let $s = |\widehat{I}| \le B$ be the final context size, $b = B_{\text{cov}}$ the initial coverage size, and $m = n - s$ the number of readout-fitting rows. Write d_x for the input dimension, d_o for the output dimension (C for classification and 1 for regression), and $d_c = d_x + d_o$ for the coverage representation dimension. The readout uses representations of dimension h and $D = 1024$ random features, giving $p = h + D + 1$ features including the intercept. Let $r_* = \min(m, p)$ be the smaller ridge-system dimension and $L = \min\{m(m-1)/2, 200000\}$ the number of row pairs used for bandwidth estimation.

Backbone evaluations. Denote context preparation by $P_F(s)$ and evaluation of q queries under that context by $T_F^{\mathrm{N}}(q; s)$ for predictions alone or $T_F^{\mathrm{R}}(q; s)$ for predictions and representations. Standard RCC performs three stages:

$$T_{\text{backbone,comp}} = \underbrace{P_F(n) + T_F^{\mathrm{N}}(n; n)}_{\text{full-context targets}} + \underbrace{P_F(b) + T_F^{\mathrm{N}}(n; b)}_{\text{coverage diagnosis}} + \underbrace{P_F(s) + T_F^{\mathrm{R}}(m; s)}_{\text{final-context replay}}.$$

RCC-lite omits the first stage. These terms remain backbone-specific because the models use different attention and context-caching architectures.

Selection and readout fitting. Let $\mathcal{K}(n, k, d_c)$ be the MiniBatchKMeans cost for coverage construction, with $k = b - C$ for classification and $k = b$ for regression. We take $\mathcal{K}(n, 0, d_c) = 0$ when clustering is skipped. Coverage construction, representative mapping, and cell assignment cost $\mathcal{K}(n, k, d_c) + O(nbd_c)$. Residual ranking and nomination cost $O(n \log n + n(d_c + d_o) + nb + b \log b)$, including the implementation's per-cell scans. The weighted-centroid reduction avoids pairwise candidate comparisons within a cell.

Bandwidth estimation costs $O(Lh)$, and random-feature construction costs $O(mhD + hD)$. The primal ridge solve costs $O(mp^2 + p^3 + mpd_o)$ when $p \le m$; the dual solve costs $O(pm^2 + m^3 + mpd_o)$ otherwise. Together, fitting costs

$$O\big(Lh + mhD + hD + mpr_* + r_*^3 + mpd_o\big).$$

The CPU fitting stage stores representations, the feature design, and the smaller Gram matrix. Including geometry and bandwidth temporaries, its additional memory is bounded by

$$O\big(n(d_c + d_o) + mh + mp + Lh + hD + r_*^2 + pd_o\big).$$

Backbone stages run sequentially, so compilation GPU memory is governed by the largest preparation or forward stage rather than their sum.

Inference and storage. After preparing the selected context once, a batch of q queries costs

$$T_{\text{query}}(q; s) = T_F^{\mathrm{R}}(q; s) + O(hD + qhD + qpd_o).$$

The additional terms regenerate the random-feature parameters from stored seeds (hD), evaluate the features (qhD), and apply the readout (qpd_o). Their working memory is $O(hD + qp + pd_o)$. Excluding shared backbone weights, the serialized artifact occupies

$$O(s(d_x + 1) + pd_o + d_x + d_o)$$

for the selected rows, readout coefficients, preprocessing statistics, and fixed-size feature-map metadata. Fitting representations and full-table residuals are discarded. Query timings and storage measurements are defined in Appendix B.3.

E ADDITIONAL EXPERIMENTAL RESULTS

E.1 RANKS AND COMPONENT ABLATIONS

Table 11 reports additional conventional learners. Table 12 reports mean ranks among the predictors listed in that table. We rank methods within each dataset and backbone panel, using average ranks

for ties, then average equally over datasets. Classification compares 11 methods and regression 10. Conventional learners and TACO use the same scores across panels; their ranks depend on the competing backbone.

Table 11: Additional conventional learners on TALENT and OpenML. Scores are dataset means.

Method	F1 (%) ↑	nRMSE (×100) ↓
Linear	68.60	66.53
Extra Trees	77.15	50.68
HistGradientBoosting	77.51	53.06

Table 12: Mean ranks of selected predictors on TALENT and OpenML. Lower is better.

	Classification			Regression		
Method	TabPFN	LimiX	TabICL	TabPFN	LimiX	TabICL
Linear	9.55	9.25	9.61	8.86	8.59	8.79
Random Forest	6.06	5.89	6.49	5.86	5.79	5.97
Extra Trees	6.42	6.28	6.84	5.79	5.76	6.14
XGBoost	6.08	5.92	6.66	6.59	6.59	6.72
LightGBM	4.96	4.79	5.56	4.81	4.88	5.16
CatBoost	4.24	3.98	4.82	3.59	3.48	3.90
HistGradientBoosting	5.38	5.19	5.84	5.33	5.36	5.64
Full context	4.04	4.21	2.58	3.07	3.14	1.69
TACO 128×	8.60	8.44	8.83	—	—	—
RCC 16×	4.38	4.92	3.70	4.14	4.03	4.38
RCC 128×	6.28	7.13	5.07	6.97	7.38	6.62

Table 13: LimiX and TabICL ablations on TALENT and OpenML. Compressed methods use 128×; "w/o residual" removes the selection signal. Bold marks the best compressed method.

Method	LimiX F1 (%) ↑	TabICL F1 (%) ↑	LimiX nRMSE (×100) ↓	TabICL nRMSE (×100) ↓
Full context	78.25	80.87	47.66	44.64
Random	60.50	63.43	65.70	65.36
KMeans	61.26	64.86	67.22	67.96
RGCS w/o residual	66.60	67.81	68.30	70.17
RGCS	67.79	68.87	65.15	67.43
Random + CRR	74.76	78.40	58.00	**52.91**
KMeans + CRR	74.91	78.28	**55.69**	54.91
Ours (RGCS + CRR)	**75.54**	**78.83**	56.24	53.23

RGCS denotes the residual-guided selector used by RCC. Adding CRR improves all three context constructions. RGCS retains the highest mean classification score after correction, whereas the best corrected selector on regression depends on the backbone.

E.2 Amortizing compilation cost

We estimate how many calls on the same table are needed for query-time savings to recover the compilation cost. For a batch of Q queries, let $t_{\mathrm{Full},Q}$ and $t_{\mathrm{RCC},Q}$ be steady-state call latencies. When $t_{\mathrm{Full},Q} > t_{\mathrm{RCC},Q}$, the required number of calls is

$$N_Q = \left\lceil \frac{T_{\mathrm{compile}}}{t_{\mathrm{Full},Q} - t_{\mathrm{RCC},Q}} \right\rceil .$$

This estimate covers compilation cost only, excluding loading, context preparation at deployment, and first-call differences. We use the same definition for standard RCC on TALENT/OpenML and RCC-lite on BeyondArena. Table 14 reports dataset medians where paired latencies are available and query-time savings are positive.

On BeyondArena, all 21 datasets with paired measurements recover compilation cost, with medians of six calls at $Q = 1$ and five at $Q = 128$. The other 11 datasets lack a Full latency because Full exceeds resource limits; their break-even points cannot be estimated. Each $Q = 128$ call processes 128 queries, so five calls correspond to 640 queries.

Table 14: Compilation-cost break-even at 128× compression. Medians use datasets with measurable, positive query-time savings; counts show these datasets out of all tasks.

		$Q = 1$		$Q = 128$	
Task	Backbone	Median calls to break even	Measurable break-even / total	Median calls to break even	Measurable break-even / total
TALENT/OpenML: RCC					
Cla.	TabPFN	36	99/100	35	99/100
Cla.	LimiX	120	99/100	120	98/100
Cla.	TabICL	70	90/100	70	90/100
Reg.	TabPFN	42	29/29	42	28/29
Reg.	LimiX	124	29/29	123	29/29
Reg.	TabICL	65	27/29	64	27/29
BeyondArena: RCC-lite					
Cla.	TabICL	6	15/22	5	15/22
Reg.	TabICL	5	6/10	5	6/10
All	TabICL	6	21/32	5	21/32

E.3 Single-row intervention results

Alignment remains informative after adjustment. The ranking comparison in Figure 4 asks whether high-scoring candidates are useful. Here we examine whether residual alignment remains associated with insertion gains after accounting for query identity, candidate identity, label pairs, and feature distance. Following Appendix B.5, Figure 5A–C groups pairs from lowest to highest adjusted alignment and plots their mean adjusted gain. The upward trends on all three backbones support an association beyond these additive controls.

For the same groups, we also compare mean original gain G between the highest and lowest deciles. These gaps are 0.0150, 0.0153, and 0.0191 nats for TabPFN, LimiX, and TabICL, respectively; they are distinct from the adjusted gains plotted in the figure. The corresponding 95% intervals are [0.0102, 0.0204], [0.0109, 0.0204], and [0.0143, 0.0243]. The gap is positive on 34, 35, and 37 of the 37 datasets, respectively.

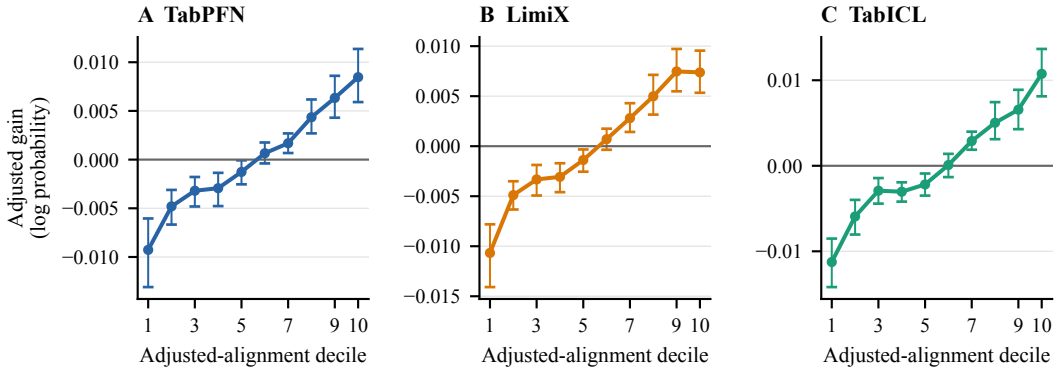

Figure 5: Residual alignment and single-row intervention gains on 37 datasets. Panels A–C show mean adjusted gain across adjusted-alignment deciles, as defined in Appendix B.5. Error bars are dataset-bootstrap 95% intervals after averaging three seeds per dataset.

E.4 TACO task-level comparison

Figure 6 shows task-level quality and speed differences. At $Q = 1$, RCC is both faster and higher in F1 on 59, 58, and 15 tasks for TabPFN, LimiX, and TabICL; at $Q = 128$, the counts are 54, 58, and 3.

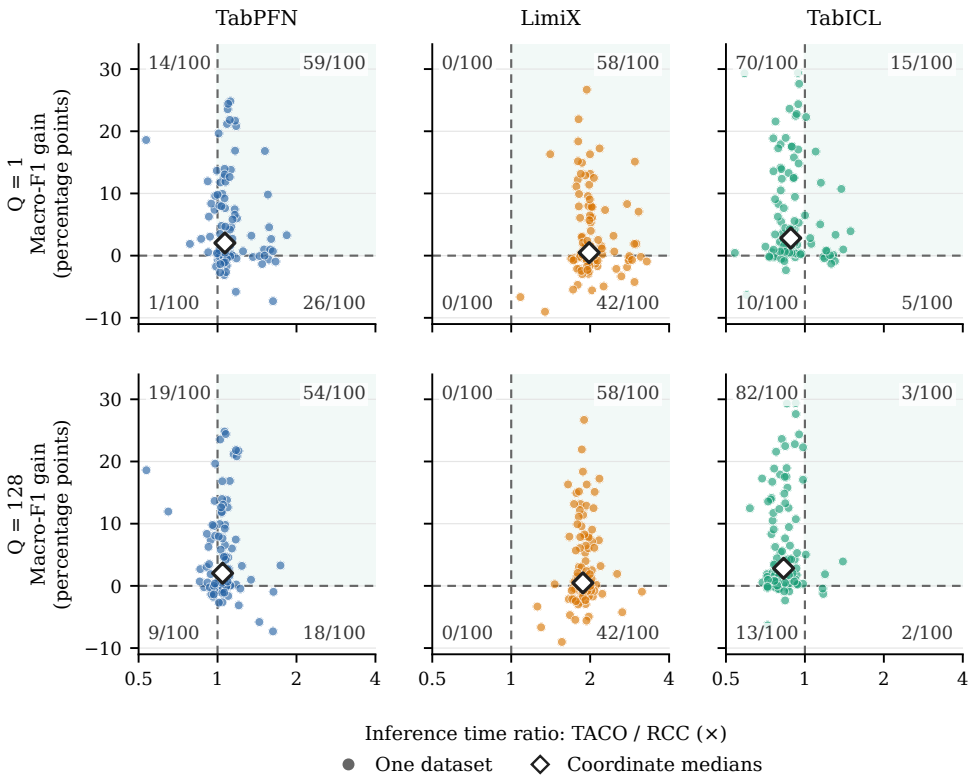

Figure 6: Comparison with TACO on TALENT at $128\times$ compression. Each point is a dataset: horizontal position shows the TACO/RCC inference-time ratio, and vertical position shows RCC minus TACO macro-F1 in percentage points. The upper-right quadrant favors RCC on both metrics. Corner labels count datasets; diamonds mark coordinate-wise medians. Rows use $Q = 1$ and $Q = 128$.

E.5 Large-scale task results

Per-task results Tables 15 and 16 report all classification and regression tasks. Quality uses the benchmark Full reference; speedups compare RCC-lite with the local Full baseline. *Mem.* marks a memory-limited Full run.

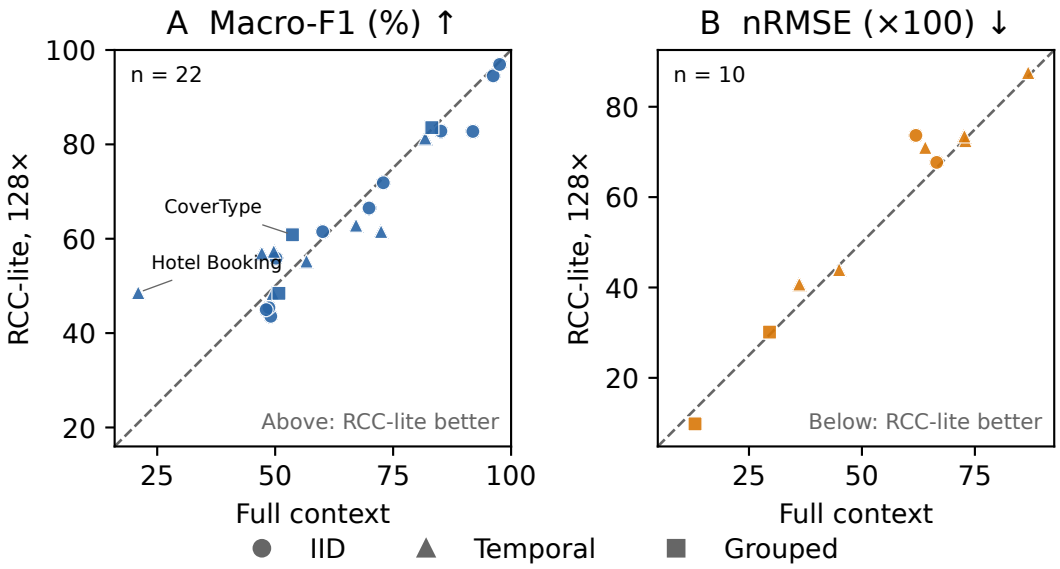

Figure 7: Predictive quality on BeyondArena at 128× compression, using the published full-context benchmark as the reference. Each point is a dataset. Panel A shows macro-F1 (%); panel B shows nRMSE (×100). The dashed line marks equal scores. Points above the line favor RCC-lite in A; points below favor it in B. Marker shapes distinguish IID, temporal, and grouped splits.

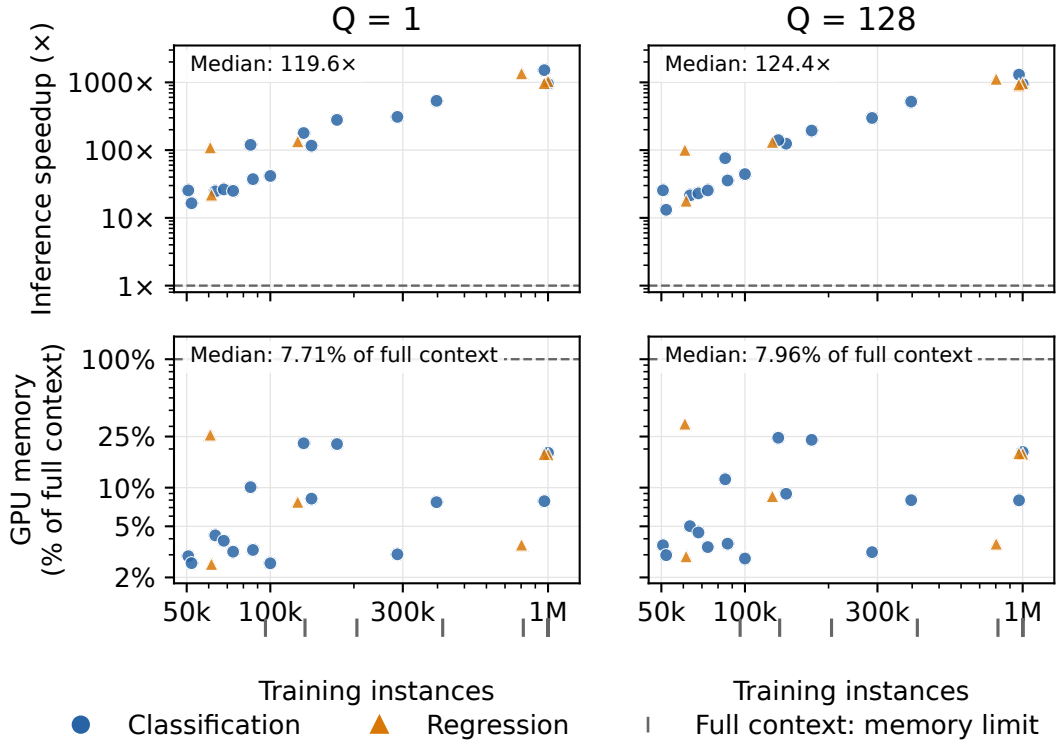

Figure 8: Inference speedup and GPU memory on BeyondArena at 128× compression, relative to the local full-context baseline. Columns show $Q = 1$ and $Q = 128$ for 21 paired datasets. Top: full-context/RCC-lite inference-time ratio. Bottom: RCC-lite peak GPU memory as a percentage of full-context memory. Dashed lines mark 1× speedup and 100% memory. Bottom ticks mark 11 datasets where full context exceeds available memory; RCC-lite completes all 32.

Table 15: Classification results on BeyondArena at 128× compression. #Instances gives training rows in thousands; #Context rows gives retained rows. Quality uses the benchmark full-context reference; Δ is RCC-lite minus full context. Speedup uses the local full-context baseline. Mem. denotes a memory-limited full-context run.

			F1 (%) ↑			Inference speedup (×)	
Dataset	#Instances (k)	#Context rows	Full context	RCC-lite	Δ	$Q = 1$	$Q = 128$
APS Failure	50.7	396	91.90	82.74	-9.16	25.4×	25.5×
SDSS 17	52.0	407	97.57	96.93	-0.64	16.4×	13.2×
Hotel Booking	63.4	495	20.98	48.58	+27.60	24.6×	21.4×
Kick	68.0	531	67.14	62.87	-4.28	26.3×	23.0×
Sepsis Survival	73.5	566	48.09	44.97	-3.12	25.0×	25.4×
WiDS Diabetes	84.9	663	72.91	71.86	-1.05	119.6×	76.1×
Airline Satisfaction	86.6	676	96.21	94.52	-1.69	37.3×	35.6×
PVA Revenue	96.1	750	48.67	45.41	-3.26	Mem.	Mem.
Give Me Some Credit	100.0	781	60.06	61.50	+1.44	41.6×	44.3×
Kickstarter	131.8	1,030	81.80	81.34	-0.46	178.8×	140.3×
Santander	133.3	1,042	69.90	66.51	-3.39	Mem.	Mem.
Acquire Valued Shoppers	140.7	1,099	49.71	57.32	+7.61	116.4×	124.4×
Homesite Quote	173.8	1,358	85.12	82.81	-2.31	279.2×	194.4×
Home Credit Risk	205.0	1,602	50.18	55.87	+5.69	Mem.	Mem.
CoverType	286.9	2,241	53.67	60.84	+7.18	309.4×	298.2×
Porto Seguro	396.8	3,100	49.07	43.56	-5.51	533.9×	519.6×
IEEE Fraud	417.6	3,262	72.45	61.53	-10.92	Mem.	Mem.
Lending Club	814.8	6,365	47.17	56.96	+9.79	Mem.	Mem.
Sepsis Prediction	970.2	7,579	50.79	48.43	-2.37	1513.7×	1299.2×
Consumer Complaints	1000.0	7,812	56.64	55.26	-1.38	945.5×	949.5×
Home Credit Stability	1000.0	7,812	49.49	48.38	-1.10	Mem.	Mem.
Amex	1000.4	7,815	83.19	83.55	+0.36	Mem.	Mem.

Table 16: Regression results on BeyondArena at 128× compression. Notation follows Table 15; lower nRMSE is better. RCC-lite completes all listed datasets.

			nRMSE (×100) ↓			Inference speedup (×)	
Dataset	#Instances (k)	#Context rows	Full context	RCC-lite	Δ	$Q = 1$	$Q = 128$
SF Permit Time	60.8	475	86.80	87.56	+0.76	109.7×	100.8×
5G Energy	61.4	480	29.55	30.12	+0.57	22.0×	17.9×
Allstate Claims	125.5	981	66.56	67.68	+1.13	134.9×	132.1×
Rossmann Sales	803.0	6,273	64.02	70.99	+6.97	1360.3×	1126.5×
Motor Temperature	969.0	7,570	13.06	9.82	-3.25	975.6×	930.6×
Climate Weather	1000.0	7,812	36.14	40.76	+4.61	1043.0×	968.8×
Cooking Time	1000.0	7,812	72.90	72.49	-0.41	Mem.	Mem.
Delivery ETA	1000.0	7,812	72.68	73.52	+0.85	Mem.	Mem.
Maps Router ETA	1000.0	7,812	44.96	43.95	-1.00	Mem.	Mem.
Mercari Price	1000.0	7,812	61.94	73.66	+11.71	Mem.	Mem.