QUTAB: A SINGLE MUTABLE-STATE TRANSFORMER FOR MECHANISTICALLY INTERPRETABLE TABULAR IN-CONTEXT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Tabular foundation models have recently emerged as a promising approach to prediction across diverse datasets, as they can adapt to new tables from labeled examples without dataset-specific training. They obtain expressive in-context learning capabilities by repeatedly updating a large grid of cell or row representations throughout the network, creating indirect information paths and making their internal information flow unlike the token-wise residual-stream computation studied in language models. Instead, they distribute the evolving computation across many states, complicating mechanistic analysis and limiting direct use of the interpretability toolkit developed for such architectures. We introduce QuTab, a 2B-parameter tabular foundation model in which the context table acts as read-only memory and all layerwise computation is carried by a single recurrent prediction state, which we call the query-label state. Context representations remain fixed throughout the Transformer, while the query-label state repeatedly attends to the context and updates itself. This concentrates the model's evolving computation into one residual trajectory, making layerwise information flow easier to isolate and analyze. We realize this architecture by introducing Directed Tabular Attention (DTA), a new attention mechanism that allows the query-label state to simultaneously attend to context cells, same-row query features, and its previous layers' hidden states. Adapted rotary positional encodings (RoPE) encode column, row, and label identity during this joint attention operation, a mechanism we call Structural RoPE. We demonstrate global probing, activation patching, and steering through this residual stream, and localize a sparse causal readout circuit. On TabArena-Lite, QuTab reaches 87.0% accuracy and 69.5% R^2, demonstrating that a model built around a single mechanistically accessible residual stream can recover much of the predictive strength of specialized table-native foundation models alongside favorable compute scaling.

1 INTRODUCTION

Tabular foundation models use in-context learning (ICL) to predict labels for new rows from labeled examples without updating model parameters (Hollmann et al., 2025; Qu et al., 2025). These table-native Transformers maintain cell or row representations that are recurrently updated through attention across features and rows. This distributes the evolving computation across many states, allowing information to be stored, transformed, and rerouted at different locations through depth. Although this organization is well adapted to tabular prediction, it makes the model's internal information flow unlike the residual stream studied in language models and complicates mechanistic analysis.

Pretrained language models provide a complementary route, retaining the residual-stream interface used by established mechanistic interpretability tools. TabGemma serializes rows into text, canonicalizes numerical values, continues pretraining a Gemma model for target imputation, and retrieves similar rows to fit informative examples into its finite token context (Schindler et al., 2025). TabGemma is strongest on semantically rich classification tasks; its reported performance is weaker on regression and the numerically dominated TabArena benchmark, and degrades as tables become wider or contain more rows. These complementary strengths motivate a model that retains table-native structured computation while presenting its prediction process through a mechanistically

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

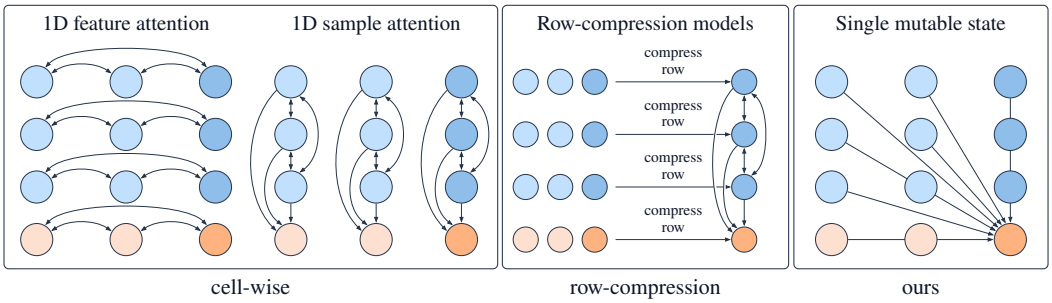


Figure 1: **Architectures for tabular foundation models.** Cell-wise models update a grid of cell states, row-compression models update row states, and QuTab updates a single query-label state.

accessible residual trajectory. In a pre-norm language model, the residual stream is the hidden vector at each token position to which successive attention and multilayer perceptron (MLP) sublayers add their updates. Inspecting or replacing this vector across layers supports mechanistic interpretability techniques such as attention decomposition, probing, activation patching, steering, and causal tracing (Elhage et al., 2021; Meng et al., 2022; Zhang & Nanda, 2024). Explanation methods for tabular foundation models instead predominantly expose column attributions through Shapley additive explanations and related post-hoc attribution methods (Lundberg & Lee, 2017; Rundel et al., 2024). Recent mechanistic studies report fit-specific probe directions, non-transferable steering, redundant layers, and architecture-specific readouts (Gupta et al., 2026b; Balef et al., 2026; Gupta et al., 2026a; Biloš et al., 2026), providing evidence that when the prediction is distributed across an evolving table representation, locating a quantity is difficult and does not by itself identify where or how it causally affects the output.

We introduce QuTab to bridge the gap between table-native computation and residual-stream interpretability without serializing the table. Cells are encoded once and retained as read-only memory, while one query-label state per query row carries all recurrent computation (Figure 1). Directed Tabular Attention (DTA) lets this state jointly read context cells, its self-feature cells, and its earlier hidden states instead of following a fixed sequence of row- and column-attention operations. Structural rotary positional embeddings (RoPE) bind row, column, and label identity into the same read. The resulting model preserves direct access to individual cells, avoids recurrently rewriting the context, and exposes one persistent intervention site for each prediction.

We test whether this residual state provides a useful interface for mechanistic interpretability. Global probes trained across independent tables recover shared directions for coefficients and intermediate regression quantities. Steering causally shifts held-out predictions, and component-level activation patching identifies a sparse readout circuit that transfers donor predictions and remains effective when other components are resampled. QuTab removes repeated full-table updates and yields favorable row scaling when the number of query rows is fixed. It supports inference with up to 10,240 context rows and 2,048 feature columns.

Our contributions are:

- a table-native Transformer with one recurrently updated query-label state per prediction and stage-wise read-only context memory;
- Directed Tabular Attention (DTA), a novel table-native attention variant that combines context evidence, self-features and query-local history;
- Structural RoPE, an extension of rotary positional embeddings that encodes row, column, and label identity; and
- evidence for mechanistic interpretability from globally shared probes, steering, component patching, and a sparse causal readout circuit.

2 Related Work

Table-native tabular in-context learning. Feature-token Transformers contextualize the columns of one row (Gorishniy et al., 2021), and tabular foundation models additionally place labeled examples in context. Grid-based approaches retain cell or feature-group representations and alternate attention across features and samples (Hollmann et al., 2025; Zhang et al., 2025; Spinaci et al., 2025). ConTextTab additionally integrates modality-specific semantic information and real-table pretraining (Spinaci et al., 2025). TabPFN-3 and TabICLv2 instead compress features into row summaries before dataset-level attention (Grinsztajn et al., 2026; Qu et al., 2026). Figure 1 provides an overview over these approaches.

FlexTab freezes target-agnostic encoder outputs for task-specific decoders, separating stored table representations from prediction-side computation (Polewczyk et al., 2026). Its encoder nevertheless contextualizes cell representations before prediction; QuTab keeps cells individually addressable and immutable, with one depth-evolving query-label state as the intervention site for each prediction.

Language models for tabular prediction. Language-model approaches serialize tables into token sequences so they can inherit semantic representations and conventional language-model tooling. TabGemma canonicalizes numerics into scientific notation, continues pretraining Gemma on target-column imputation, and retrieves similar rows at inference to fit relevant examples into its context window (Schindler et al., 2025). It reports strong classification on semantically rich CARTE and TextTab tasks, but weaker regression and performance on numerically dominated TabArena tasks; its analysis also identifies degradation as row and column counts grow.

Rotary positional embeddings and encodings. Rotary positional embeddings (RoPE) rotate queries and keys so their dot product depends on relative position (Su et al., 2024). TabICL applies RoPE for within-row column order and two-dimensional tabular positional encoding combines row-wise and column-wise table traversals (Qu et al., 2025; Li et al., 2025). Vision-language models use multidimensional RoPE, which assigns disjoint channel ranges to geometric axes (Wang et al., 2024). Rotary value embeddings apply corresponding rotations to values, making retrieved messages depend on relative position (García-Castellanos et al., 2026). We instead use RoPE-derived rotations to encode column, row, and label identity.

Mechanistic interpretability. Transformer-circuit analysis decomposes attention and MLP contributions along a residual stream, while activation patching and causal tracing restore internal states and measure behavioral recovery (Elhage et al., 2021; Olsson et al., 2022; Meng et al., 2022; Zhang & Nanda, 2024). Existing studies of tabular foundation models find fit-specific probes, non-transferable steering, redundant layerwise representations, and causal readouts that remain dependent on the task and architecture (Gupta et al., 2026b; Balef et al., 2026; Gupta et al., 2026a; Biloš et al., 2026).

3 QuTab

Let a table contain N labeled context rows, Q query rows, $R = N + Q$ total rows, F feature columns, and one label column y. All cells are encoded with hidden dimension $d_c = 256$, while only the recurrently updated query-label state has a larger hidden dimension $d_q = 2048$. We write encoded context features as $\boldsymbol{X}_c \in \mathbb{R}^{N \times F \times d_c}$ and self-features (the query rows' observed features) as $\boldsymbol{X}_q \in \mathbb{R}^{Q \times F \times d_c}$. Decoupling these widths reduces memory and projection cost for the $R \times F$ fixed cell representations while concentrating recurrent capacity in the Q query-label states.

Figure 2 shows the model architecture. Cell embedding followed by a residual cell Swish-gated linear unit (SwiGLU) first produces the encoded representations from the input table. The L layers are split into S stages, for each of which these fixed representations are projected to grouped key/value (K/V) memory with head dimension d_h. This follows cross-layer K/V sharing (Brandon et al., 2024). We denote the resulting context keys and values by $k^c_{r,f}, v^c_{r,f}$ and the self-feature keys and values by $k^q_{i,f}, v^q_{i,f}$, where r, i, and f index context rows, query rows, and feature columns. Each layer applies DTA (Section 3.3) and a SwiGLU update only to the query-label state. After all $L = 30$ layers, a final root-mean-square normalization (RMSNorm) and task-specific Gaussian error linear unit (GELU) decoder produce the prediction (Google Research, 2026a).

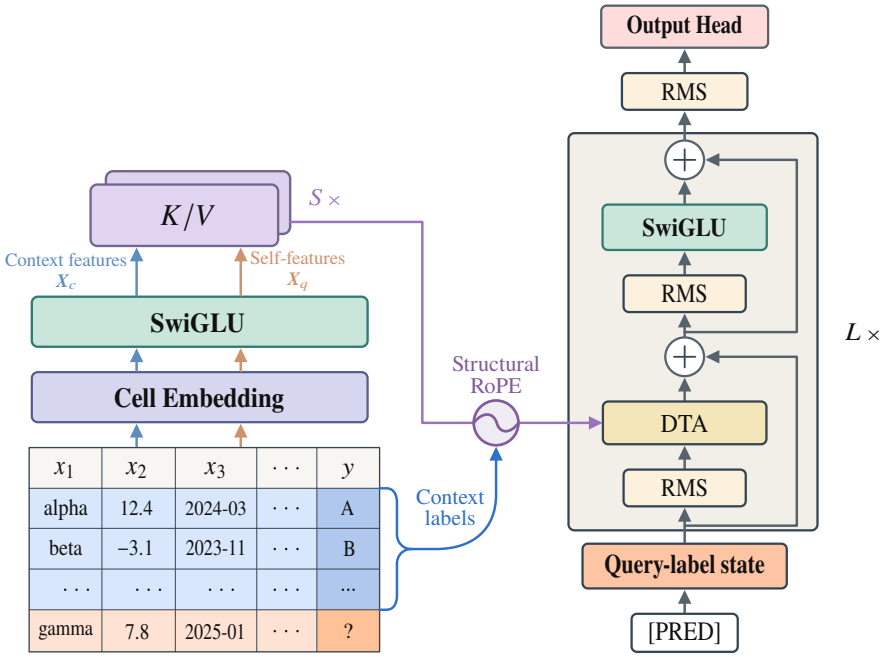


Figure 2: **QuTab model architecture.** Context features and self-features are encoded once, then projected into stage-wise read-only K/V memory. The query-label state passes through repeated DTA and SwiGLU residual updates before the final RMSNorm and output head.

3.1 Input Encoding

Following TabFM, each preprocessed scalar cell is embedded by a learned Fourier map with separate numerical and categorical frequency banks; concatenated sine and cosine features are projected by a type-specific linear map (Google Research, 2026c). Unlike TabFM's feature grouping, QuTab encodes cells independently. We add a learned projection of the column identity, apply affine LayerNorm (Ba et al., 2016), and refine each cell with a residual SwiGLU (Shazeer, 2020). This produces immutable context and self-feature tensors $\boldsymbol{X}_c$ and $\boldsymbol{X}_q$ (Appendix B.1). The query-label state $\boldsymbol{q}_{i,0}$ is initialized with a learned prediction token [PRED].

3.2 Structural RoPE

During attention, the query-label state must distinguish cells with similar semantic values and different structural roles. Standard RoPE rotates queries and keys so that their dot product depends on relative position (Su et al., 2024). Structural RoPE instead binds column, row, and label identity to the cell keys and values, while the query remains unrotated. As in multidimensional RoPE (Wang et al., 2024), different structural axes occupy disjoint fields. Intuitively, the unrotated field represents what a cell contains, while the three rotated fields record where it came from and which label belongs to its row. Let $\mathcal{R}_\theta$ denote the standard two-dimensional rotary transformation applied to adjacent pairs of vector dimensions, with a separate phase for each pair. Below, we describe Structural RoPE for one stage, one K/V group and one query row, omitting indices for readability. Every main key and value, denoted as x, is partitioned into content, column, row, and label fields,

$$x = [x^{\text{content}} \,\|\, x^{\text{col}} \,\|\, x^{\text{row}} \,\|\, x^{\text{label}}]. \tag{1}$$

We leave the first field unchanged. For context cell (r, f) with label y_r, binding applies

$$B^{\text{ctx}}_{r,f}(x) = [x^{\text{content}} \,\|\, \mathcal{R}_{\phi_f} x^{\text{col}} \,\|\, \mathcal{R}_{\psi_r} x^{\text{row}} \,\|\, \mathcal{R}_{\lambda(y_r)} x^{\text{label}}]. \tag{2}$$

Here ϕ_f and ψ_r are the phase codes for feature column f and context row r, and $\lambda(y_r)$ is a label-dependent phase code. For classification, $\lambda(y_r)$ selects one fixed phase code for class y_r. Row,

column, and classification-label phases are sampled once, independently and uniformly on the full circle from a fixed seed. This provides stable identity codes without imposing the higher similarity between neighboring IDs induced by regular RoPE. For regression, label-dependent phases vary continuously with a bounded, standardized target coordinate. Self-feature cells use the same column code ϕ_f, a dedicated query-row code, and a fixed unknown-label code. Identical rotations are applied to K and V, allowing identity to affect selection and the retrieved main value (García-Castellanos et al., 2026).

We additionally provide direct row, column, and label identities through dedicated content-independent carriers. Intuitively, this allows the model to address cells by row, column, or label independently of their content. The row carrier ρ_r and column carrier χ_f contain sine/cosine pairs from fixed random identity codebooks. For classification, η_{y_r} is a one-hot label carrier; for regression, η_{y_r} combines the standardized target value with sine/cosine features of its bounded coordinate. Appendix B.2.2 gives the dimensions, normalization, and full regression-label construction. We append these fields to the keys and values:

$$\bar{k}^c_{r,f} = [B^{\text{ctx}}_{r,f}(k^c_{r,f}) \,\|\, \rho_r \,\|\, \chi_f \,\|\, \eta_{y_r}], \qquad \bar{v}^c_{r,f} = [B^{\text{ctx}}_{r,f}(v^c_{r,f}) \,\|\, \rho_r \,\|\, \chi_f \,\|\, \eta_{y_r}]. \tag{3}$$

The analogous construction produces augmented self-feature keys $\bar{k}^q_{i,f}$ and values $\bar{v}^q_{i,f}$ for query row i. Appendix B.2.1 details the Structural RoPE elements and codebook assignments; Appendix B.4 develops the supported circuit primitives. Figures 13 and 18 profile their learned use in query/key (QK) scores and value/output (OV) writes (Elhage et al., 2021).

3.3 Directed Tabular Attention

DTA (Figure 3) places context- and self-features in one softmax instead of forcing information through separate row- and column-attention operations. Each head can therefore learn a point read, a broader row- or column-conditioned read, a history read, or a mixture of these patterns (Appendix B.3).

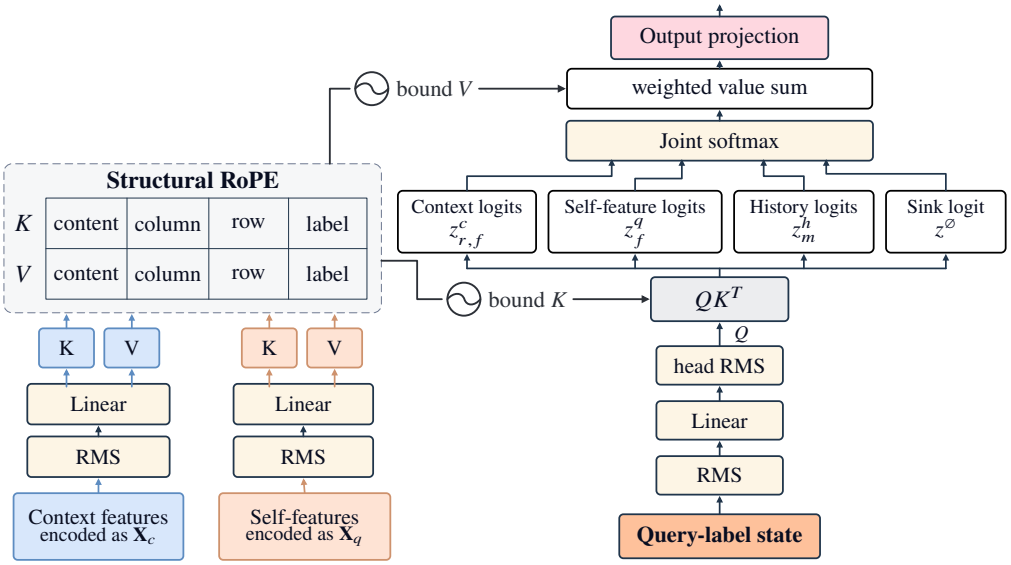


Figure 3: **Directed Tabular Attention (DTA).** Context and self-feature projections provide keys and values for Structural RoPE. The query-label state jointly scores structurally bound context cells, its self-feature cells, earlier query history, and a zero-valued sink. Values are weighted by the joint softmax probabilities, combined, and projected to the residual stream.

We adapt mixture-of-depths attention (MoDA) (Zhu et al., 2026) to the query-label state, allowing it to jointly attend to dedicated K/V projections of its own preceding-layer states as part of DTA. A dedicated history writer RMS-normalizes the input $\boldsymbol{q}_{i,m-1}$ to layer m and projects one key and value for each head. At layer ℓ, the available history is $\mathcal{H}_{i,<\ell} = \{(k^h_{i,m}, v^h_{i,m}) : 1 \le m < \ell\}$. At layer 1, the history branch is omitted.

At large table shapes, one DTA head normalizes scores over millions of cells. We therefore use explicit controls for attention allocation and residual-update magnitude. A learnable zero-value sink modulates allocated attention mass (OpenAI, 2025), a bounded output gate controls the residual update (Qiu et al., 2025), and a learnable temperature sharpens or softens each readout (Henry et al., 2020). Under equal uncorrected logits, a branch's aggregate softmax mass is proportional to its number of entries. Bigger tables would therefore reduce the attention mass allocated to the self-features and earlier hidden states. We introduce depth- and row-count corrections alongside learnable scalar self-, history, and sink biases to compensate for this effect. Appendix A.6.2 reports the learned use of these components across heads and layers.

Each layer projects the query-label state to a main query q of dimension d_h and auxiliary queries u per head. The auxiliary queries score the row, column, and label carriers and additionally a learned row-mean key. For one query-label state and head, we use compact augmented score notation based on Equation 3:

$$\widehat{q} = [q \,\|\, u^{\mathrm{row}} \,\|\, u^{\mathrm{col}} \,\|\, u^{\mathrm{label}} \,\|\, u^{\mathrm{mean}}], \qquad \widehat{k}^c_{r,f} = [\bar{k}^c_{r,f} \,\|\, \kappa_r], \tag{4}$$

where κ_r is the stage-specific projection of the mean encoded cell in row r. It supplies a shared content-dependent row score. Self-feature keys $\widehat{k}^q_f = [\bar{k}^q_f \,\|\, \kappa^q_i]$ are constructed analogously using the projected mean of the fixed query row i. The corresponding context and self-feature values are denoted as $\bar{v}^c_{r,f}$ and $\bar{v}^q_f$, and are not augmented by the row mean. Together, the structural pathways can support circuits such as retrieving values from column 2 in rows selected by column 5 (Appendix B.4.2).

For one fixed query-label state, layer and head, we write DTA compactly with output map W_O as follows (Appendix B.3 describes the exact implementation):

$$z^c_{r,f} = \langle \widehat{q}, \widehat{k}^c_{r,f} \rangle / \sqrt{d_h}, \tag{5}$$

$$z^q_f = \langle \widehat{q}, \widehat{k}^q_f \rangle / \sqrt{d_h} + b_{\mathrm{self}} + \log N, \tag{6}$$

$$z^h_m = \langle q, k^h_m \rangle / \sqrt{d_h} + b^h_m + \log \frac{NF}{|\mathcal{H}_{i,<\ell}|}, \tag{7}$$

$$z^{\varnothing} = b_{\mathrm{sink}} + \log(NF) + \mathrm{softplus}(b_{\mathrm{self}}), \tag{8}$$

$$[p^c_{r,f}, p^q_f, p^h_m, p^{\varnothing}] = \mathrm{softmax}([z^c_{r,f}, z^q_f, z^h_m, z^{\varnothing}]), \tag{9}$$

$$o = W_O \left(\sum_{r=1}^{N} \sum_{f=1}^{F} p^c_{r,f} \bar{v}^c_{r,f} + \sum_{f=1}^{F} p^q_f \bar{v}^q_f + \sum_{1 \le m < \ell} p^h_m v^h_m \right). \tag{10}$$

Here b_{self} is the learned self-feature bias, b^h_m is the learned history bias for reading layer m at layer ℓ, and b_{sink} is the learned sink bias. The softplus function is $\mathrm{softplus}(x) = \log(1 + e^x)$ (Nair & Hinton, 2010).

Restricting recurrent computation to the query-label stream and decoupling the hidden dimensions of the cell and query-label representations reduces computational cost and yields attention-pair scaling linear in the number of context rows N when F and Q are fixed:

$$\mathcal{A}_{\mathrm{QuTab}} = LQ(N+1)F + \frac{1}{2}QL(L-1). \tag{11}$$

Appendix E.2 derives this count and compares it with other tabular foundation models.

4 Evaluation

Training setup. We train separate classification and regression models on the TabICLv2 GraphSCM prior (Qu et al., 2026). Stage 1 uses cosine decay followed by a stable phase for 22M/11M tables; Stage 2 adds 1M tables per task with larger contexts and more columns, preserving optimizer state and learning rates. Classification uses cross-entropy and regression a stabilized per-table RMSE. Both use RMS-matched batched Muon for hidden matrices (Jordan et al., 2024; Liu et al., 2025; Qu et al., 2026) and AdamW for remaining parameters (Loshchilov & Hutter, 2019). Appendix C details the curricula, objectives, and optimizer.

4.1 MECHANISTIC INTERPRETABILITY.

Recent studies of tabular foundation models take important steps beyond feature attribution, but expose interpretability limitations of established architectures. We compare established interfaces and experimental results with QUTAB (Table 1).

Table 1: Interpretability interfaces and interventions across models. A green check denotes a demonstrated property or successful experiment; a red cross denotes an absent architectural property or unsuccessful experiment; a grey bar denotes a result not demonstrated (Appendix A).

Model	Cell access	Causal bottleneck	Global probing	Steering	Component patching	Sparse circuit
TabPFN v2/2.5	✗	✗	✗	✗	✓	—
TabICLv2	✗	✗	—	—	—	—
ConTextTab	✓	✗	✓	—	✗	—
QuTab (Ours)	✓	✓	✓	✓	✓	✓

Cell access Cell access describes the property that each cell of a table retains an individually addressable representation. This lets an intervention isolate a feature cell instead of a token that has already mixed several features or an entire row. Default TabPFNv2 and TabPFN-2.5 group two and three features per token (Hollmann et al., 2025; Gupta et al., 2026a); TabICLv2 compresses the features of each row before dataset-level attention (Qu et al., 2026). ConTextTab retains separate cell states (Spinaci et al., 2025), and QUTAB additionally makes encoded cells immutable.

Causal bottleneck A causal bottleneck is one evolving prediction state through which all prediction-relevant information must pass. Intervening on that state therefore addresses a defined route to the output. When computation evolves across many cells or rows, information can also change along paths outside the intervention site. TabPFN evolves feature-group states, TabICLv2 evolves row representations, and ConTextTab recurrently updates a cell grid (Hollmann et al., 2025; Qu et al., 2026; Spinaci et al., 2025). QUTAB instead routes recurrent computation through one query-label state (Appendix A.2).

Global probing In mechanistic interpretability, probing fits a simple readout to model activations to test whether a property is decodable across independent inputs (Hewitt & Manning, 2019). We call this global probing when one probe is trained across independent tables, in contrast to fitting a separate probe within each table. For TabPFNv2, coefficient probing failed globally and only succeeded within each table (Gupta et al., 2026b). For TabICLv2, regression targets remained weakly readable under within-table probes (Biloš et al., 2026). For ConTextTab, a study reports a global probe for a known binary boundary (Aßmann, 2026).

We analyze this property for QUTAB's regression model. Global probes use one query-label state from each of 8,192 tables with disjoint train/validation/test splits. They target separate linear coefficients, product/quotient intermediates, and final answers. The full protocol is detailed in Appendix A.3. As shown in Figure 4a, at validation-selected sites our probe recovers α and β at $R^2 = 0.375$ and 0.338, and product and quotient intermediates at 0.786 and 0.769, respectively, when fit on laws of the form $y = \alpha a + \beta b$, $y = ab + c$, and $y = a/b + c$. Several regression quantities thus occupy shared coordinates across independently sampled QuTab tables.

Steering Activation steering adds a direction estimated from contrasting examples to an internal state and tests whether it predictably changes the model's behavior on new inputs (Rimsky et al., 2024). In TabPFN-2.5, Gupta et al. (2026a) found that their contrastive direction did not transfer (0.0% mean squared error (MSE) improvement and -0.8% recovery). Applying a related protocol to QuTab's query-label state, we find that steering coefficient signs over L16–20 recovers 40.4% of the output gap on 512 unseen tables (Figure 4b).

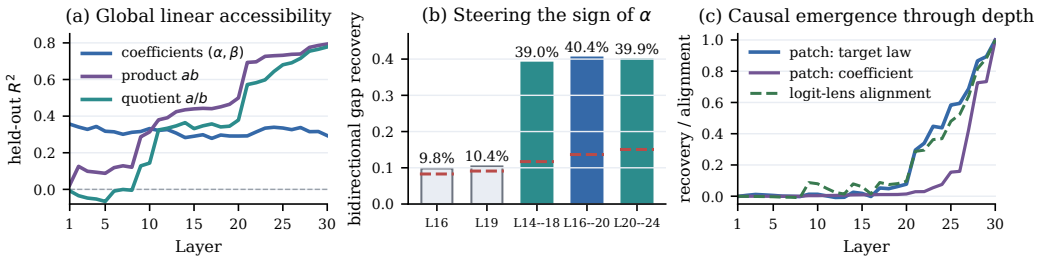


Figure 4: **Global probing, steering, and activation patching.** (a) Linear probes generalize across independent tables, with product and quotient intermediates becoming increasingly accessible through depth. (b) Single-layer steering interventions are weak, whereas distributing the same nominal dose budget over five layers yields about 40% bidirectional recovery; dashed lines mark the 95th percentile (p95) of random controls. (c) Full-query-state activation patching and logit-lens readout independently locate prediction formation in the late-middle stack.

Component patching Activation patching replaces an internal activation with its value from a matched counterfactual run and measures the resulting output change (Zhang & Nanda, 2024; Heimersheim & Nanda, 2024). Here, component patching means applying this intervention to individual attention-head or feed-forward network (FFN) outputs, rather than to an entire residual state or multihead sub-block. This finer granularity can isolate components for a tractable circuit explanation. TabPFN-2.5 supports individual-head patches (Gupta et al., 2026a). A study of TabICLv2 has only published block-level patches (Biloš et al., 2026) and for ConTextTab, post-LayerNorm placement explicitly precludes clean component patching (Aßmann, 2026).

For QuTab, full query-label-state patching rises from 7.7% at L20 to 68.8% at L27 alongside the logit-lens readout (Belrose et al., 2023; Sandmann et al., 2025); Figure 4c shows curves for both. This late-middle causal emergence is consistent with language-model findings that task-relevant causal effects often concentrate in middle-to-late layers (Meng et al., 2022; Wang et al., 2023).

Component patching across all attention heads and FFNs in QuTab identifies a 14-head circuit analyzed below (Appendix Figure 7).

Sparse circuit A sparse circuit is a small set of model components that implements a specified behavior and whose role is tested by causal interventions (Wang et al., 2023; Conmy et al., 2023). For TabPFNv2, a study finds an L9 attention-weighted vote that is spread across all six heads and no individual head dominates per-head ablation; transfer and sufficiency remain untested (Biloš et al., 2026). For TabICLv2, a prototype readout has been characterized, but no corresponding sparse component set is validated (Biloš et al., 2026). ConTextTab's boundary-direction signal requires all 12 layer-0 row-attention heads (Aßmann, 2026).

Our causal experiments use matched pairs of five-column tables with identical feature values and either an additive or multiplicative target. We test causal transfer by bidirectionally patching the 14 heads selected by component patching between matched runs with additive and multiplicative target laws. To test circuit sufficiency, we resample every other head and FFN from the paired run while leaving the circuit unpatched (Wang et al., 2023; Zhang & Nanda, 2024). On 512 pairs evaluated after fixing the circuit, it transfers 95.9% of the donor prediction gap, achieves 0.909 normalized sufficiency under complement resample ablation, reproduces the donor output with 0.990 normalized fidelity, and moves predictions correctly on 98.3% of pairs (Figure 5, left; Appendix A.6). In its computational role, this circuit resembles language-model Name Mover heads (Wang et al., 2023): upstream computation determines which answer is appropriate, while these heads retrieve context-label evidence through OV and add it to the query-label residual. Context contributes 100.0% of the median head effect, predominantly through the label carrier (97.0%) and OV pathway (93.5%). These heads progress from broad context-target summarization to selective row reads and final context-target writes (Appendix Table 7). Figure 5 (left) shows high sufficiency across target rules, but not when query self-features change or the table is unrelated: the circuit can write an answer after the upstream task and query-label state are formed, but cannot recompute that state for a new query or table. It is therefore a readout circuit. Figure 5 (right) shows sharpening row selectivity; in a

planted-support screen, all ten strongest relevant-row heads belong to the circuit while active-column attention stays at baseline, supporting row/label retrieval as its role.

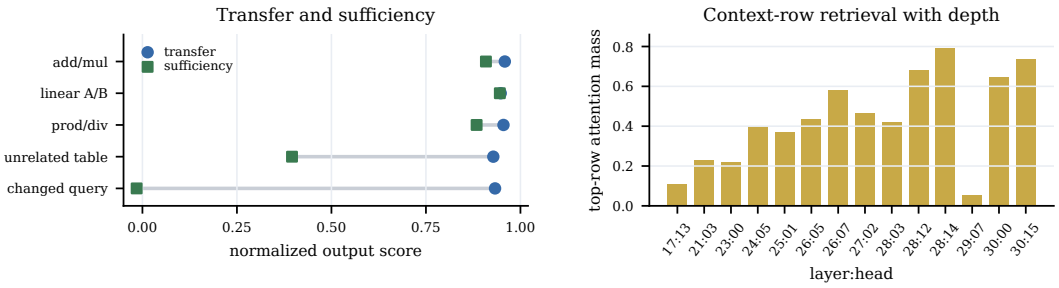


Figure 5: **Sparse circuit.** Component patching identifies the heads tested by counterfactual transfer and complement resample ablation (left). Right: Top-row mass progressively sharpens.

4.2 Predictive Performance

We evaluate on TabArena-Lite (Erickson et al., 2025), reporting mean classification accuracy and mean soft-clipped regression R^2 as in Polewczyk et al. (2026). With at most 10,240 context rows and 100 columns, QuTab achieves 87.0% accuracy across 38 classification datasets and 69.5% R^2 across 13 regression datasets. Appendix D gives the protocol, training-phase comparison, inference-budget sweep, and per-dataset results. Appendix D.3 discusses the regression gap. We hypothesize that QuTab's sparse target-retrieval may provide less of the broad context averaging used by other tabular foundation models to reduce variance in regression (Biloš et al., 2026).

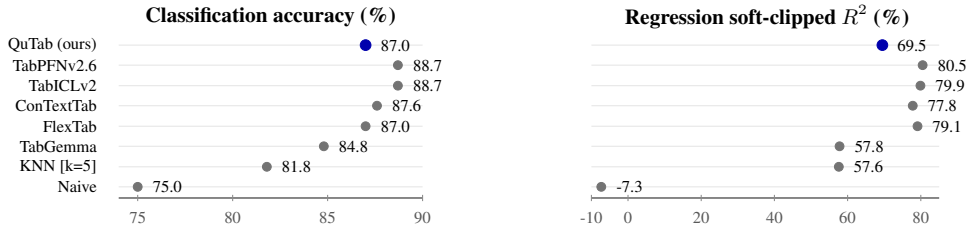


Figure 6: **TabArena-Lite evaluation.** Mean classification accuracy and soft-clipped regression R^2 for QuTab and reference models (Polewczyk et al., 2026; Schindler et al., 2025). k-nearest neighbors (KNN) and Naive are fully interpretable by construction, while TabGemma's language-model residual stream supports the same mechanistic interpretability techniques considered here.

5 Limitations and Future Work

By design, the restricted information flow forbids post-encoder context–context reasoning and may restrict expressiveness on tasks requiring iterative communication between context rows. For large contexts, unified attention might become too diffuse, and unintentional reads could poison retrieved information.

In the evaluated implementation, Structural RoPE uses finite codebooks for 2,048 columns, 10,240 context-row slots, and 10 classes. Training covers at most 512 columns and 10,240 context rows; Appendix D.2 evaluates extrapolation to 10,240 context rows and 2,048 columns. We demonstrate global probing, steering, component patching, and a sparse circuit for the regression checkpoint, but these results do not establish that every language-model interpretability technique transfers.

Future work includes training on larger and more diverse tables, adding semantic column-name embeddings (Spinaci et al., 2025), extending structural codebooks with YaRN-style scaling (Peng et al., 2024), and improving inference efficiency.

Artificial intelligence use statement

Generative artificial intelligence (AI) tools were used in a supporting capacity for this work. AI assistants helped draft, debug, and refactor portions of the training and evaluation code; monitor experiments; support exploratory experiments and ablations; search the literature; design figures; and polish the manuscript. They also provided feedback on experimental protocols and hypotheses and assisted in interpreting results. The authors reviewed the AI-assisted material and remain responsible for all design decisions and claims.

Reproducibility statement

An anonymized review artifact includes inference code and hooks for interpretability. The final release will additionally include the trained checkpoints. Section 3 specifies the information-flow constraint and attention rule; Section 4 records the training setup; Appendix B enumerates architecture dimensions and trainable parameters; Appendix B.4 states the intended structural reads; Appendix C.2 documents the optimizer; and Appendices E.1 and E.2 give the packing configuration and forward floating-point operation (FLOP) accounting.

References

Joshua Ainslie, James Lee-Thorp, Michiel de Jong, Yury Zemlyanskiy, Federico Lebron, and Sumit Sanghai. GQA: Training generalized multi-query transformer models from multi-head checkpoints. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 4895–4901, 2023. doi: 10.18653/v1/2023.emnlp-main.298. URL https://aclanthology.org/2023.emnlp-main.298/.

Emmanuel Ameisen, Jack Lindsey, Adam Pearce, Wes Gurnee, Nicholas L. Turner, Brian Chen, Craig Citro, David Abrahams, Shan Carter, Basil Hosmer, Jonathan Marcus, Michael Sklar, Adly Templeton, Trenton Bricken, Callum McDougall, Hoagy Cunningham, Thomas Henighan, Adam Jermyn, Andy Jones, Andrew Persic, Zhenyi Qi, T. Ben Thompson, Sam Zimmerman, Kelley Rivoire, Thomas Conerly, Chris Olah, and Joshua Batson. Circuit tracing: Revealing computational graphs in language models. *Transformer Circuits Thread*, 2025. URL https://transformer-circuits.pub/2025/attribution-graphs/methods.html.

Julian Aßmann. Row-attention extracts, column-attention projects: How ConTextTab solves in-context linear classification. ICML 2026 Workshop on Mechanistic Interpretability (poster), 2026. URL https://openreview.net/forum?id=mWYat25dvr.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016. URL https://arxiv.org/abs/1607.06450.

Amir Rezaei Balef, Mykhailo Koshil, and Katharina Eggensperger. Is one layer enough? understanding inference dynamics in tabular foundation models. In *International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=IxpVGcg39J.

Nora Belrose, Igor Ostrovsky, Lev McKinney, Zach Furman, Logan Smith, Danny Halawi, Stella Biderman, and Jacob Steinhardt. Eliciting latent predictions from transformers with the tuned lens. *arXiv preprint arXiv:2303.08112*, 2023. URL https://arxiv.org/abs/2303.08112.

Yoav Benjamini and Yosef Hochberg. Controlling the false discovery rate: A practical and powerful approach to multiple testing. *Journal of the Royal Statistical Society: Series B (Methodological)*, 57(1):289–300, 1995. doi: 10.1111/j.2517-6161.1995.tb02031.x. URL https://academic.oup.com/jrsssb/article/57/1/289/7035855.

Marin Biloš, James T. Wilson, Anderson Schneider, and Yuriy Nevmyvaka. A mechanistic study of tabular foundation models. *arXiv preprint arXiv:2605.21288*, 2026. URL https://arxiv.org/abs/2605.21288.

William Brandon, Mayank Mishra, Aniruddha Nrusimha, Rameswar Panda, and Jonathan Ragan-Kelley. Reducing transformer key-value cache size with cross-layer attention. In *Advances in Neural Information Processing Systems*, volume 37, 2024. doi: 10.52202/079017-2758.

URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/9e23d020c18e4c40d81c6a0fc7a46f68-Abstract-Conference.html.

Arthur Conmy, Augustine N. Mavor-Parker, Aengus Lynch, Stefan Heimersheim, and Adrià Garriga-Alonso. Towards automated circuit discovery for mechanistic interpretability. In *Advances in Neural Information Processing Systems*, volume 36, 2023. doi: 10.52202/075280-0719. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/34e1dbe95d34d7ebaf99b9bcaeb5b2be-Abstract-Conference.html.

Nelson Elhage, Neel Nanda, Catherine Olsson, Tom Henighan, Nicholas Joseph, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, Nova DasSarma, Dawn Drain, Deep Ganguli, Zac Hatfield-Dodds, Danny Hernandez, Andy Jones, Jackson Kernion, Liane Lovitt, Kamal Ndousse, Dario Amodei, Tom Brown, Jack Clark, Jared Kaplan, Sam McCandlish, and Chris Olah. A mathematical framework for transformer circuits. Transformer Circuits Thread, 2021. URL https://transformer-circuits.pub/2021/framework/index.html.

Moonjung Eo, Min-Kook Suh, Hye-Seung Cho, Jiwon Kim, Seoyoon Kim, Sangjun Nam, and Soonyoung Lee. EXAONE Tabular 1.0: Technical report. *arXiv preprint arXiv:2608.25774*, 2026. URL https://arxiv.org/abs/2608.25774.

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: Datasets and Benchmarks Track*, volume 38, 2025. doi: 10.52202/085713-0519. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/1697e3fb412da11dc9488249f9e7bbc9-Abstract-Datasets_and_Benchmarks_Track.html.

Trevor Gale, Deepak Narayanan, Cliff Young, and Matei Zaharia. MegaBlocks: Efficient sparse training with mixture-of-experts. In *Proceedings of Machine Learning and Systems*, 2023. URL https://proceedings.mlsys.org/paper_files/paper/2023/hash/5a54f79333768effe7e8927bcccffe40-Abstract-mlsys2023.html.

Alejandro García-Castellanos, Maurice Weiler, and Erik J. Bekkers. RoVE: Rotary value embeddings attention for relative position-dependent value pathways. *arXiv preprint arXiv:2606.11275*, 2026. URL https://arxiv.org/abs/2606.11275.

Google Research. TabFM: Tabular foundation models. Software repository, 2026a. URL https://github.com/google-research/tabfm.

Google Research. TabFM 1.0.0 pytorch classification checkpoint configuration. Hugging Face model release, 2026b. URL https://huggingface.co/google/tabfm-1.0.0-pytorch/blob/main/classification/config.json.

Google Research. TabFM jax model implementation. Source code, revision fbb665569425fd2f490c6576b3af967876fe11ff, 2026c. URL https://github.com/google-research/tabfm/blob/fbb665569425fd2f490c6576b3af967876fe11ff/tabfm/src/jax/model.py.

Yury Gorishniy, Ivan Rubachev, Valentin Khrulkov, and Artem Babenko. Revisiting deep learning models for tabular data. In *Advances in Neural Information Processing Systems*, 2021. URL https://proceedings.neurips.cc/paper/2021/hash/9d86d83f925f2149e9edb0ac3b49229c-Abstract.html.

Léo Grinsztajn, Klemens Flöge, Oscar Key, Felix Birkel, Philipp Jund, Brendan Roof, Mihir Manium, Shi Bin Hoo, Magnus Bühler, Anurag Garg, Dominik Safaric, Jake Robertson, Benjamin Jäger, Simone Alessi, Adrian Hayler, Vladyslav Moroshan, Lennart Purucker, Philipp Singer, Alan Arazi, Julien Siems, Jan Hendrik Metzen, Georg Grab, Nick Erickson, Siyuan Guo, Eliott Kalfon, Simon Bing, David Salinas, Clara Cornu, Lilly Charlotte Wehrhahn, Diana Kriuchkova, Kursat Kaya, Lydia Sidhoum, Marie Salmon, Jerry Chen, Madelon Hulsebos, Yann LeCun, Samuel Müller, Bernhard Schölkopf, Sauraj Gambhir, Noah Hollmann, and Frank Hutter. TabPFN-3: Technical report. *arXiv preprint arXiv:2605.13986*, 2026. URL https://arxiv.org/abs/2605.13986.

Atharva Gupta, Dhruv Kumar, Murari Mandal, and Saurabh Deshpande. Where computation lives inside TabPFN: Causal localisation of attention head function. In *ICML 2026 Workshop on Foundation Models for Structured Data*, 2026a. URL https://openreview.net/forum?id=LXSawSSeA9. Poster.

Aviral Gupta, Armaan Sethi, and Dhruv Kumar. TabPFN through the looking glass: An interpretability study of TabPFN and its internal representations. *arXiv preprint arXiv:2601.08181*, 2026b. URL https://arxiv.org/abs/2601.08181.

Stefan Heimersheim and Neel Nanda. How to use and interpret activation patching, 2024. URL https://arxiv.org/abs/2404.15255.

Alex Henry, Prudhvi Raj Dachapally, Shubham Shantaram Pawar, and Yuxuan Chen. Query-key normalization for transformers. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pp. 4246–4253, 2020. doi: 10.18653/v1/2020.findings-emnlp.379. URL https://aclanthology.org/2020.findings-emnlp.379/.

John Hewitt and Christopher D. Manning. A structural probe for finding syntax in word representations. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 4129–4138, 2019. doi: 10.18653/v1/N19-1419. URL https://aclanthology.org/N19-1419/.

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

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

Dharshan Kumaran, Arthur Conmy, Federico Barbero, Simon Osindero, Viorica Patraucean, and Petar Veličković. How do LLMs compute verbal confidence? In *International Conference on Machine Learning*, 2026a. URL https://openreview.net/forum?id=t7oQT3uX66.

Dharshan Kumaran, Nathaniel Daw, Simon Osindero, Petar Veličković, and Viorica Patraucean. Causal evidence that language models use confidence to drive behaviour. *Nature Machine Intelligence*, 8:1411–1425, 2026b. doi: 10.1038/s42256-026-01293-x. URL https://www.nature.com/articles/s42256-026-01293-x.

Jia-Nan Li, Jian Guan, Wei Wu, Zhengtao Yu, and Rui Yan. 2D-TPE: Two-dimensional positional encoding enhances table understanding for large language models. In *Proceedings of the ACM on Web Conference 2025*, pp. 2450–2463. ACM, 2025. doi: 10.1145/3696410.3714920. URL https://doi.org/10.1145/3696410.3714920.

Shen Li, Yanli Zhao, Rohan Varma, Omkar Salpekar, Pieter Noordhuis, Teng Li, Adam Paszke, Jeff Smith, Brian Vaughan, Pritam Damania, and Soumith Chintala. PyTorch Distributed: Experiences on accelerating data parallel training. *Proceedings of the VLDB Endowment*, 13(12):3005–3018, 2020. doi: 10.14778/3415478.3415530. URL https://www.vldb.org/pvldb/vol13/p3005-li.pdf.

Disen Liao and Freda Shi. How tokenization limits phonological knowledge representation in language models and how to improve them. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 13921–13938, 2026. doi: 10.18653/v1/2026.acl-long.634. URL https://aclanthology.org/2026.acl-long.634/.

Jingyuan Liu, Jianlin Su, Xingcheng Yao, Zhejun Jiang, Guokun Lai, Yulun Du, Yidao Qin, Weixin Xu, Enzhe Lu, Junjie Yan, Yanru Chen, Huabin Zheng, Yibo Liu, Shaowei Liu, Bohong Yin, Weiran He, Han Zhu, Yuzhi Wang, Jianzhou Wang, Mengnan Dong, Zheng Zhang, Yongsheng Kang, Hao Zhang, Xinran Xu, Yutao Zhang, Yuxin Wu, Xinyu Zhou, and Zhilin Yang. Muon is scalable for LLM training. *arXiv preprint arXiv:2502.16982*, 2025. URL https://arxiv.org/abs/2502.16982.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=Bkg6RiCqY7.

Scott M. Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In *Advances in Neural Information Processing Systems*, volume 30, 2017. URL https://papers.nips.cc/paper/7062-a-unified-approach-to-interpreting-model-predictions.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. Locating and editing factual associations in GPT. In *Advances in Neural Information Processing Systems*, 2022. doi: 10.52202/068431-1262. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/6f1d43d5a82a37e89b0665b33bf3a182-Abstract-Conference.html.

Vinod Nair and Geoffrey E. Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on Machine Learning*, pp. 807–814, 2010. URL https://icml.cc/2010/papers/432.pdf.

Catherine Olsson, Nelson Elhage, Neel Nanda, Nicholas Joseph, Nova DasSarma, Tom Henighan, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, Dawn Drain, Deep Ganguli, Zac Hatfield-Dodds, Danny Hernandez, Scott Johnston, Andy Jones, Jackson Kernion, Liane Lovitt, Kamal Ndousse, Dario Amodei, Tom Brown, Jack Clark, Jared Kaplan, Sam McCandlish, and Chris Olah. In-context learning and induction heads. Transformer Circuits Thread, 2022. URL https://transformer-circuits.pub/2022/in-context-learning-and-induction-heads/.

OpenAI. gpt-oss-120b & gpt-oss-20b model card. Technical report, OpenAI, 2025. URL https://cdn.openai.com/pdf/419b6906-9da6-406c-a19d-1bb078ac7637/oai_gpt-oss_model_card.pdf.

Bowen Peng, Jeffrey Quesnelle, Honglu Fan, and Enrico Shippole. YaRN: Efficient context window extension of large language models. In *International Conference on Learning Representations*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/874a4d89f2d04b4bcf9a2c19545cf040-Abstract-Conference.html.

Marek Polewczyk, Maximilian Schambach, Marco Spinaci, Sam Thelin, and Johannes Höhne. FlexTab: A flexible encoder–decoder architecture for in-context learning across diverse tabular tasks. *arXiv preprint arXiv:2606.30336*, 2026. URL https://arxiv.org/abs/2606.30336.

Zihan Qiu, Zekun Wang, Bo Zheng, Zeyu Huang, Kaiyue Wen, Songlin Yang, Rui Men, Le Yu, Fei Huang, Suozhi Huang, Dayiheng Liu, Jingren Zhou, and Junyang Lin. Gated attention for large language models: Non-linearity, sparsity, and attention-sink-free. In *Advances in Neural Information Processing Systems*, volume 38, 2025. doi: 10.52202/085713-3345. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/904e89bb4e632e75fb47f093b620b257-Abstract-Conference.html.

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://openreview.net/forum?id=SxsyLjIfWB.

Nina Rimsky, Nick Gabrieli, Julian Schulz, Meg Tong, Evan Hubinger, and Alexander Turner. Steering Llama 2 via contrastive activation addition. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 15504–15522, 2024. doi: 10.18653/v1/2024.acl-long.828. URL https://aclanthology.org/2024.acl-long.828/.

David Rundel, Julius Kobialka, Constantin von Crailsheim, Matthias Feurer, Thomas Nagler, and David Rügamer. Interpretable machine learning for TabPFN. In *Explainable Artificial Intelligence: Second World Conference, xAI 2024, Proceedings, Part II*, volume 2154 of *Communications in Computer and Information Science*, pp. 465–476. Springer, 2024. doi: 10.1007/978-3-031-63797-1_23. URL https://link.springer.com/chapter/10.1007/978-3-031-63797-1_23.

Elias Sandmann, Sebastian Lapuschkin, and Wojciech Samek. The algorithm is not the behavior: Learned priors override look-ahead in a chess-playing neural network. *arXiv preprint arXiv:2508.21380*, 2025. URL https://arxiv.org/abs/2508.21380.

Günther Schindler, Maximilian Schambach, Michael Medek, and Sam Thelin. TabGemma: Text-based tabular ICL via LLM using continued pretraining and retrieval. In *Workshop on AI for Tabular Data at EurIPS*, 2025. URL https://openreview.net/forum?id=cv2uilalQx.

Noam Shazeer. GLU variants improve transformer. *arXiv preprint arXiv:2002.05202*, 2020. URL https://arxiv.org/abs/2002.05202.

Marco Spinaci, Marek Polewczyk, Maximilian Schambach, and Sam Thelin. ConTextTab: A semantics-aware tabular in-context learner. In *Advances in Neural Information Processing Systems*, 2025. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/d807e7678ba3afd3a904f4af52819e77-Abstract-Conference.html.

Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. RoFormer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024. doi: 10.1016/j.neucom.2023.127063. URL https://doi.org/10.1016/j.neucom.2023.127063.

Philippe Tillet, H. T. Kung, and David Cox. Triton: An intermediate language and compiler for tiled neural network computations. In *Proceedings of the 3rd ACM SIGPLAN International Workshop on Machine Learning and Programming Languages*, pp. 10–19, 2019. doi: 10.1145/3315508.3329973. URL https://doi.org/10.1145/3315508.3329973.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html.

Constantin Venhoff, Ashkan Khakzar, Sonia Joseph, Philip H. S. Torr, and Neel Nanda. Too late to recall: Explaining the two-hop problem in multimodal knowledge retrieval. In *Advances in Neural Information Processing Systems*, volume 38, 2025. doi: 10.52202/085713-1613. URL https://proceedings.nips.cc/paper_files/paper/2025/hash/4526cfacdbca6b6e184568dac91bf070-Abstract-Conference.html.

Jesse Vig. A multiscale visualization of attention in the transformer model. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pp. 37–42, 2019. doi: 10.18653/v1/P19-3007. URL https://aclanthology.org/P19-3007/.

Kevin Ro Wang, Alexandre Variengien, Arthur Conmy, Buck Shlegeris, and Jacob Steinhardt. Interpretability in the wild: A circuit for indirect object identification in GPT-2 small. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=NpsVSN6o4ul.

Peng Wang, Shuai Bai, Sinan Tan, Shijie Wang, Zhihao Fan, Jinze Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Yang Fan, Kai Dang, Mengfei Du, Xuancheng Ren, Rui Men, Dayiheng Liu, Chang Zhou, Jingren Zhou, and Junyang Lin. Qwen2-VL: Enhancing vision-language model's perception of the world at any resolution. *arXiv preprint arXiv:2409.12191*, 2024. URL https://arxiv.org/abs/2409.12191.

Han-Jia Ye, Si-Yang Liu, and Wei-Lun Chao. A closer look at TabPFN v2: Understanding its strengths and extending its capabilities. In *Advances in Neural Information Processing Systems*, volume 38, 2025. doi: 10.52202/085713-4527. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/c57b3718381cb2bf9c1ccd63377f2448-Abstract-Conference.html.

Fred Zhang and Neel Nanda. Towards best practices of activation patching in language models: Metrics and methods. In *International Conference on Learning Representations*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/06a52a54c8ee03cd86771136bc91eb1f-Abstract-Conference.html.

Xiyuan Zhang, Danielle C. Maddix, Junming Yin, Nick Erickson, Abdul Fatir Ansari, Boran Han, Shuai Zhang, Leman Akoglu, Christos Faloutsos, Michael W. Mahoney, Cuixiong Hu, Huzefa Rangwala, George Karypis, and Bernie Wang. Mitra: Mixed synthetic priors for enhancing tabular foundation models. In *Advances in Neural Information Processing Systems*, 2025. doi: 10.52202/085713-0535. URL https://papers.nips.cc/paper_files/paper/2025/hash/177d68f4adef163b7b123b5c5adb3c60-Abstract-Conference.html.

Lianghui Zhu, Yuxin Fang, Bencheng Liao, Shijie Wang, Tianheng Cheng, Zilong Huang, Chen Chen, Lai Wei, Yutao Zeng, Ya Wang, Yi Lin, Yu Li, and Xinggang Wang. Mixture-of-depths attention. *arXiv preprint arXiv:2603.15619*, 2026. URL https://arxiv.org/abs/2603.15619.

A Mechanistic Interpretability Details

QuTab provides one query-label residual stream as a stable interface for applying mechanistic interpretability methods. The experiments below demonstrate their use on the trained regression checkpoint and characterize the accessibility and causal influence of specific quantities. Table 1 separately compares two architectural properties (cell access and causal bottleneck) and four experimental demonstrations (global probing, steering, component patching, and a sparse circuit). A green check requires the named property or a successful experiment at the stated granularity; a red cross denotes an absent architectural property or a reported unsuccessful experiment; a grey bar means that the comparable experiment has not been demonstrated. The combined TabPFN v2/2.5 row summarizes evidence from two related model versions: the coefficient probe across independent table fits and the layer-9 (L9) attention-weighted class-label vote concern TabPFNv2, whereas individual-head patching and held-out steering concern TabPFN-2.5. The architectural properties extend to other tabular foundation models that inherit a cell- or row-compression-based approach. The experimental evidence is limited to models that have been mechanistically studied. The six subsections below explain every table entry under these scoped criteria.

A.1 Cell access

Cell access asks whether one feature cell remains an individually addressable representation at the model's table-reading interface. This is desirable for mechanistic interpretability because it supports controlled interventions and attribution analyses at the level of individual cells. Default TabPFNv2 groups two preprocessed features per token and TabPFN-2.5 groups three, so neither retains one state per original feature cell and neither fulfills cell access (Hollmann et al., 2025; Gupta et al., 2026a). TabICLv2's column encoder compresses the features of a row before dataset-level attention, leaving row rather than individual-cell states for its recurrent in-context computation; it likewise does not fulfill cell access (Qu et al., 2026). ConTextTab retains separate per-cell states through its axial computation and fulfills this property (Spinaci et al., 2025). QuTab encodes each scalar cell separately. These encoded cells remain read-only, and the query-label state can address them individually, so QuTab also fulfills this property. Moreover, among the models compared here, it is the only one whose cell representations remain unchanged through depth and cannot accumulate information from other cells. This allows statements such as "QuTab attends to column 2 of row 5" to refer directly to the underlying table.

A.2 Causal bottleneck

Causal interventions are central to validating mechanistic explanations. For these, a clear intervention site whose effects can be traced through the model is a common desideratum (Heimersheim & Nanda, 2024; Elhage et al., 2021). In a causally masked language model, patching the current token provides such a site because earlier token states cannot read it (Vaswani et al., 2017). For tabular foundation models, computation is often distributed across many cell or row states. This introduces two related issues. First, there is no single obvious state on which to intervene: the query-label cell, the entire

query row, and other table states all remain viable choices (Gupta et al., 2026a; Biloš et al., 2026; Aßmann, 2026). Second, when other mutable states can read the query-label state, intervening on it can also change their later computation (Hollmann et al., 2025; Spinaci et al., 2025).

We thus desire what we call a causal bottleneck: one depth-evolving prediction state through which recurrent, prediction-relevant information must pass after the one-time table encoding. Patching that state then targets a defined route to the output.

TabPFNv2 and TabPFN-2.5 update multiple feature-group states, so information can continue to change outside any one patched token; neither has this causal bottleneck (Hollmann et al., 2025; Gupta et al., 2026a). TabICLv2 evolves multiple row representations during dataset-level computation and likewise lacks the bottleneck (Qu et al., 2026). ConTextTab retains per-cell access but repeatedly updates a cell grid and thus also does not fulfill this property (Spinaci et al., 2025). QuTab's context and self-feature representations are encoded once, and each stage forms read-only K/V from those fixed representations. Only the query-label state receives recurrent DTA and SwiGLU updates, and the output head reads that state. Our MoDA adaptation (Zhu et al., 2026) allows the query-label state to attend to history K/V projections of its own preceding-layer states (Section 3.3, Equation 7). Their stored information can reach the final prediction only when read by the current query-label state. QuTab therefore fulfills this property.

A.3 Global probing

Probing is a popular tool for studying a model's computation. It builds on the observation that internal quantities can sometimes be decoded with low-dimensional, often linear, readouts (Hewitt & Manning, 2019; Belrose et al., 2023). We use global probing to mean one probe fitted across independently sampled tables and evaluated on held-out tables. This distinction matters because studies of tabular foundation models have also fitted probes within individual fits or datasets (Gupta et al., 2026b; Biloš et al., 2026). We suspect that the reported failure of global coefficient probing in TabPFNv2 is related to its distributed architecture: different table sizes involve different numbers of states and may route computation differently. For TabPFNv2, coefficient probing across independent table fits fails even though a probe within an individual fit succeeds (Gupta et al., 2026b). Biloš et al. (2026) fit TabICLv2 probes within datasets and find regression targets weakly readable, but do not report one global probe trained and tested across independent tables. Global probing therefore remains undemonstrated for TabICLv2. A ConTextTab study reports a probe shared across tables for a known binary boundary, fulfilling the criterion (Aßmann, 2026). Our desired causal bottleneck provides a convenient location for probing, as the model's prediction-side computation has to pass through this state.

Global probing experiment. Each example is one query-label state from the first query row of an independently sampled synthetic regression table with 1,024 context rows and 64 query rows. We sample 8,192 tables independently for each family and split them into 6,144/1,024/1,024 train/validation/test tables. The linear family draws two feature columns and the product and division families draw three, initially from independent standard Gaussians. Features are whitened using context statistics: we subtract the empirical context mean, multiply by the inverse square root of its covariance, and apply the same transformation to query rows. This removes finite-sample mean and covariance deviations despite Gaussian sampling. We probe three target-law families: $y = \alpha a + \beta b$, $y = ab + c$, and $y = a/b + c$, where a, b, and c are feature columns and y is the generated target. The linear coefficient vector has unit norm and a uniformly sampled angle, and the denominator in $a/b + c$ is bounded away from zero.

The probes target the coefficients α, β, the intermediates $ab, a/b$, and the final targets. Activations and probe targets are standardized using training-set moments. Each linear probe is an affine readout $h \mapsto w^{\mathsf{T}}h + b$ fitted by mean squared error. Separate probes are fit to the pre-attention, post-attention, and post-FFN states of all 30 layers using AdamW for 3,000 steps with batch size 256, initial learning rate 2×10^{-2}, weight decay 10^{-4}, and cosine decay to 2×10^{-4}.

Validation scores select the saved probe checkpoint independently at each site and output coordinate. For each quantity, validation also selects the site reported in Table 2; "Through L27" repeats selection after restricting candidate sites to layers 1–27.

On held-out test tables, we compute $R^2 = 1 - \sum_j (t_j - \widehat{t}_j)^2 / \sum_j (t_j - \bar{t})^2$ at the validation-selected site. Thus $R^2 = 0$ is the mean-target baseline, while $R^2 = 1$ equals perfect recovery.

Independently shuffled probe targets provide negative controls. They are fit under the same protocol and evaluated at the corresponding real quantity's validation-selected site.

Table 2: Global probe results. Sites are selected using validation data, and scores use held-out test tables. "Through L27" excludes the final three layers. "Shuffled R^2" scores a probe trained on randomly permuted targets at the same selected site.

Quantity	Selected site	Test R^2	Through L27	Shuffled R^2
Coefficient α	L1 post-FFN	0.375	0.375	-0.180
Coefficient β	L1 post-attn.	0.338	0.338	-0.238
Product ab	L30 post-attn.	0.786	0.739	-0.379
Quotient a/b	L30 post-attn.	0.769	0.716	-0.294
Product-law answer	L30 post-attn.	0.931	0.910	-0.403
Division-law answer	L30 post-attn.	0.929	0.906	-0.423
Linear answer	L30 post-FFN	0.996	0.991	-0.343

QuTab demonstrates successful global probing of coefficients, intermediate values, and final predictions (Table 2). The product-, division-, and linear-law answer rows are the final targets $ab + c$, $a/b + c$, and $\alpha a + \beta b$, respectively. Figure 4a shows the mean coefficient-probe curve and the product and quotient intermediate curves; the final-answer probes are reported here. Separate affine readouts of one 2,048-dimensional query-label state explain 37.5% and 33.8% of held-out coefficient variance, while the corresponding shuffled-target controls yield $R^2 = -0.180$ and -0.238. The intermediate product and quotient scores of 0.786 and 0.769 are stronger, and all seven shuffled-target controls at the validation-selected sites are negative. Prior regression-probe studies interpret partial scores in this range as evidence of linearly accessible signal: Liao & Shi (2026) report $R^2 = 0.33$–0.39 alongside negative random controls, and Kumaran et al. (2026b) describe a fivefold-cross-validated ridge probe with peak $R^2 = 0.39$ as explaining substantial variance in verbal confidence. The probe scores quantify linear accessibility; the steering experiment separately tests causal influence.

We also perform a control for the ab product probe: At its validation-selected L30 post-attention site, ab is strongly decodable on product-law tables ($R^2 = 0.786$), while a separately fitted probe on matched linear-law tables reaches only $R^2 = 0.007$. Moreover, a probe fitted directly to the input features is at chance ($R^2 = -0.003$). Thus the signal is specific to the model's nonlinear product computation rather than a directly available input statistic or a consequence of target standardization.

A.4 Steering

Mechanistic interpretability uses steering to test whether an internal direction causally influences a model's predictions (Rimsky et al., 2024). A direction is an additive vector in the model's residual coordinates. Contrastive steering estimates that direction from activation differences between conditions and applies it to new inputs (Rimsky et al., 2024).

In TabPFN-2.5, the reported contrastive direction does not transfer (0.0% MSE improvement and -0.8% recovery), so held-out steering is unsuccessful under that protocol (Gupta et al., 2026a). No matched held-out steering result is reported for TabICLv2 or ConTextTab (Biloš et al., 2026; Aßmann, 2026).

Having a single writable state makes this intervention well defined for QuTab. Here, an intermediate state is the query-label residual after a specified block, before the final decoder. The paired-table setup and recovery metrics below are also used by the component and circuit analyses in the following subsections.

We produce counterfactual pairs of tables with identical features, structural slots, and query rows that differ only in the law used to calculate their targets. Here we use an additive law and a multiplicative law. Each table has $N = 1{,}024$ context rows, $Q = 64$ query rows, and $F = 5$ continuous features. Feature values are drawn from a standard Gaussian, then whitened: we subtract the empirical context mean and multiply by the inverse square root of the empirical context covariance, applying the same

transformation to query rows. Finite Gaussian samples have empirical means and covariances that deviate from zero and identity, respectively; whitening removes those fluctuations. Three active columns, a, b, and c, are drawn uniformly without replacement; the other two are nuisance features. The raw target laws are $y_{\text{add}} = a + b + c$ and $y_{\text{mul}} = abc$, each mapped through its own context empirical cumulative distribution function (CDF) to a standard-normal marginal. The mapped labels populate the paired context and query targets. Separate forward passes produce the clean predictions for both tables; depending on intervention direction, one is the source (donor) f_s and the other the destination (recipient) f_d. Adding a steering direction or patching donor activations into the recipient run produces the intervened prediction f_p.

With j indexing evaluated predictions, signed output-gap recovery is

$$\text{Rec}_{\text{gap}} = \frac{\sum_j (f_{p,j} - f_{d,j})(f_{s,j} - f_{d,j})}{\sum_j (f_{s,j} - f_{d,j})^2}. \tag{12}$$

The donor prediction gap $f_s - f_d$ is the model's output difference between paired tables with identical features but different target laws. The numerator measures how far the intervened recipient moves along that signed output difference: recovery is zero when the intervention leaves the recipient unchanged and one when it reproduces the donor prediction; it can also be negative or exceed one. For the additive–multiplicative contrast, we average equally over the two intervention directions. With $\mathcal{L}_s$, $\mathcal{L}_d$, and $\mathcal{L}_p$ denoting source, destination, and intervened MSE against the same target, loss recovery is

$$\text{Rec}_{\text{loss}} = \frac{\mathcal{L}_d - \mathcal{L}_p}{\mathcal{L}_d - \mathcal{L}_s}. \tag{13}$$

For coefficient-sign steering, each training contrast pairs the same features with $\alpha > 0$ versus $\alpha < 0$ while keeping the other coefficient fixed. At each post-FFN query-label site, the learned direction is the mean 2,048-dimensional difference between the negative- and positive-α query-label states over 2,048 independent training pairs. Injecting the direction or its negative tests whether held-out predictions move toward the paired contrast. Doses and layer windows are selected on 256 validation tables, and final effects are measured on 512 unseen tables. Multi-layer interventions use the same nominal dose budget: a K-layer window gives each layer $1/\sqrt{K}$ of the selected dose, relative to that layer's residual RMS. All rows in Table 3 use validation-selected dose 1.0. Controls comprise 16 coordinate-shuffled and 16 norm-matched random directions. Each control is evaluated without retuning at the same route and dose as the learned direction. "Ctrl. p95" is the 95th percentile of bidirectional gap recovery across these 32 route-matched controls.

Table 3: Held-out multi-layer steering, restricted to windows ending no later than layer 27.

Concept and layers	First → second	Second → first	Mean gap	Loss rec.	Ctrl. p95
$\alpha{>}0 \leftrightarrow \alpha{<}0$, L16–20	0.412	0.397	**0.404**	0.391	0.137
$\alpha{>}0 \leftrightarrow \alpha{<}0$, L20–24	0.411	0.387	0.399	0.423	0.151
$\alpha{>}0 \leftrightarrow \alpha{<}0$, L14–18	0.414	0.366	0.390	0.407	0.117
Additive ↔ multiplicative, L4–8	0.670	0.240	**0.455**	0.401	0.176
Additive ↔ multiplicative, L20–24	0.670	0.238	0.454	0.334	0.231
Additive ↔ multiplicative, L17–19	0.605	0.229	0.417	0.335	0.232

For the α-sign rows, first and second denote $\alpha > 0$ and $\alpha < 0$; for the law rows, they denote additive and multiplicative targets. Add/multiply steering is asymmetric: additive-to-multiplicative recovery is 0.605–0.670, whereas multiplicative-to-additive recovery is 0.229–0.240. The corresponding directional control p95 values are 0.219, 0.154, and 0.188 for L4–8, L20–24, and L17–19, respectively. Thus all three reverse effects exceed their route-matched directional controls, although they remain substantially weaker than the forward effects. The coefficient-sign result of 0.404 means the intervened prediction moves 40.4% of the paired donor–recipient output gap. It exceeds the route-matched random-control p95 of 0.137 and the roughly 0.10 single-layer recoveries in Figure 4b. Related intervention studies also interpret controlled partial effects as evidence about model mechanisms: Venhoff et al. (2025) recover 35% of a factual-recall accuracy gap versus 16% with random patching, and Rimsky et al. (2024) raise held-out answer-choice probability from 0.30 to 0.75 with

one contrastive direction. Their outcome measures differ from our normalized output-gap recovery, so they serve as interpretive precedents.

A.5 Component patching

Similar to steering, patching can validate a hypothesis about computation inside a model. Activation patching replaces an internal activation in one run with the corresponding activation from another run and measures the resulting output change (Heimersheim & Nanda, 2024). It is commonly used to discover circuits (Conmy et al., 2023). Here, component patching means the matched-run replacement of one attention-head or FFN output and measurement of its effect on the prediction. Component-level interventions require identifiable attention-head or FFN outputs that can be exchanged between matched runs. Interpretability work on TabPFN-2.5 performed individual-head patches before the attention output projection and found task- and layer-dependent causal effects, demonstrating component patching (Gupta et al., 2026a). For TabICLv2, the published patches replace complete row or block states, leaving the finer-grained experiment unreported (Biloš et al., 2026). In the ConTextTab analysis, post-LayerNorm placement prevents clean cross-run head-output patching at this intervention boundary (Aßmann, 2026).

QuTab's attention-head and FFN contributions can be isolated before they write to the query-label residual stream.

Localization experiments Before scanning individual components, we locate prediction formation using the complete query-label state. We apply a logit-lens readout, which passes every intermediate query-label state through the checkpoint's final RMSNorm and complete nonlinear regression head. This adapts the standard logit-lens early-exit principle to a nonlinear regression head (Belrose et al., 2023; Sandmann et al., 2025). Full-state activation patching replaces the entire query residual at one site with the matched donor state. The law-patch pairs share identical features and differ only in whether the targets follow the additive or multiplicative law described above. The coefficient-patch pairs likewise share features but use orthogonal unit-norm linear coefficient vectors; the reported value is bidirectional output-gap recovery after transplanting the complete query-label state between those matched runs.

Table 4 reports every post-block value from L20 through L30; earlier layers remain near zero. The two measurements agree that the task-dependent output begins forming around L21 and becomes strongly causal before the final output-writing blocks.

Table 4: Causal emergence through depth. Patch entries are bidirectional output-gap recovery; lens is the additive–multiplicative prediction-gap alignment with the final decoder output.

Layer	Law patch	Coefficient patch	Logit-lens alignment
20	0.077	0.014	0.095
21	0.295	0.029	0.287
22	0.339	0.029	0.295
23	0.447	0.054	0.363
24	0.438	0.075	0.368
25	0.584	0.154	0.480
26	0.594	0.159	0.523
27	0.688	0.426	0.640
28	0.866	0.726	0.816
29	0.894	0.733	0.883
30	1.000	1.000	1.000

At L27, law-contrast full-state patching recovers 0.688 of the donor–recipient output gap, versus 0.077 at L20; the logit lens using the frozen final decoder shows a parallel rise in alignment with the final prediction. QuTab's scan over all 480 heads and 30 FFNs demonstrates component-level granularity. Appendix A.6 explains how these effects are subsequently tested and used for circuit discovery.

Prior full-state patching also treats partial restoration as causal evidence. In ROME, one restored token state has an 8.7-point indirect effect against an 18.6-point total effect (47% as a ratio of averages), described as strongly causal (Meng et al., 2022). Kumaran et al. (2026a) report 24.3% confidence recovery from one full-state patch versus -1.4% at a control and explain why partial recovery is expected. Their outcomes differ from our signed output-gap measure.

A.6 SPARSE CIRCUIT

A central approach in mechanistic interpretability is to identify compact circuits that causally contribute to a specified behavior, making that behavior more tractable to study than the full network (Wang et al., 2023; Conmy et al., 2023). For the table's sparse-circuit criterion, we require a small selected component set that transfers a specified behavior when patched and remains sufficient when its complement is resampled.

In TabPFNv2, an L9 attention-weighted vote combines context-label evidence across all six heads, with no single head dominating per-head ablation; a selected small subset has not been tested for joint transfer and sufficiency (Biloš et al., 2026). TabICLv2 has a characterized class-prototype readout, but no selected component subset has been validated by both interventions (Biloš et al., 2026). ConTextTab's boundary-direction signal involves all 12 layer-0 row-attention heads under within-run ablation; selected-subset transfer and complement sufficiency have not been demonstrated (Aßmann, 2026). QuTab's separately identifiable head and FFN outputs write to one query-label residual stream, providing intervention sites for both tests.

Counterfactual transfer patches only the selected circuit from the donor into the recipient. Complement resample ablation patches all other heads and FFNs from the donor into the recipient, leaving the circuit unpatched. For the resulting prediction $f_{\bar{C}}$, circuit sufficiency is

$$\mathrm{Suff}_C = 1 - \frac{\sum_j (f_{\bar{C},j} - f_{d,j})^2}{\sum_j (f_{s,j} - f_{d,j})^2}, \tag{14}$$

averaged equally over the two patch directions. It equals one when the recipient prediction is preserved despite resampling every external component from the donor and zero when the deviation from the recipient is as large as the original donor–recipient gap. Transfer and sufficiency test different claims: a circuit may overwrite an unrelated recipient toward the donor output while still depending on external computation to infer the right answer for that recipient. To measure whether the patched output actually reproduces the donor, we also report normalized donor-output fidelity,

$$\mathrm{Fid}_C = 1 - \frac{\sum_j (f_{p,j} - f_{s,j})^2}{\sum_j (f_{s,j} - f_{d,j})^2}, \tag{15}$$

averaged equally over patch directions. A value of one means exact donor-output reproduction; unlike signed output-gap recovery, this score penalizes errors orthogonal to the donor–recipient gap.

Circuit refinement and faithfulness. Candidate components are selected in four stages. First, marginal counterfactual patches scan all 480 heads and 30 FFNs on 512 discovery pairs. Second, on 128 pairs, we replace each component individually while leaving all others unchanged and measure its squared output change normalized by the squared donor–recipient output gap. Components are ranked by the larger absolute value of their marginal output-gap effect and this single-component disruption score; effects below 0.001 are discarded and at most 128 candidates are admitted. Third, every ranked prefix and its full-model complement are evaluated on 512 validation pairs, selecting the smallest prefix whose transfer and normalized sufficiency both reach 0.85. This yields 17 components. A follow-up removes the negligible L30 FFN and compares the resulting 16-head candidate, the 14-head subset excluding two counteracting heads, those two heads separately, and all leave-one-out variants of the subset on three 512-pair seeds held out from component discovery. These comparisons select the 14-head circuit reported in Table 5. Bidirectional noising/denoising patches only the named components from the paired donor run. Complement resample ablation patches every other head and FFN, leaving the named circuit unpatched in the recipient run to test sufficiency. This procedure selects a sparse circuit under the stated threshold.

The values in Table 5 pool all 1,536 refinement pairs; they were held out from initial discovery but were used to compare variants and select the 14-head circuit. Across the three 512-pair seeds,

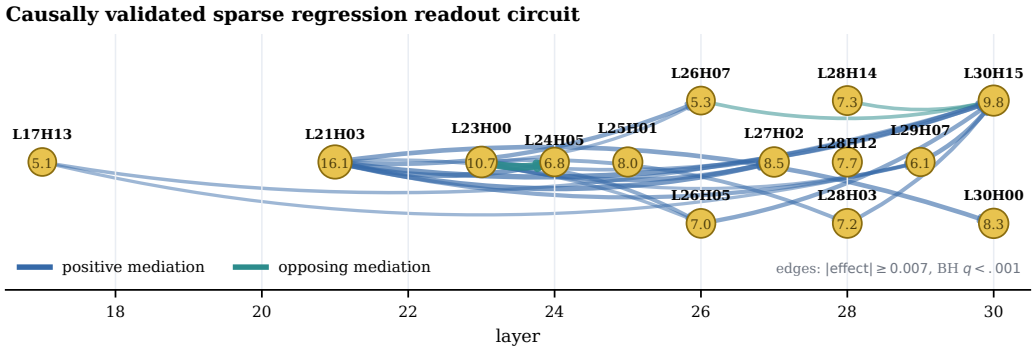


Figure 7: **Causally validated sparse regression readout circuit.** The largest validated mediation effects are shown; node labels give standalone gap effects in percentage points.

Table 5: Circuit refinement on 1,536 matched table pairs. Joint is the minimum of counterfactual transfer and normalized sufficiency under complement resample ablation.

Components	Size	Transfer	Norm. suff.	Joint
14-head circuit	14	**0.949**	**0.916**	**0.916**
16-head candidate	16	0.903	0.834	0.834
Circuit + L24H03	15	0.935	0.882	0.882
Circuit + L26H00	15	0.928	0.875	0.875
Counteracting pair	2	-0.191	-0.096	-0.191
Circuit without L29H07	13	0.936	0.907	0.907
Circuit without L26H05	13	0.913	0.904	0.904
Circuit without L24H05	13	0.919	0.904	0.904

transfer/normalized-sufficiency is $0.952/0.920$, $0.946/0.902$, and $0.950/0.927$. A paired bootstrap over the pooled refinement examples, conditional on the selected circuit, gives 95% intervals $[0.944, 0.955]$ for transfer and $[0.908, 0.924]$ for normalized sufficiency. Table 6 instead evaluates the frozen circuit on new pairs.

The two counteracting heads reduce the overshoot rate from 32.2% to 12.4% in the multiply-to-add direction and from 24.2% to 11.3% in the reverse direction. Nevertheless, they improve source fidelity on only 21.9% and 16.3% of examples and increase both source and ground-truth MSE. They may temper output overshoot in the intact model. The 13-head leave-one-out results show that the current set is not minimal under the stated thresholds.

Cross-law specificity and boundary controls. After freezing the 14 head coordinates, we test six new 512-pair families. Counterfactual transfer asks whether transplanting only the circuit writes the donor output; circuit sufficiency asks whether the recipient output is preserved when every external component is resampled from the matched counterfactual run. Table 6 shows that transfer remains high even when the circuit overwrites the output for an unrelated table, whereas the sufficiency test identifies which computation is actually contained in the circuit. The circuit is nearly sufficient when features and query are held fixed and only the target law changes, but depends on external computation to represent a new query or unrelated table. The label-shuffle result and label-carrier-only patching find that the circuit transports context-derived target evidence. This transfer/sufficiency asymmetry suggests that upstream components infer which answer is appropriate for the present query, whereas the sparse circuit retrieves and writes the corresponding label-side evidence.

For the frozen-circuit additive–multiplicative test, paired-bootstrap 95% intervals are $[0.950, 0.967]$ for transfer, $[0.988, 0.992]$ for donor-output fidelity, and $[0.892, 0.924]$ for sufficiency. Ground-truth loss recovery is 0.990; donor and patched target MSEs are $0.364/0.371$ and $0.045/0.063$ in the two directions. Together, these frozen-coordinate tests show that QuTab's selected 14-head set fulfills the sparse-circuit criterion for this readout behavior.

Table 6: Frozen-circuit generalization on 512 new pairs per family. Fidelity is normalized donor-output fidelity; label-carrier entries patch only the label-code third of the 192-dimensional structural carrier.

Counterfactual change	Transfer	Fidelity	Sufficiency	Label carrier
Additive ↔ multiplicative	0.959	0.990	0.909	0.926
Orthogonal linear rules	0.948	0.990	0.945	0.912
Product ↔ division	0.955	0.989	0.884	0.921
Clean labels ↔ shuffled labels	0.905	0.961	0.768	0.870
Unrelated multiplicative tables	0.928	0.980	0.396	0.928
Same context, changed query	0.933	0.979	−0.015	0.936

A.6.1 Mechanistic Decomposition

To understand how the selected heads implement the readout, we examine what each head reads, which pathway writes the result, and how its effect reaches later heads. QuTab's separate head outputs and shared query-label residual stream permit these interventions at consistent sites across depth. This follows the protocol used in LLM circuit work: choose a behavior and causal metric, localize components, vary the dataset and intervention unit to infer function, label nodes from concrete activating variables and output effects, and validate proposed edges by intervention (Conmy et al., 2023; Ameisen et al., 2025).

For each retained head, matched activation patching separately replaces its attention pattern (QK), retrieved values and output map (value/OV), 128-dimensional main value, 192-dimensional structural carrier, the carrier's row/column/label thirds, and context/query/history branches. Independent head probes then decode coefficients, input values, intermediates, final targets, and model predictions from each view. Attention statistics are aggregated over 256 matched pairs, and directed sender-to-receiver path patching uses another 512 pairs with Benjamini–Hochberg correction (Benjamini & Hochberg, 1995). Our known synthetic laws supply exact latent variables for interpreting the head probes, while the circuit-specificity suite tests whether the explanation extends beyond the discovery contrast.

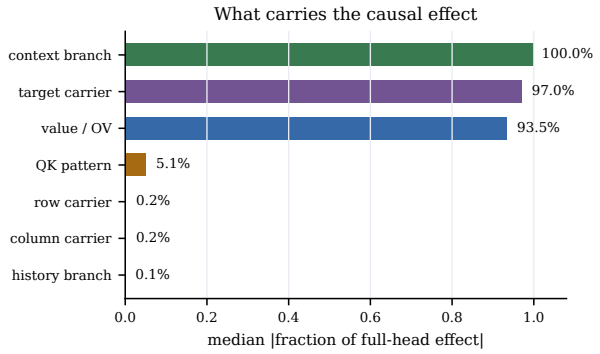


Figure 8: **What carries the causal effect.** Across heads, counterfactual law transfer is almost entirely context-derived value/OV written through the label carrier. Fractions can slightly exceed 100% because independently patched subpaths can cancel.

Although 107 of 112 tested directed edges in the 16-head candidate are significant, most effects are below 0.01 and the largest edge within the selected 14-head circuit is only 0.022. The observations thus mostly support parallel context readers with limited interactions: L21H03 is the strongest upstream node, while L30H00/H15 are late writers.

Independent support-row localization. We additionally scan all 480 heads over 512 five-column, 20-context-row tables whose query target is an inverse-distance weighted average of labels on one to three nearest rows, with distance computed on one to three declared active columns. Every one of the ten heads with greatest attention enrichment on planted support rows belongs to the

Table 7: Identified roles within the sparse readout circuit. "Rows" is $1/\sum_r \pi_r^2$, where π_r is the head's attention mass on context row r, renormalized over context cells.

Component	Gap effect	Rows	Supported role
L17H13	0.051	39.2	Early broad context-target summary; 98.7% of its effect is in the label carrier.
L21H03	0.161	11.6	Dominant selective reader; decodes product ($R^2 = 0.405$), quotient (0.383), product answer (0.837), and division answer (0.862); sends the largest positive paths to later writers.
L23H00	0.107	12.3	Second selective reader; routes target evidence toward L24H05, L27H02, and later writers.
L24H05	0.068	5.9	Intermediate context-label/OV relay; receives opposing paths from L23H00 and L21H03 and carries product/division-answer signal ($R^2 = 0.778/0.794$).
L25H01	0.080	6.0	Intermediate context-label/OV relay; carries product and quotient intermediates ($R^2 = 0.366/0.373$) and sends positive paths to later writers.
L26H05	0.070	5.1	Selective context-label writer; its carrier target code decodes the final prediction at $R^2 = 0.817$ and sends positive paths to both L30 writers.
L26H07	0.053	2.9	Sharp row/label relay; receives positive paths from L21H03, L23H00, and L25H01 and routes them toward L27H02 and the final writers.
L27H02	0.085	2.0	Sharp row/label retriever; on the planted support diagnostic, it places $10.62\times$ the uniform-cell baseline on designated support-row/active-column cells and selects a support row on 52.0% of tables.
L28H03	0.072	5.6	Selective context-label writer; its carrier target code decodes the final prediction at $R^2 = 0.826$ and sends its strongest positive path to L30H15.
L28H12	0.077	1.9	Sharp row/label retriever; strongest support-row alignment among all 480 heads ($6.67\times$ the uniform-row baseline; 58.4% top-row hit), while active-column mass remains uniform.
L28H14	0.073	1.7	Near-single-cell writer: 82.6% of conditional context attention occupies one cell on ordinary GraphSCM tables; 92–100% of its causal effect is label-carrier dominated.
L29H07	0.061	89.9	Broad recalibration candidate and the main QK exception (QK $= 0.019$ versus full $= 0.061$).
L30H00/H15	0.083/0.098	2.5/2.0	Sharp final context-target writers; each places over 90% of its effect in value/OV and the label carrier.

independently selected 14-head circuit; seven of the ten with greatest enrichment on task-relevant cells (the intersections of planted support rows and active columns) do as well. We report attention mass divided by the uniform mass implied by the subset size, so $1\times$ denotes uniform attention. L28H12 assigns $6.67\times$ this baseline to designated rows, while L27H02 assigns $10.62\times$ baseline mass to cells lying in both designated rows and active columns. Their active-column mass alone is $0.97\times/1.01\times$ baseline, supporting a row/label-retrieval role. The checkpoint solves this shifted KNN-style family only moderately (median absolute error 0.401; 35.5% within 0.25), so we treat the ranking only as supplementary directional evidence.

A.6.2 Attention Patterns and Learned Controllers

Standard transformer head views display one input at a time and encode attention magnitude locally (Vig, 2019); we support corpus-level conclusions with aggregate head metrics and deliberately selected exemplars. Prior TabPFN attention analyses average over heads and rows and remain

observational (Ye et al., 2025); here global head selection is separated from individual-table display and from the causal QK/value decomposition. We screen all 480 heads over 512 independent GraphSCM tables with 20 context rows, one query row, and five columns for distinct patterns, then randomly draw representative tables by predeclared score quantiles. L23H08 assigns 100.0% mean mass to its own query row; L19H04 assigns 90.7% to history; and L01H04 assigns 66.5% to the sink. Among context readers, L28H14 places 82.6% on one cell (1.69 effective rows, 1.01 columns). L26H00 is the closest row-spanning head but still covers only 2.75 effective columns within its top row. L20H06 instead places 85.7% of context attention in its top column globally (1.42 effective columns); the displayed example puts 98.0% in one column while spanning 8.77 effective rows. L28H05 reads broadly across 15.8 effective rows and 4.33 columns.

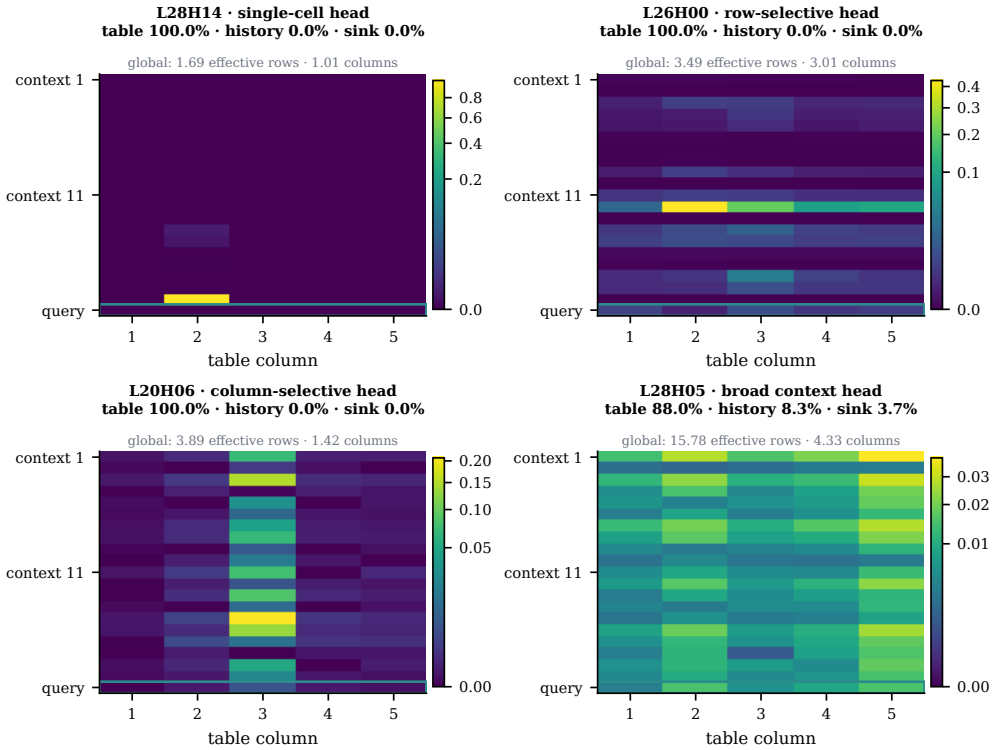


Figure 9: **Distinct context-attention motifs selected by an exhaustive global scan.** From upper left: a single-cell head, row-selective head, column-selective head, and broad context head. The cyan outline marks the query row.

The overview in Figure 12 depicts the regression checkpoint's learned global control parameters.

Structural binding and carrier use. Figure 13 profiles all 480 heads over 512 held-out GraphSCM regression tables. Columns separate row, column, and label identity; rows separate bound and explicit-carrier pathways for QK and OV. QK use is the mean absolute raw additive contribution to the pre-softmax score. OV use is the RMS residual update after native attention weighting, the trained output projection, and the learned output gate.

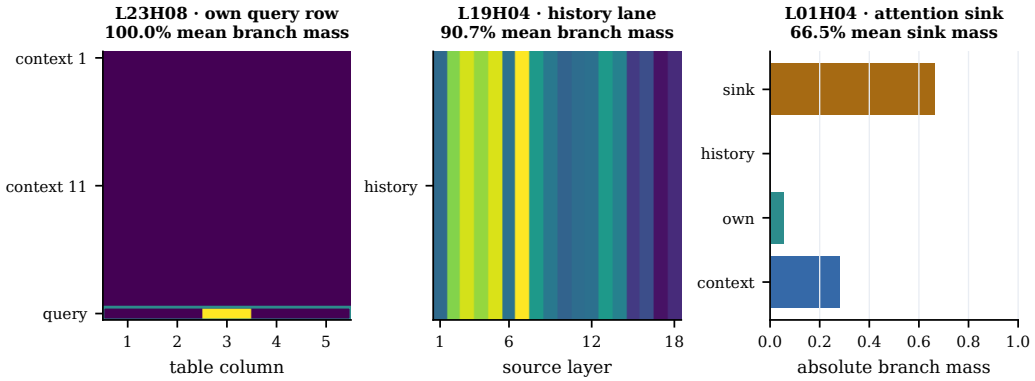


Figure 10: **Distinct non-context attention motifs.** L23H08 places all mass in its own query row, L19H04 reads the history lane, and L01H04 uses the learned sink.

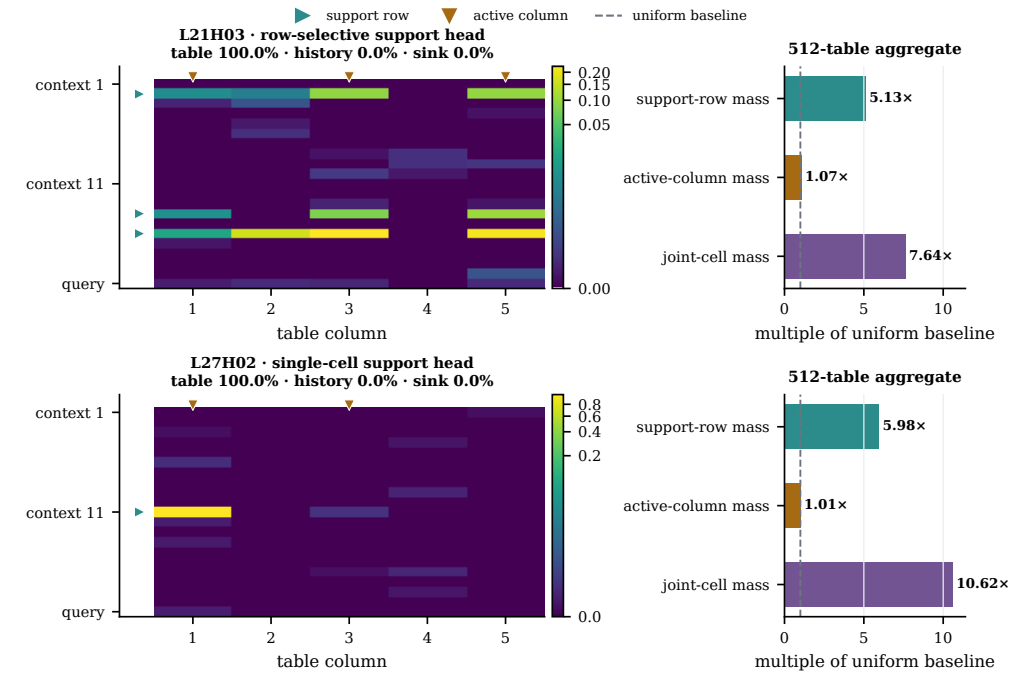


Figure 11: **Circuit heads retrieve planted support rows.** Left, locally scaled single-table captures: cyan edge markers designate support rows and amber edge markers designate active columns. L21H03 distributes attention across one selected support row, whereas L27H02 concentrates it in one support-row/active-column cell. Right, attention mass is divided by the uniform mass implied by each subset; 1× (dashed) is uniform attention.

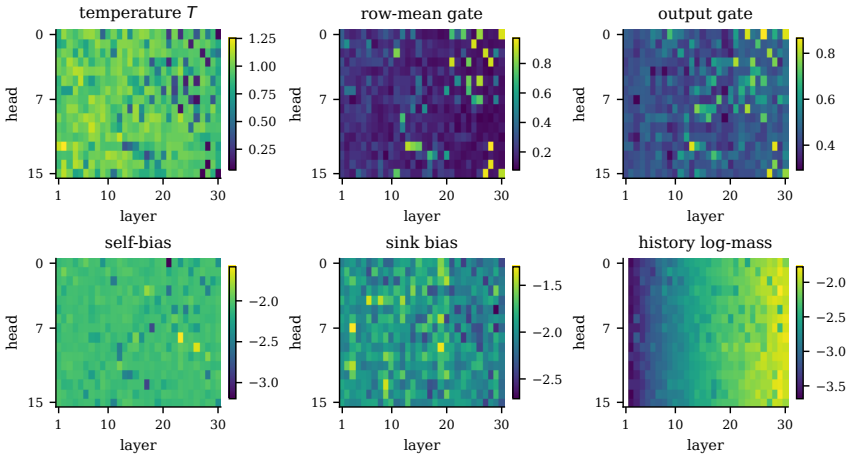


Figure 12: **Learned attention controllers over all 30 layers and 16 heads.** The first panel plots softmax temperature $T_{\ell,h} = 1/\tau_{\ell,h}$, so higher values mean softer attention. Late-layer temperature and row-mean-key specialization are visible. The model also increasingly allocates attention mass to history over depth.

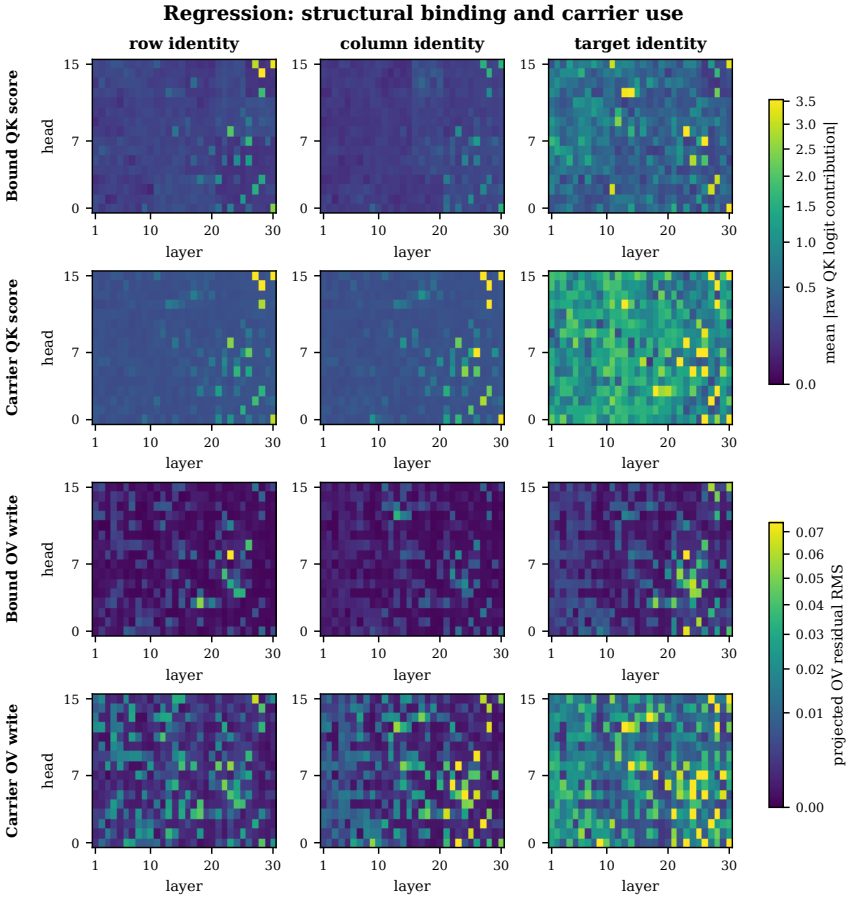


Figure 13: **Regression uses both bound and explicit structural pathways.** Columns show row, column, and label identity. Rows show bound-field QK scores, auxiliary carrier-routing QK scores, bound-value OV writes, and explicit-carrier OV writes. Each cell averages one query over 512 held-out GraphSCM tables.

A.7 CLASSIFICATION CHECKPOINT

We apply global probing, steering, full-state activation patching, and attention-pattern analysis to the frozen classification checkpoint. The scalar output for causal experiments is the model's class-1 probability.

A.7.1 GLOBAL PROBING

We follow the global-probe protocol in Appendix A.3, including its 8,192 independently sampled tables per family, 6,144/1,024/1,024 split, activation sites, probe optimization, validation selection, and shuffled-target controls. Here the binary labels are the signs of $ab + c$ and $a/b' + c$, where $b' = \mathrm{sign}(b)(|b| + 0.5)$; native classification accuracies are 95.3% and 94.6%, respectively. Table 8 reports the held-out results.

Table 8: Global probe results for classification. Sites are selected on validation tables; R^2 and the shuffled-target control use held-out tables. Native accuracy is on the complete corresponding probe family.

Quantity	Selected site	Test R^2	Shuffled R^2	Native acc.
Product ab	L29 post-FFN	.715	$-.349$	.953
Quotient a/b'	L30 post-FFN	.603	$-.399$	.946

A.7.2 STEERING

For the primary contrast, paired tables share the same two feature columns and structural slots. The class laws are the signs of $(a + b)/\sqrt{2}$ and $(-a + b)/\sqrt{2}$; a query row whose label differs between the two laws is placed at the evaluated position.

We follow the direction-estimation, dose-normalization, validation/test split, and route-matched control protocol in Appendix A.4. The outcome here is bidirectional recovery of the class-1 probability gap between the two intact runs. At the validation-selected L20–24 window and dose 1.0, the two directions recover 48.1% and 47.1% of this gap, averaging 47.6%; the random-control 95th percentile is 7.9%. Bidirectional Brier-loss recovery is 45.9%.

As a second contrast, three active features among five determine either a sum or product law. L20–24 steering gives 82.8% recovery in one direction and 27.0% in the other (54.9% mean).

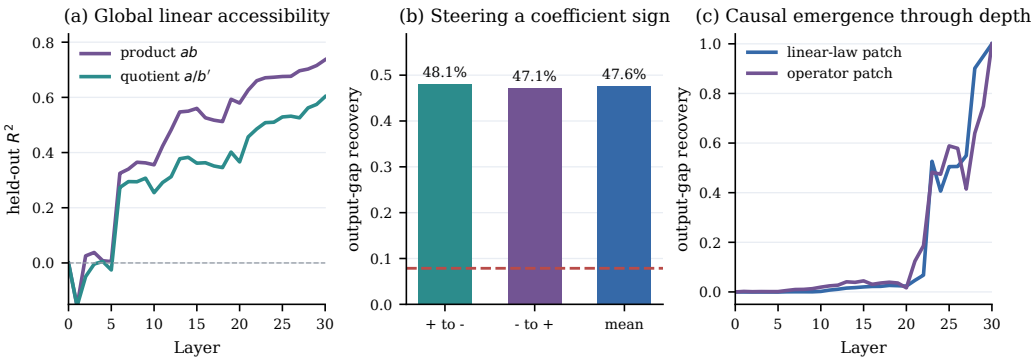


Figure 14: **Classification probing, steering, and activation patching.** (a) Global linear probes recover product and quotient intermediates across independent tables. (b) Contrastive steering of a linear-law coefficient changes class-1 probability on held-out tables; the dashed line marks the 95th percentile of random controls. (c) Full-query-state patching locates prediction formation through depth.

A.7.3 FULL-STATE ACTIVATION PATCHING

We locate where a full query-label-state replacement can alter the prediction. For counterfactual tables with either orthogonal unit-norm linear class laws or additive versus multiplicative class laws, we use the full-state patching protocol of Appendix A.5. Signed output-gap recovery is computed as in Appendix A.4, using class-1 probabilities instead of regression predictions. For orthogonal linear laws, post-block transfer rises from 2.3% at L20 to 54.9% at L27; for the operator contrast it rises from 1.7% to 41.5%.

A.7.4 ATTENTION PATTERNS AND LEARNED CONTROLLERS

We repeat the regression all-head scan (Appendix A.6.2) on 512 independent five-feature GraphSCM classification tables, reducing each to 20 context rows and one query row with preprocessing refitted. Heads are selected by global statistics, and each displayed map is one of four predetermined table exemplars; Figures 15 and 16 show context and non-context motifs, while Figures 17 and 18 show learned controllers and structural-pathway use.

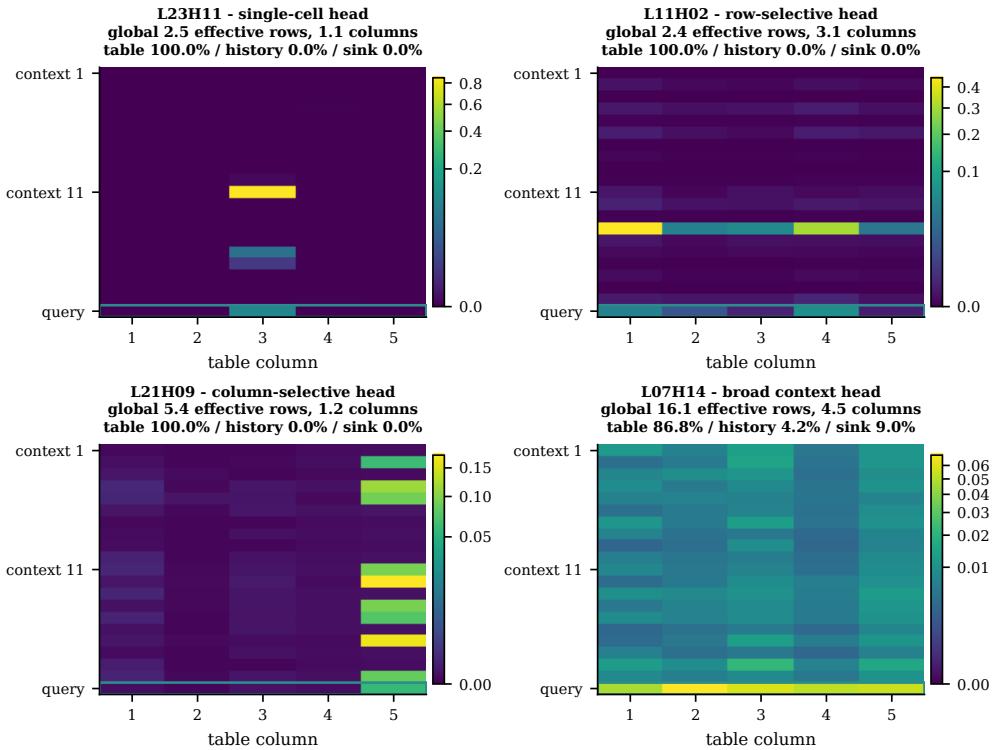


Figure 15: **Distinct context-attention motifs in classification.** From upper left: a single-cell head, row-selective head, column-selective head, and broad context head. The cyan outline marks the query row; each panel uses its own attention color scale.

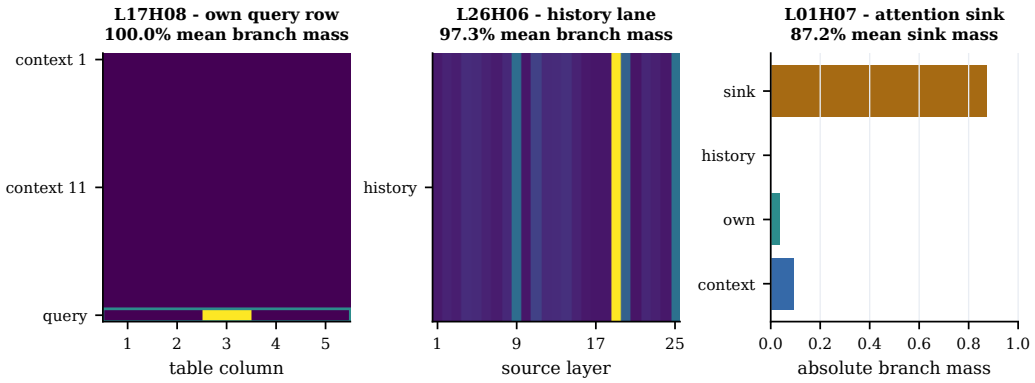


Figure 16: **Distinct non-context attention motifs in classification.** L17H08 places all mass in its own query row, L26H06 reads the history lane (97.3%), and L01H07 uses the learned sink (87.2%). These branch allocations are observational.

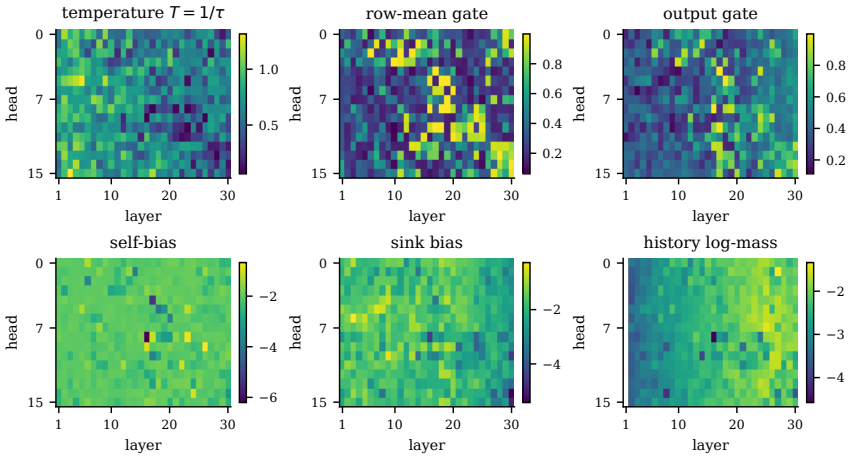


Figure 17: **Learned attention controllers in classification.** The layout and temperature convention match Figure 12.

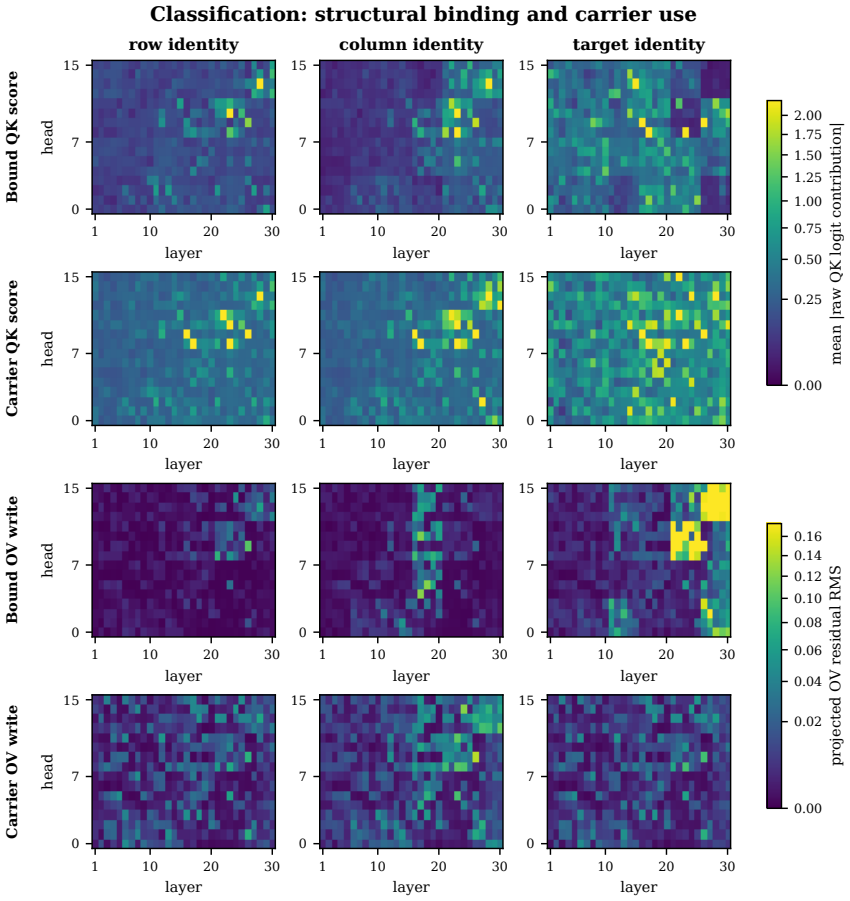


Figure 18: **Classification uses both bound and explicit structural pathways.** The measurement and layout match Figure 13; each cell averages one query over 512 held-out GraphSCM classification tables.

B Architecture Details

The classification and regression checkpoints use learned scalar Fourier inputs, 256-dimensional read-only cell representations, and a 2,048-dimensional query-label residual stream. Six stage memories are each reused by five blocks, and every block contains one DTA module followed by one 5,632-wide SwiGLU.

Table 9: Active 2B QuTab preset.

Query width d_q	2,048	Query heads H	16
Cell width d_c	256	Table KV groups G	4
Head dimension d_h	128	History KV heads	16
Stages S	6	Blocks/stage	5
Total blocks L	30	Main fields	$1 \times 32 + 3 \times 32$
Cell SwiGLU	768	Query SwiGLU	5,632
Auxiliary lanes/head	4×64	Fourier frequencies/type	32
Maximum row slots	10,240	Maximum column slots	2,048
Training queries	64	Class outputs	10
Regression outputs	1	Task decoder/checkpoint	1
Classification parameters	1,984,968,618	Regression parameters	1,984,931,745

The main fields comprise one unrotated content field and three rotated structural fields.

Table 10: Parameter accounting for the implemented model.

Component	Classification	Regression
Scalar Fourier input maps	50,240	50,240
Cell encoder	590,080	590,080
Task prediction token	2,048	2,048
Six stage projectors	1,975,296	1,975,296
Thirty query blocks	1,730,458,560	1,730,458,560
Twenty-nine history writers	243,447,808	243,447,808
Temperature, gates, and history controls	8,864	8,864
Final norm and task decoder	8,435,722	8,398,849
Total	1,984,968,618	1,984,931,745

B.1 Input Encoding

Let x be a preprocessed scalar cell and f its feature column. A per-column categorical mask selects one of the separate learned banks of 32 numerical and categorical Fourier frequencies, denoted by ω^{num} and ω^{cat}. For cell type $a \in \{\mathrm{num}, \mathrm{cat}\}$, the 64-dimensional Fourier vector and the initial cell embedding are

$$\Phi_a(x) = [\sin(x\omega_1^a), \ldots, \sin(x\omega_{32}^a), \cos(x\omega_1^a), \ldots, \cos(x\omega_{32}^a)], \tag{16}$$

$$e(x, f) = \mathrm{LayerNorm}(W_a \Phi_a(x) + b_a + W_{\mathrm{col}} \chi_f). \tag{17}$$

Here W_a, b_a are type-specific learned projections, W_{col} is bias-free, χ_f is the fixed 64-dimensional column code defined below, and LayerNorm has learned gain and bias. A 768-wide, pre-RMSNorm SwiGLU refines each cell through a residual addition:

$$\mathrm{Enc}(e) = e + W_2[\mathrm{SiLU}(W_g \,\mathrm{RMSNorm}(e)) \odot W_v \,\mathrm{RMSNorm}(e)], \tag{18}$$

where W_g, W_v, and W_2 are bias-free. Applying the same encoder once to context and query feature cells gives the fixed tensors $\boldsymbol{X}_c \in \mathbb{R}^{N \times F \times d_c}$ and $\boldsymbol{X}_q \in \mathbb{R}^{Q \times F \times d_c}$, with $d_c = 256$. Context labels supply the label phases and carriers described below.

Each query-label state $\boldsymbol{q}_{i,0} \in \mathbb{R}^{d_q}$ starts from the learned [PRED] token, with $d_q = 2048$. After all 30 query blocks, a final RMSNorm and a $2{,}048 \to 4{,}096 \to o$ decoder with GELU produce the prediction, where $o = 10$ for classification and $o = 1$ for regression. The task decoder is separate for the two checkpoints.

B.2 Structural RoPE

B.2.1 Codebooks and Binding

Row, column, and classification-label phase codebooks are generated once with seed 260823. Table-local position and structural slot are distinct: a table supplies unique integer row and column slots, and those integers index the frozen phase/carrier codebooks. During training, every synthetic table draws deterministic per-table row and column permutations from the full 10,240- and 2,048-slot codebooks. The matched synthetic tables used for probing and causal experiments likewise draw deterministic per-table permutations, with paired counterfactual runs sharing exactly the same assignments. Evaluation and the inference wrapper use consecutive row and retained-column slots from the frozen codebooks. For the reported evaluation, classification labels are assigned a NumPy seed-0 permutation of class slots after sorting the original labels.

For each K/V group, the projected 128-dimensional cell key or value x has four 32-dimensional fields: $x = [x^{\mathrm{content}} \| x^{\mathrm{col}} \| x^{\mathrm{row}} \| x^{\mathrm{label}}]$. The content field is unchanged. A context cell (r, f) with label y_r is bound by

$$B_{r,f}^{\mathrm{ctx}}(x) = [x^{\mathrm{content}} \| \mathcal{R}_{\phi_f} x^{\mathrm{col}} \| \mathcal{R}_{\psi_r} x^{\mathrm{row}} \| \mathcal{R}_{\lambda(y_r)} x^{\mathrm{label}}], \tag{19}$$

where $\mathcal{R}$ applies a two-dimensional rotation to each adjacent pair. The self-feature binding $B^{\text{self}}_{i,f,?}$ uses the same column phase ϕ_f, a dedicated query-row phase, and a fixed unknown-label phase. Both keys and values receive these rotations. Keys are RMS-normalized per K/V group before binding; the projected values are not normalized. The phase codebooks supply the first 16 pairs of each 64-dimensional row, column, or label code to the corresponding rotary field.

Addressing and binding primitives. For structural axis $A \in \{\text{row}, \text{col}\}$ and identity ι, where $\iota = r$ denotes context row r when $A = \text{row}$ and $\iota = f$ denotes feature column f when $A = \text{col}$, write the normalized P-pair phase code as

$$\gamma^A_\iota = \frac{1}{\sqrt{P}}[\cos\theta^A_{\iota,1}, \sin\theta^A_{\iota,1}, \ldots, \cos\theta^A_{\iota,P}, \sin\theta^A_{\iota,P}]. \tag{20}$$

This unit-Euclidean-norm code is convenient for analyzing identity matching. The value carriers used by the model are $\rho_r = \sqrt{P}\,\gamma^{\text{row}}_r$ and $\chi_f = \sqrt{P}\,\gamma^{\text{col}}_f$; their score-side copies are normalized to unit RMS, giving $\widehat{\rho}_r = \sqrt{2P}\,\gamma^{\text{row}}_r$ and $\widehat{\chi}_f = \sqrt{2P}\,\gamma^{\text{col}}_f$. For nominal identities the phases are fixed independent draws on the full circle, so

$$\langle \gamma^A_\iota, \gamma^A_{\iota'} \rangle = \frac{1}{P}\sum_{t=1}^{P} \cos(\theta^A_{\iota,t} - \theta^A_{\iota',t}), \qquad \begin{cases} = 1, & \iota = \iota', \\ \mathbb{E} = 0, & \iota \neq \iota'. \end{cases} \tag{21}$$

The implementation uses 32 phase pairs for each row and column carrier and the first 16 pairs for each rotary field. Classification-label rotations use an independent phase codebook; classification-label value carriers are one-hot vectors, and their 64-dimensional score-side copies are scaled by eight to unit RMS. Equation 21 provides a soft equality test for the unit-norm analytical codes; the score-side dot products use the unit-RMS copies above. Integer adjacency carries no geometric meaning.

The value path retains the unnormalized carriers. For a context cell it uses $[B^{\text{ctx}}_{r,f}(v^c_{r,f})\|\rho_r\|\chi_f\|\eta_{y_r}]$; for a self-feature cell it uses $[B^{\text{self}}_{i,f,?}(v^q_{i,f})\|\rho_*\|\chi_f\|\eta_?]$. Here ρ_* and $\eta_?$ are fixed codes for the query row and unknown label. The score path uses unit-RMS copies $\widehat{\rho}$, $\widehat{\chi}$, and $\widehat{\eta}$ of these carriers, in addition to the bound main key. Thus the carrier values preserve their original scale while the explicit identity scores are normalized.

Rotary transformations satisfy the identity

$$\langle \mathcal{R}_\alpha u, \mathcal{R}_\beta v \rangle = \langle u, \mathcal{R}_{\beta-\alpha} v \rangle. \tag{22}$$

A learned query field aligned with $\mathcal{R}_{\theta_i} u$ can therefore select keys carrying the same identity. QuTab applies the rotations to cell keys and values; its query projections learn the corresponding directions without an explicit query rotation.

B.2.2 Regression-Label Geometry

For regression, let $z = (y - \mu_{\text{ctx}})/\sigma_{\text{ctx}}$ be the target standardized using the context-label mean and standard deviation. We cap z to $[-100, 100]$ for numerical safety and define $u = \text{clip}(z/4, -1, 1)$. The factor four is a design choice that assigns four context standard deviations to each side of the phase interval; clipping preserves target order up to ties at the endpoints. With log-spaced frequencies $\omega_t \in [\pi/16, 3\pi/4]$, the first 16 phase pairs $\omega_t u$ provide the 32-dimensional regression-label rotation in Equation 2.

The 64-dimensional label carrier is

$$\eta^{\text{reg}}(u, z) = [1, z, \cos(\omega_1 u), \sin(\omega_1 u), \ldots, \cos(\omega_{31} u), \sin(\omega_{31} u)]. \tag{23}$$

Its trigonometric part is

$$\varphi(u) = [\cos(\omega_1 u), \sin(\omega_1 u), \ldots, \cos(\omega_{31} u), \sin(\omega_{31} u)]. \tag{24}$$

Then

$$\langle \varphi(u), \varphi(u') \rangle = \sum_{t=1}^{31} \cos(\omega_t (u - u')), \tag{25}$$

a multi-scale stationary similarity over the bounded standardized coordinate. Phase features support local and global label similarity, and the first 16 frequencies bind the label slice of the learned main K/V payload.

B.3 Directed Tabular Attention

Stage memories. At the beginning of each stage, the model separately normalizes the context and self-feature cells and applies one shared $256 \to 2Gd_h$ projection with $G = 4$ K/V groups and head dimension $d_h = 128$. Each K/V group is shared by four of the 16 query heads, following grouped-query attention (Ainslie et al., 2023). The key projection is RMS-normalized per group before Structural RoPE, while the value projection is used without that normalization. A separate $256 \to G \times 64$ projection maps RMS-normalized row-mean cell states to the row-mean keys, followed by per-group RMS normalization of the projected keys:

$$\kappa_r = \mathrm{RMS}_{\mathrm{key}}\left(W_\mu \,\mathrm{RMS}_{\mathrm{src}}\left(\frac{1}{F}\sum_{f=1}^{F}(\boldsymbol{X}_c)_{r,f}\right)\right). \tag{26}$$

The same construction on $\boldsymbol{X}_q$ gives κ_i^q for query row i. The resulting cell K/V tensors and row-mean keys are fixed for all five blocks in that stage.

Query and value projections. A query projection maps $2{,}048 \to 16 \times 384$, producing one 128-dimensional main query and four 64-dimensional auxiliary queries per head. The concatenated main outputs and the 16×192 carrier outputs use separate bias-free projections to width 2,048, after which we apply a $2{,}048 \to 5{,}632 \to 2{,}048$ SwiGLU. Each of the first 29 blocks also has an independent history writer that produces one 128-dimensional key and value per head from its RMS-normalized block input. A learned depth embedding is added to the projected key before per-head RMS normalization. The attention, stage, history, and SwiGLU matrix projections are bias-free. Each writer contains $2d_q^2 + 3d_q = 8{,}394{,}752$ parameters: two matrices, a source-normalization gain, a depth embedding, and a key-normalization gain.

For a fixed query row i, the writer attached to layer m receives the input state $\boldsymbol{q}_{i,m-1}$. Writing $a_{i,m} = \mathrm{RMSNorm}_{d_q}(\boldsymbol{q}_{i,m-1})$, its per-head history key and value have dimension $d_h = 128$ and are

$$k^h_{i,m,h} = \mathrm{RMSNorm}_{d_h}(W^{h,K}_{m,h} a_{i,m} + e^h_{m,h}), \qquad v^h_{i,m,h} = W^{h,V}_{m,h} a_{i,m}. \tag{27}$$

Here $e^h_{m,h}$ is the learned depth embedding; the RMS normalizations include learned gains. At reader layer ℓ, the same 128-dimensional main query used for cell content scores reads the $m < \ell$ history keys with a $1/\sqrt{d_h}$ dot product. History values contribute only to the 128-dimensional main readout, not to the 192-dimensional structural carrier readout.

Attention controllers. Each layer and head has a learned log-temperature $\alpha_{\ell,h}$:

$$\tau_{\ell,h} = \exp(-\alpha_{\ell,h}), \qquad \alpha_{\ell,h} \in [-\log 16, \log 16]. \tag{28}$$

Post-update projection keeps $\tau_{\ell,h}$ within $[1/16, 16]$. For the raw projected query blocks q^{raw} and $u^{a,\mathrm{raw}}$, with $a \in \{\mathrm{row}, \mathrm{col}, \mathrm{label}, \mathrm{mean}\}$, the effective queries used by DTA are

$$q = \tau_{\ell,h}\big(\mathrm{RMS}_{d_h}(q^{\mathrm{raw}}) \odot w^q_{\ell,h}\big), \tag{29}$$

$$u^a = \frac{\tau_{\ell,h}}{\sqrt{3d_h}}\big(\mathrm{RMS}_{64}(u^{a,\mathrm{raw}}) \odot w^a_{\ell,h}\big). \tag{30}$$

Here RMS_d denotes gain-free RMS normalization over d coordinates, and $w^q_{\ell,h}$ and $w^a_{\ell,h}$ are learned per-coordinate gains. Score-side carrier copies are normalized to unit RMS. With independent unit-variance coordinates and unit gains, the main score has variance $\tau^2_{\ell,h}$, while each 64-dimensional auxiliary score has variance $64\tau^2_{\ell,h}/(3d_h) = \tau^2_{\ell,h}/6$ for $d_h = 128$. The row, column, and label scores together contribute $\tau^2_{\ell,h}/2$, half the main score variance under this approximation. The row-mean score adds $\sigma(g^{\mathrm{mean}}_{\ell,h})^2\tau^2_{\ell,h}/6$; learned gains and correlations can change these values.

Equations 4–6 use one augmented dot product to summarize these pathways. The exact score-side vectors represented by that notation are

$$\widehat{q}_{\mathrm{score}} = [q \,\|\, \sqrt{d_h}u^{\mathrm{row}} \,\|\, \sqrt{d_h}u^{\mathrm{col}} \,\|\, \sqrt{d_h}u^{\mathrm{label}} \,\|\, \sqrt{d_h}\sigma(g^{\mathrm{mean}}_{\ell,h})u^{\mathrm{mean}}], \tag{31}$$

$$\widehat{k}^{c,\mathrm{score}}_{r,f} = [B^{\mathrm{ctx}}_{r,f}(k^c_{r,f}) \,\|\, \widehat{\rho}_r \,\|\, \widehat{\chi}_f \,\|\, \widehat{\eta}_{y_r} \,\|\, \kappa_r]. \tag{32}$$

Their dot product divided by $\sqrt{d_h}$ reproduces the exact context score in Equation 35: the main field receives the outer normalization, the auxiliary fields retain their scaling from Equation 30, and the row-mean gate is applied once. The self-feature score uses its query-row and unknown-label score codes and the query-row mean key. The value path uses the unnormalized ρ, χ, and η carriers of Equation 3.

Score and value pathways. Split the learned main query and key into a content field and three bound fields. For context cell (r, f) with label y_r, one head computes

$$s^{\text{main}}_{r,f} = \frac{1}{\sqrt{d_h}}\big[\langle q^{\text{content}}, k^{\text{content}}_{r,f}\rangle + \langle q^{\text{col}}, \mathcal{R}_{\phi_f} k^{\text{col}}_{r,f}\rangle + \langle q^{\text{row}}, \mathcal{R}_{\psi_r} k^{\text{row}}_{r,f}\rangle + \langle q^{\text{label}}, \mathcal{R}_{\lambda(y_r)} k^{\text{label}}_{r,f}\rangle\big], \tag{33}$$

$$\begin{aligned} s^{\text{id}}_{r,f} &= s^{\text{col}}_f + s^{\text{row}}_r + s^{\text{label}}_r \\ &= \langle u^{\text{col}}, \widehat{\chi}_f\rangle + \langle u^{\text{row}}, \widehat{\rho}_r\rangle + \langle u^{\text{label}}, \widehat{\eta}_{y_r}\rangle, \end{aligned} \tag{34}$$

$$z^c_{r,f} = s^{\text{main}}_{r,f} + s^{\text{id}}_{r,f} + \sigma(g^{\text{mean}}_{\ell,h})\langle u^{\text{mean}}, \kappa_r\rangle. \tag{35}$$

The effective queries use the normalization and scaling of Equation 30. In particular, the main dot product alone is divided by $\sqrt{d_h}$; the separately scaled auxiliary dot products are added without another $\sqrt{d_h}$ divisor. The augmented dot product in Equations 5–6 is a compact representation of these pathways. Softmax exponentiation factorizes the score:

$$e^{z^c_{r,f}} = e^{s^{\text{main}}_{r,f}} e^{s^{\text{col}}_f} e^{s^{\text{row}}_r} e^{s^{\text{label}}_r} e^{\sigma(g^{\text{mean}}_{\ell,h}) s^{\text{mean}}_r}, \qquad s^{\text{mean}}_r = \langle u^{\text{mean}}, \kappa_r\rangle. \tag{36}$$

Each factor can sharpen a content, column, row, label, or row-mean-key condition. Their product implements conjunctions inside one cell-level softmax.

Let $p^c_{r,f}$ be the resulting context probability after the joint DTA softmax. The context contributions to the main value and three carrier values are

$$o^{\text{main},c} = \sum_{r,f} p^c_{r,f}\, B^{\text{ctx}}_{r,f}(v^c_{r,f}), \tag{37}$$

$$g^{\text{row},c} = \sum_{r,f} p^c_{r,f}\, \rho_r, \quad g^{\text{col},c} = \sum_{r,f} p^c_{r,f}\, \chi_f, \quad g^{\text{label},c} = \sum_{r,f} p^c_{r,f}\, \eta_{y_r}. \tag{38}$$

After adding the self-feature and history contributions, the block projects o^{main} and $[g^{\text{row}} \| g^{\text{col}} \| g^{\text{label}}]$ separately, as in Equation 48. The main value retains bound content. The carriers expose the structural marginals selected by the head.

Self-feature pathway. For a fixed query row i, the self-feature binding $B^{\text{self}}_{i,f,?}$ follows Equation 19 with the query-row and unknown-label phase codes. Its structural score is

$$s^{\text{str},q}_f = \langle u^{\text{row}}, \widehat{\rho}_*\rangle + \langle u^{\text{col}}, \widehat{\chi}_f\rangle + \langle u^{\text{label}}, \widehat{\eta}_?\rangle + \sigma(g^{\text{mean}}_{\ell,h})\langle u^{\text{mean}}, \kappa^q_i\rangle, \tag{39}$$

so its full logit is $z^q_f = \langle q, B^{\text{self}}_{i,f,?}(k^q_f)\rangle/\sqrt{d_h} + s^{\text{str},q}_f + b_{\text{self}} + \log N$. Here ρ_* is the dedicated query-row value code shared by all query rows, $\widehat{\rho}_*$ is its unit-RMS score copy, and κ^q_i is the corresponding key for the mean encoded self-features of query row i. Its augmented value is $\bar{v}^q_f = [B^{\text{self}}_{i,f,?}(v^q_f) \| \rho_* \| \chi_f \| \eta_?]$. Writing $P^q = \sum_f p^q_f$, the complete per-head carrier readout is

$$g^{\text{row}} = \sum_r \left(\sum_f p^c_{r,f}\right) \rho_r + P^q \rho_*, \tag{40}$$

$$g^{\text{col}} = \sum_f \left(\sum_r p^c_{r,f} + p^q_f\right) \chi_f, \tag{41}$$

$$g^{\text{label}} = \sum_r \left(\sum_f p^c_{r,f}\right) \eta_{y_r} + P^q \eta_?. \tag{42}$$

History and branch allocation. The self-feature bias b_{self} and sink bias b_{sink} in Equations 6 and 8 are learned scalars. Each history entry has bias

$$b^h_{\ell,h,m} = b^{\text{mass}}_{\ell,h} + b^{\text{lag}}_{\ell,h,m}, \qquad 1 \le m < \ell, \tag{43}$$

with one mass bias per reader head and a separate bias for each available lag. For a fixed query row and reader layer ℓ, let $q_{i,\ell,h}$ be the effective main query of head h from Equation 30. Then $M = |\mathcal{H}_{i,<\ell}| = \ell - 1$ and the exact history logit is

$$z^h_{i,\ell,h,m} = \langle q_{i,\ell,h}, k^h_{i,m,h}\rangle/\sqrt{d_h} + b^h_{\ell,h,m} + \log(NF/M), \qquad m < \ell. \tag{44}$$

The 128-dimensional main query and history key appear directly in this score; the 64-dimensional structural codes are not appended to history. The context, self-feature, history, and sink branches contain NF, F, M, and one score, respectively. The count corrections in Equations 5–8 give each available branch the same base multiplicity:

$$\underbrace{NF}_{\text{context}} = \underbrace{Fe^{\log N}}_{\text{self}} = \underbrace{Me^{\log(NF/M)}}_{\text{history}} = \underbrace{e^{\log(NF)}}_{\text{sink}}, \qquad M > 0. \tag{45}$$

The number of entries therefore cancels out of the branch's base weight; content scores and learned biases determine the attention allocation. The sink adds $b_{\text{sink}} + \text{softplus}(b_{\text{self}})$ to its count correction, coupling its allocation to the self-feature bias. For the fixed row, layer, and head, all available scores enter one softmax:

$$z^{\varnothing} = b_{\text{sink}} + \log(NF) + \text{softplus}(b_{\text{self}}), \qquad [p^c, p^q, p^h, p^{\varnothing}] = \text{softmax}([z^c, z^q, z^h, z^{\varnothing}]). \tag{46}$$

The sink has zero value. At layer 1, $M = 0$ and the history branch is absent.

Head output. Equation 10 summarizes the value readout with one output map. In the implementation, the same DTA probabilities form each head's 128-dimensional main readout from bound context values, self-feature values, and history values:

$$o^{\text{main}}_h = \sum_{r,f} p^c_{r,f} B^{\text{ctx}}_{r,f}(v^c_{r,f}) + \sum_f p^q_f B^{\text{self}}_{i,f,?}(v^q_{i,f}) + \sum_{m<\ell} p^h_m v^h_{i,m,h}. \tag{47}$$

Its 192-dimensional carrier readout is $c_h = [g^{\text{row}}_h \| g^{\text{col}}_h \| g^{\text{label}}_h]$ from context and self-feature mass. A sigmoid gate scales each head before separate bias-free main and carrier output projections:

$$\Delta \boldsymbol{q}_i = W^{\text{main}}_O \,\text{concat}_h\left[\sigma(g^{\text{out}}_{\ell,h}) o^{\text{main}}_h\right] + W^{\text{carrier}}_O \,\text{concat}_h\left[\sigma(g^{\text{out}}_{\ell,h}) c_h\right]. \tag{48}$$

History contributes to o^{main}_h; the sink has zero value. We initialize $g^{\text{out}}_{\ell,h} = 0$, giving gate value 0.5 within the learnable range $(0, 1)$.

B.4 Architectural Circuit Primitives

QuTab's structural pathways were chosen to support common tabular operations, including nearest-neighbor retrieval, label voting, threshold estimation, cross-column lookup, and regression from local or global evidence. Because the cell representations are frozen, these operations must be assembled by query-side circuits. The constructions below establish representational capabilities of the architecture. Appendix A.6.2 measures the learned use of the structural pathways.

B.4.1 Set reads need content-independent identity scores

Whole-set selection. A whole-row read requires a positive score shared by every cell in one row. The auxiliary row code supplies

$$a^{\text{row}}_{r,f} = \beta\langle \gamma^{\text{row}}_{r^*}, \gamma^{\text{row}}_r\rangle \approx \beta\, \mathbf{1}[r = r^*]. \tag{49}$$

This score is constant over f. A rotary row field rotates learned, cell-dependent keys and cannot guarantee a common sign. For $k_{r,f_2} = -k_{r,f_1}$,

$$\langle q, \mathcal{R}_{\theta_r} k_{r,f_2}\rangle = -\langle q, \mathcal{R}_{\theta_r} k_{r,f_1}\rangle. \tag{50}$$

The common rotation preserves the sign conflict. Equation 49 raises both cells. Column and label scores provide analogous set operations, while the content main field supplies learned scalar-content lookup across structural identities.

A direct cell lookup combines row and column scores. A column scan omits the row condition, and a row scan omits the column condition. Content scores can further restrict any of these sets by value. This shared construction supplies point reads, slices, and content-filtered slices.

Value–address association. Attention returns an additive superposition. An unbound two-cell read gives

$$o = s_1 v_1 + s_2 v_2. \tag{51}$$

This output does not identify which value came from which address. Separate carriers report address marginals and still omit the pairing. Opposite contributions such as -5 and $+5$ may also cancel.

The 32-dimensional column-bound value field stores value–column pairing inside the superposition. Reading one row gives

$$o^{\mathrm{col}}_{\mathrm{row}} = \sum_f s_f \mathcal{R}_{\phi_f} v^{\mathrm{col}}_f. \tag{52}$$

A later column-aligned probe has response

$$\left\langle \mathcal{R}_{\phi_{f^*}} u, o^{\mathrm{col}}_{\mathrm{row}} \right\rangle = s_{f^*} \langle u, v^{\mathrm{col}}_{f^*} \rangle + \sum_{f \neq f^*} s_f \left\langle u, \mathcal{R}_{\phi_f - \phi_{f^*}} v^{\mathrm{col}}_f \right\rangle. \tag{53}$$

The desired column is phase aligned. Under independent relative phases and fixed probe, weights, and payloads, the cross-column interference I has zero mean and variance

$$\mathrm{Var}(I) = \frac{1}{2} \sum_{f \neq f^*} s_f^2 \sum_{t=1}^{16} \|u_t\|^2 \|v^{\mathrm{col}}_{f,t}\|^2, \tag{54}$$

where t indexes two-dimensional rotary pairs. For comparable payload norms, evenly distributed pair energy, and an aligned desired payload, the signal-to-interference power ratio (SIR) is approximately

$$\mathrm{SIR} \approx \frac{32 s_{f^*}^2}{\sum_{f \neq f^*} s_f^2}. \tag{55}$$

Useful recovery therefore requires the selected column's squared weight to dominate the competing squared weights divided by 32. Uniform mixing over F columns gives approximately $32/(F-1)$, motivating selective reads or several heads that separate columns. Figure 9 reports learned attention concentration relevant to this condition; it does not measure this phase-model SIR. Column-bound values preserve column membership during a row read. Row-bound values provide the symmetric circuit for reading one column across several rows.

Label field. Each context feature cell carries the label code of its row. This construction replaces explicit context-label tokens. For classification,

$$g^{\mathrm{label}} = \sum_{r,f} p^c_{r,f}\, e_{y_r} \tag{56}$$

is the attention-weighted class histogram. The label-bound main value preserves the association between retrieved content and y_r. The carrier exposes the class coordinate directly to the residual stream.

B.4.2 Conditional cross-column reads

Row-mean-key route. The stage memory includes one content-derived key κ_r per row (Equation 26). The encoded context cells $(\boldsymbol{X}_c)_{r,f}$ already incorporate column identity before their nonlinear refinement. Their mean supplies a learned row summary. Matching this summary to a query-side direction gives every cell in the row a shared score; a column address can then select the values to retrieve:

$$z_{r,f} = \sigma(g^{\mathrm{mean}}_{\ell,h}) \langle u^{\mathrm{mean}}, \kappa_r \rangle + \beta \langle \gamma^{\mathrm{col}}_f, \gamma^{\mathrm{col}}_2 \rangle. \tag{57}$$

The row-summary match and column address act on the same cell logit. The retrieved column-2 values therefore come from rows selected by their learned summary. This shared row preference encourages coherent row reads.

Two-block route. A column-specific condition can be assembled explicitly by reading the conditioning column and writing the selected row identities. Block 1 can read high values from column 5 and return the matching row set:

$$g_1^{\mathrm{row}} = \sum_r \alpha_r \rho_r. \tag{58}$$

Block 2 uses a query direction aligned with g_1^{row} together with the column-2 score. For a concentrated selection, the normalized row-code match is dominated by the selected rows:

$$\left\langle \sum_{r'} \alpha_{r'} \gamma_{r'}^{\mathrm{row}}, \gamma_r^{\mathrm{row}} \right\rangle = \alpha_r + \sum_{r' \neq r} \alpha_{r'} \langle \gamma_{r'}^{\mathrm{row}}, \gamma_r^{\mathrm{row}} \rangle.$$

The second term is finite-code interference, so this route is most reliable for a small selected row set.

Example. Consider three context rows:

	column 5	column 2	label
r_1	8	-3	A
r_2	1	6	B
r_3	9	4	A

For the predicate $x_{r,5} > 7$, the two-block route first selects r_1 and r_3 through column 5, then concentrates on $(r_1, 2)$ and $(r_3, 2)$. The row-bound output contains

$$o^{\mathrm{row}} \approx s_1 \mathcal{R}_{\psi_{r_1}} v_{-3}^{\mathrm{row}} + s_3 \mathcal{R}_{\psi_{r_3}} v_4^{\mathrm{row}}, \tag{59}$$

together with row carrier $s_1 \rho_{r_1} + s_3 \rho_{r_3}$ and label carrier $(s_1 + s_3) e_A$. Row binding preserves the association of -3 with r_1 and 4 with r_3.

Self-feature cells share column codes with context cells. Their row and label codes are fixed to r_* and the unknown-label identity. Early blocks can read the observed query row and construct predicates for later context reads. Value binding preserves feature–column associations. The $\log N$ correction compensates for the $NF{:}F$ branch-cardinality ratio.

B.4.3 Prediction circuits

The following constructions show query-side computations supported by these pathways. Appendix A.6 reports the mechanisms observed in the trained model.

Nearest neighbors. After early blocks summarize the self-features, a row-mean-key head can assign conditional context-row weights

$$\alpha_r \propto \exp\big(\sigma(g_{\ell,h}^{\mathrm{mean}}) \langle u^{\mathrm{mean}}, \kappa_r \rangle\big). \tag{60}$$

With context attention mass near one, the class carrier $\sum_r \alpha_r e_{y_r}$ performs a soft neighbor vote in the learned row-summary space. The z coordinate of the regression carrier gives the local estimate $\widehat{z}_q \approx \sum_r \alpha_r z_r$. Different heads can compare different directions in these learned row summaries. Column-specific neighborhoods can use a column-addressed feature comparison followed by the row-carrier route above. Temperature controls the neighborhood width.

Majority voting. A diffuse head with context mass near one and approximately uniform context attention returns

$$g^{\mathrm{label}} \approx \frac{1}{NF} \sum_{r,f} e_{y_r} = \frac{1}{N} \sum_r e_{y_r}. \tag{61}$$

The output is the empirical class distribution. Structural and content scores turn this circuit into a conditional vote. The sink absorbs probability when the selected evidence does not justify an update.

Threshold rules. Label and column scores can restrict two heads to opposite classes within one feature column. Define the soft extrema

$$M_\tau(S) = \sum_{x \in S} \frac{e^{\tau x}}{\sum_{z \in S} e^{\tau z}}\, x, \qquad m_\tau(S) = M_{-\tau}(S). \tag{62}$$

For a clean binary split, the heads can estimate

$$t \approx \frac{1}{2}\left[M_\tau(\{x_{r,f} : y_r = 0\}) + m_\tau(\{x_{r,f} : y_r = 1\})\right]. \tag{63}$$

The query MLP compares $x_{q,f}$ with t. Further heads cover the reverse orientation and noisy class boundaries. The label carrier score supplies the class-conditioned sets.

Multivariate regression. For a small addressed feature subset, query-side blocks can implement sequential regression updates. At step t, several heads share one row address r_t and select its feature columns separately, while their label carriers supply the same row's z_{r_t}. The query MLP combines these retrieved features with a stored coefficient hypothesis, forms the signed residual, and updates

$$\beta^{(t+1)} = \beta^{(t)} + \eta_t x_{r_t}\left(z_{r_t} - x_{r_t}^{\mathsf{T}}\beta^{(t)}\right). \tag{64}$$

The multiplication occurs after the aligned row values are retrieved, so the update can have either sign despite nonnegative attention weights. Query-state storage scales with the addressed feature subset rather than the number of context rows; the available heads, width, and depth bound the subset size and number of updates. The same query-label state can retain the query features for the final linear prediction.

History and alternative computation paths. Layer ℓ can read an earlier query-label state through

$$o_\ell^{\mathrm{hist}} = \sum_{1 \le m < \ell} \pi_{\ell m}\, v_m^h, \tag{65}$$

Here v_m^h is written from the input $\boldsymbol{q}_{m-1}$ to layer m; depth embeddings and lag biases distinguish the stored states. A threshold, neighbor set, or regression hypothesis can be computed early and recovered after later blocks explore another route. The current residual does not have to preserve every intermediate quantity. Stage boundaries expose new K/V projections of the same read-only encoded cells. The sink and output gate control whether a completed head contributes to the residual.

C Training Details

C.1 Data and Objectives

We train classification on four A100 graphics processing units (GPUs) and regression on two. Stage 1 contains approximately 22M classification tables and 11M regression tables, generated online from the TabICLv2 GraphSCM prior. It comprises a cosine-decay phase followed by a stable learning-rate phase. Stage 2 continues each model for roughly 1M additional tables using the broader curriculum in Table 11, while retaining optimizer state and the stable learning rates. Both stages use $Q = 64$ query rows and a global update of 256 distinct tables: 64 per classification rank and 128 per regression rank. Groups of four tables share a sampled shape but have independently generated contents. We do not use paired permutation training.

Classification tables contain at most ten classes with labels indexed from zero. The classification head always produces ten logits, and query-level cross-entropy includes all ten even when a table contains fewer classes. With B tables in an update, Q_b query rows in table b, and $p_{b,j,c}$ the softmax probability of class c over those ten logits, the classification loss is

$$\mathcal{L}_{\mathrm{clf}} = -\frac{1}{\sum_{b=1}^{B} Q_b} \sum_{b=1}^{B} \sum_{j=1}^{Q_b} \log p_{b,j,y_{b,j}}. \tag{66}$$

For regression, let y be the raw target, μ_{ctx} the context-label mean, and $\sigma_{\mathrm{ctx}} = \max(\mathrm{std}(y_{\mathrm{ctx}}), 10^{-6})$ the context-label standard deviation with a numerical floor. The supervised target is $z = \mathrm{clip}((y -$

Table 11: **Two-stage training curriculum.** Row and column counts denote logical table shapes. The table budgets are approximate and are given as classification/regression. $U[a, b]$ denotes a discrete uniform distribution.

Stage	Context rows	Feature columns	Tables (cls./reg.)
1	50% fixed at 1,024; 50% $U[256, 1024]$	$U[1, 100]$	22M / 11M
2	50% fixed at 2,048; 50% $U[256, 10240]$	80% $U[1, 100]$; 20% $U[101, 512]$	∼1M / ∼1M

$\mu_{\text{ctx}})/\sigma_{\text{ctx}}, -100, 100)$; the $[-100, 100]$ cap prevents extreme standardized targets from causing numerical issues. With $t_{b,j}$ the raw decoder output and $z_{b,j}$ the standardized query target, we optimize the stabilized per-table RMSE

$$\mathcal{L}_{\text{reg}} = \frac{1}{B}\sum_{b=1}^{B}\sqrt{\frac{1}{Q_b}\sum_{j=1}^{Q_b}(t_{b,j} - z_{b,j})^2 + 10^{-6}}. \tag{67}$$

The added 10^{-6} stabilizes the derivative near zero error, and the outer average weights tables equally. At inference, the inference wrapper maps raw decoder output t_q back to the original target units as

$$\widehat{y}_q = \mu_{\text{ctx}} + \sigma_{\text{ctx}} t_q, \qquad \sigma_{\text{ctx}} \geq 10^{-6}. \tag{68}$$

Figures 19 and 20 show the complete recorded training-loss traces. The transition to larger tables changes the loss distribution, so the final portion is not directly comparable to the preceding small-table phase.

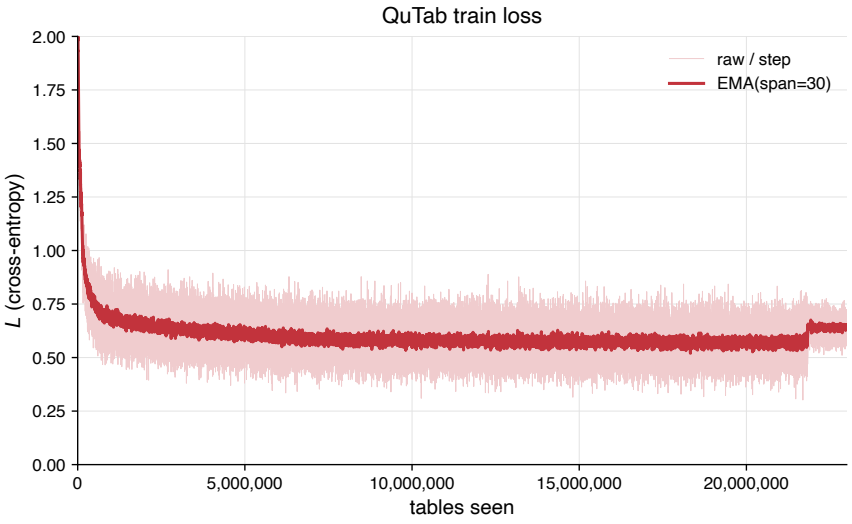


Figure 19: **Classification training loss.** Cross-entropy against tables seen. The pale line shows raw per-step loss and the dark line its exponential moving average with span 30; the final change in loss accompanies the broader training curriculum.

C.2 Optimizer and Distributed Update

The Muon group contains the cell-encoder matrices, stage K/V and row projections, query-block attention and output projections, query SwiGLU matrices, and history K/V projections. It uses Nesterov momentum 0.95, five Newton–Schulz steps, RMS-matched update scaling, and decoupled weight decay 0.1 (Jordan et al., 2024; Liu et al., 2025). Equal-shaped matrices are batched during the polynomial multiplication. All remaining parameters use fused AdamW (Loshchilov & Hutter,

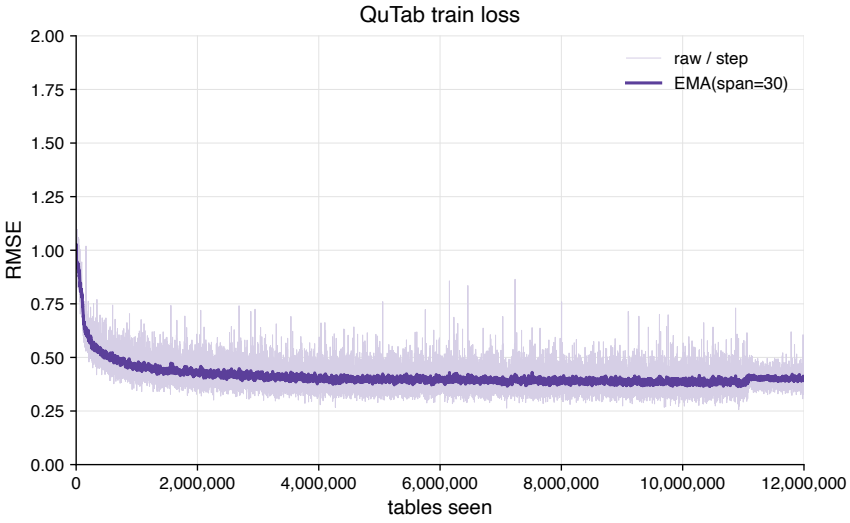


Figure 20: **Regression training loss.** RMSE against tables seen, with raw per-step loss in pale purple and its span-30 exponential moving average in dark purple. The final portion includes continuation on the broader curriculum.

2019). AdamW projection, decoder, and normalization-gain weights receive weight decay 0.1; Fourier-frequency vectors, biases, the prediction token, and structural controls receive no weight decay. The initial peak learning rates are 2×10^{-4} for Muon and non-structural AdamW groups and 10^{-3} for structural controls.

Stage 1 begins with linear learning-rate warmup, followed by smooth cosine decay to 2×10^{-5} for Muon and non-structural AdamW groups and 10^{-4} for structural controls. Its stable phase holds these rates. Stage 2 retains the optimizer momentum, Adam moments, FP32 master parameters, and per-group learning rates; it introduces neither a new warmup nor a further learning-rate reduction.

Appendix E.1 specifies how each logical update is physically packed. After all local packs complete, full gradients are averaged across ranks in fixed 32 MiB buckets and then subjected to one global norm clip at 10; this is the only clipping operation. The synchronization is a full all-reduce, not an owner-reduce, performed after the final backward. PyTorch DDP instead overlaps bucketed all-reduce with backward (Li et al., 2020). Stage 1 uses owner-sharded optimizer state and FP32 master parameters: each owner updates its synchronized parameters and broadcasts the updated BF16 values. Stage 2 imports the saved state into replicated optimizer buffers and applies the same update on every rank after gradient synchronization.

D Evaluation Details

D.1 Protocol and Metrics

We evaluate the official TabArena-v0.1 Lite `r0f0` split (one predefined outer split per dataset) from revision `de1be372095e23e72e2f42992eceea036b97dea6` (Erickson et al., 2025), using context-fitted preprocessing and deterministic seed 0. Support rows and feature subsets are sampled independently at each envelope; classification support retains every class. Row and retained-column slots are consecutive, while classification-label slots use a seed-0 permutation in original-label order. All test rows are evaluated with 64-query internal chunks, without task-specific training, tuning, or ensembling. The suite contains 38 classification and 13 regression datasets. Unless a training phase is specified, results use the models after both training stages. Each evaluation job is run on one A100 GPU.

Following the aggregate reporting of Polewczyk et al. (2026), we report the unweighted mean of per-dataset classification accuracy and soft-clipped regression R^2. The soft-clipping leaves nonnegative

R^2 unchanged and maps negative values through $\tanh(R^2)$ before averaging. Figure 6 compares QuTab at 10,240 rows and 100 columns with reference values reported by FlexTab and TabGemma (Polewczyk et al., 2026; Schindler et al., 2025). Tables 12 and 13 report training-phase and inference-budget comparisons.

D.2 Training phase and inference-envelope sensitivity

Table 12 compares the end of Stage 1's cosine-decay and stable phases with Stage 2 at the same inference budget. From the stable phase to Stage 2, mean regression R^2 rises from 67.95% to 69.52%, while classification accuracy changes from 86.93% to 86.96%. This comparison includes both additional training and the broader curriculum; it does not isolate their individual contributions.

Table 12: **Predictive performance through training.** All phases are evaluated on the same 38 classification and 13 regression datasets at up to 10,240 context rows and 100 columns, using the same A100 evaluation protocol.

Training phase	Mean acc. (%)	Mean soft-clipped R^2 (%)
Stage 1: cosine decay	86.81	66.73
Stage 1: stable	86.93	67.95
Stage 2	**86.96**	**69.52**

The completed curriculum covers up to 10,240 context rows and 512 columns. The larger row budgets in Table 13 therefore lie within the training range, while the 2,048-column setting tests width extrapolation. Increasing the row cap improves both aggregate scores. Wider column caps bring little classification gain and reduce mean regression R^2. Only six TabArena datasets exceed 100 columns, and none reaches both 10,240 rows and 2,048 columns.

Table 13: **TabArena-Lite inference-envelope sweep.** Each row evaluates the same completed models on 38 classification and 13 regression datasets. Caps are the maximum context rows and columns admitted per dataset; bold marks the best mean in each task.

Row/column cap	Mean acc. (%)	Mean soft-clipped R^2 (%)
2,048 / 100	86.57	67.21
10,240 / 100	86.96	**69.52**
10,240 / 512	86.98	68.63
10,240 / 2,048	**87.01**	68.02

D.3 Per-dataset results and regression inductive bias

Tables 14 and 15 report the complete per-dataset results at the 10,240-row, 100-column envelope used in Figure 6. All 51 evaluations completed successfully. The classification scores average to 86.96% and the regression scores to 69.52%. Every regression R^2 is nonnegative, so soft clipping does not change this mean.

The regression deficit remains concentrated on specific datasets. QuTab reaches 44.24% R^2 on `QSAR-TID-11` and 49.78% on `physiochemical_protein`, leaving gaps of approximately 25–29 percentage points to the two reference models in Table 15. `Food_Delivery_Time` remains difficult for all three methods. QuTab is closer to the references on `QSAR_fish_toxicity`, `healthcare_insurance_expenses`, and Fiat-500. Relative to the stable phase, Stage 2 improves 9 of 13 regression datasets, with gains of 7.68, 5.71, and 5.01 points on `QSAR-TID-11`, `wine_quality`, and `physiochemical_protein`, respectively; `concrete_compressive_strength` declines by 4.24 points.

Mechanistic results suggest a specific inductive-bias hypothesis. TabPFNv2 classification uses a sharp label vote, but its regression attention is softer and its predictions track a weighted mean over many context targets; Biloš et al. (2026) relate this change to the bias–variance trade-off for continuous targets. TabICLv2 regression likewise retains diffuse attention and constructs a distributed nonlinear readout. Both mechanisms aggregate context labels broadly.

Table 14: **Per-dataset TabArena-Lite classification accuracy.** Scores use the single frozen classification checkpoint and the predefined outer split.

Dataset	Acc. (%)
Amazon_employee_access	94.32
anneal	98.00
APSFailure	98.87
bank-marketing	89.07
Bank_Customer_Churn	86.29
Bioresponse	74.42
blood-transfusion-service-center	79.60
churn	91.42
coil2000_insurance_policies	93.83
credit-g	75.45
credit_card_clients_default	81.52
customer_satisfaction_in_airline	91.83
diabetes	76.56
Diabetes130US	91.20
E-CommereShippingData	65.78
Fitness_Club	75.60
GiveMeSomeCredit	93.21
hazelnut-spread-contaminant-detection	89.88
heloc	72.07
hiva_agnostic	96.49
HR_Analytics_Job_Change_of_Data_Scientists	79.42
in_vehicle_coupon_recommendation	74.24
Is-this-a-good-customer	89.04
jm1	81.40
kddcup09_appetency	98.21
Marketing_Campaign	88.49
maternal_health_risk	84.62
MIC	87.30
NATICUSdroid	93.63
online_shoppers_intention	88.76
polish_companies_bankruptcy	93.10
qsar-biodeg	88.07
SDSS17	95.88
seismic-bumps	93.27
splice	90.88
students_dropout_and_academic_success	75.25
taiwanese_bankruptcy_prediction	97.10
website_phishing	90.47
Mean	86.96

QuTab instead exposes an addressable, query-centric memory, and our causal analysis of matched synthetic laws localizes a sparse context-target readout whose late heads narrow to one or two rows. Selective retrieval is well matched to discrete class evidence, whereas continuous estimates can benefit from smoothing evidence across examples. We therefore hypothesize that QuTab may favor sparse target retrieval over broad averaging on some regression tasks. Raising the row cap from 2,048 to 10,240 at 100 columns improves mean regression R^2 from 67.21% to 69.52%, showing that additional context remains useful despite this selective readout bias.

The target representation may reinforce this bias. Classification labels use one-hot carriers and fixed phases, whereas regression uses $u = \mathrm{clip}(z/4, -1, 1)$ for rotary phases and retains z in the value carrier (Appendix B.2.2). This gives ordinary targets a multi-scale similarity geometry but maps values beyond four standard deviations to the same phase endpoint. The direct z carrier preserves signed magnitude, so information is not wholly lost, but Structural RoPE no longer distinguishes distances within either clipped tail. Because context states are never rewritten, the query-label state must also infer the smoothing bandwidth and global target statistics.

Table 15: **Per-dataset TabArena-Lite regression R^2.** All methods use the same predefined split. Aligned per-dataset reference values were available only for TabICLv2 and TabPFNv2.6, so the aggregate-only references from Figure 6 are omitted. The TabICLv2 and TabPFNv2.6 values are reconstructed from reported RMSE using the shared test-target variance; their means approximately reproduce the aggregate reference values in Figure 6.

Dataset	QuTab	TabICLv2	TabPFNv2.6
airfoil_self_noise	84.84	97.86	98.23
Another-Dataset-on-used-Fiat-500	83.27	87.03	86.95
concrete_compressive_strength	78.18	95.32	95.13
diamonds	91.53	98.46	98.38
Food_Delivery_Time	25.32	33.65	38.12
healthcare_insurance_expenses	85.25	87.71	87.79
houses	74.35	88.36	88.25
miami_housing	89.62	93.94	94.28
physiochemical_protein	49.78	75.47	74.68
QSAR-TID-11	44.24	69.88	73.31
QSAR_fish_toxicity	64.14	64.48	65.21
superconductivity	85.48	93.10	92.72
wine_quality	47.75	52.53	53.04
Mean	69.52	79.83	80.47

E Efficiency and Scaling

E.1 Ragged Execution

The read-only-memory architecture avoids recurrent full-table updates, but a direct table-wise PyTorch implementation fragments work across many small general matrix multiplications (GEMMs) and attention launches. We therefore use ragged batching (Gale et al., 2023). A physical pack concatenates the context cells, self-feature cells, and query-label states from several shape-compatible tables; compact offsets retain each table's row boundaries and independent softmax domain. Shared projections and MLPs then execute as larger GEMMs, while table-local attention prevents reads across tables. Stage 1 uses the ragged Triton DTA kernel (Tillet et al., 2019); Stage 2 uses the A100 forward and backward kernels with explicit logical-shape masks. Each pack's forward pass is followed immediately by its backward pass before the next pack is evaluated, limiting the lifetime of saved activations. Bfloat16 (BF16) gradients accumulate into 32-bit floating-point (FP32) slots across all packs in the logical update.

Each global update contains 256 logical tables. Classification assigns 64 tables to each of four ranks, whereas regression assigns 128 tables to each of two ranks. In Stage 1, each rank applies deterministic first-fit-decreasing packing under limits of 32 tables, 400,000 cells, 2,048 queries, and 32 million

structural elements per physical pack. Stage 2 uses shape buckets with one, two, or four tables per physical pack, padding capacity dimensions and masking invalid rows and columns; attention and table losses use the logical shapes. The physical packs are processed sequentially while gradients accumulate in FP32 across the complete logical update. The structural-element budget counts the per-query-head row and column workspaces: a table with N context rows, F features, Q queries, and H heads contributes $HQ(N + F)$ elements, summed across all tables in the physical pack.

For reproducibility, the ragged update follows this invariant sequence:

1. Sample 64 classification tables on each of four ranks or 128 regression tables on each of two ranks, for exactly 256 tables globally.
2. Construct deterministic physical packs under the stage-specific capacity policy, preserving table-local softmax domains and masking any padding.
3. Zero gradients once. For every pack, run the BF16 forward pass immediately followed by its backward pass, and accumulate gradients into FP32 slots. Classification uses query-level cross-entropy; regression averages one stabilized RMSE objective per table.
4. After the final local backward, average every complete gradient across ranks using fixed 32 MiB all-reduce buckets.
5. Compute one norm over the complete synchronized gradient surface and clip it at 10. If the norm is non-finite, skip the update collectively before changing optimizer state.
6. Apply batched RMS-matched Muon to the hidden matrices and fused AdamW to the remaining groups, using the saved learning-rate policy.
7. For the sharded Stage 1 optimizer, broadcast updated BF16 parameters from their owners. The replicated Stage 2 optimizer updates identical parameters independently on each rank.
8. Advance the table-based schedule and save only at completed optimizer boundaries. Stage 1 follows warmup, cosine decay, and stable-rate training; Stage 2 continues at the stable rates without restarting the schedule.

Physical repacking changes only execution order. It does not change objective coefficients, table-local attention domains, or the complete matrix seen by each Muon orthogonalization.

E.2 Reproducible Cost Accounting

We compare logical attention-pair scaling and dominant forward contractions between models. Let N denote labeled context rows, Q query rows, $R = N + Q$ total rows, and F feature columns. For each row-compressing model, I is its number of inducing tokens and L_{ind}, L_{feat}, and L_{ICL} are its induced-column, within-row feature, and dataset-level in-context-learning depths. For TabPFNv2, $C = \lceil F/2 \rceil + 1$ is the default number of two-feature groups plus one label token. Table 16 gives the dominant attention-pair terms for the compared architectures.

For QuTab, each of L blocks scores $Q(NF + F)$ context and self-feature cells; history lengths $0, \ldots, L - 1$ add $QL(L - 1)/2$ pairs. Their sum gives Equation 11.

Table 16: Dominant logical attention-pair scaling.

Model	Recurrent representation	Attention pair scaling
TabPFN-3	R row summaries	$2L_{\mathrm{ind}}RFI + L_{\mathrm{feat}}RF^2 + L_{\mathrm{ICL}}(N^2 + QN)$
TabICLv2	R row summaries	$2L_{\mathrm{ind}}RFI + L_{\mathrm{feat}}RF^2 + L_{\mathrm{ICL}}(N^2 + QN)$
TabFM	R row summaries	$2L_{\mathrm{ind}}RFI + L_{\mathrm{feat}}RF^2 + L_{\mathrm{ICL}}(N^2 + QN)$
TabPFNv2	RC grouped cells	$L_{\mathrm{v2}}[RC^2 + C(N^2 + QN)]$
EXAONE Tabular	RF cells + summaries	$L_{\mathrm{EX}}[RF^2 + F(N^2 + QN)]$
QuTab	Q query-label states	$LQ(N + 1)F + \frac{1}{2}QL(L - 1)$

TabPFN-3, TabICLv2, and TabFM build column representations through inducing points before within-row and dataset-level attention (Grinsztajn et al., 2026; Qu et al., 2026; Google Research, 2026a). TabPFNv2 and EXAONE Tabular repeatedly update grouped cells or cell and row states

(Hollmann et al., 2025; Eo et al., 2026). Their dataset-level terms contain N^2, whereas QuTab's fixed-F, fixed-Q table read grows linearly with N.

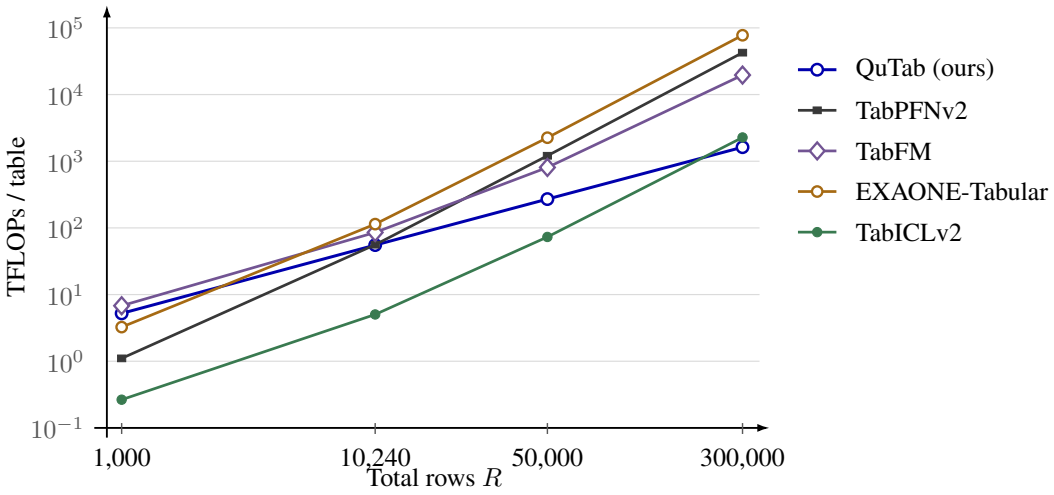


Figure 21: **Analytical forward compute with increasing row count.** Dominant forward FLOPs at $R \in \{1{,}000, 10{,}240, 50{,}000, 300{,}000\}$, $F = 100$, and $Q = 200$. Values beyond 10,240 rows are extrapolations that are not supported by the released QuTab checkpoints.

The released $(I, L_{\text{ind}}, L_{\text{feat}}, L_{\text{ICL}})$ values are $(128, 3, 3, 24)$ for TabPFN-3, $(128, 3, 3, 12)$ for TabICLv2, and $(256, 6, 6, 24)$ for TabFM, where TabFM's effective depths include its two column and two row passes (Grinsztajn et al., 2026; Qu et al., 2026; Google Research, 2026b). TabPFNv2 uses width $d_{\text{v2}} = 192$, six heads, and $L_{\text{v2}} = 12$ alternating feature- and sample-attention blocks, while EXAONE Tabular uses $L_{\text{EX}} = 12$ (Hollmann et al., 2025; Eo et al., 2026). For QuTab, $L = 30$. Let d_c be the cell width, d_q the query-label state width, h_c the cell-SwiGLU width, h_q the query-SwiGLU width, S the stage count, H the query-head count, G the table-memory KV-group count, d_h the head width, $a = 64$ the auxiliary/carrier width, and o the task-output dimension. Counting the implementation's dominant forward contractions, with two floating-point operations per multiply-add, gives

$$\begin{aligned}\mathcal{F}_{\text{QuTab}} = 2\big[&RF(64d_c + 3d_c h_c) + 64Fd_c + S(2RFd_c Gd_h + Rd_c Ga) \\ &+ LQd_q H(2d_h + 7a) + 3LQd_q h_q + 2(L-1)Qd_q Hd_h \\ &+ 2LQHd_h(N+1)F + QHd_h L(L-1) \\ &+ LQHa(5N + 2F + 5) + Q(2d_q^2 + 2d_q o)\big]. \end{aligned} \tag{69}$$

The terms respectively account for scalar Fourier projection and cell encoding; stage K/V and row-key projections; query, output, MLP, and history-writer projections; main and history attention; structural scores and carrier readouts; and the task decoder. DTA is linear in F and has no context–context N^2 term. Setting $Q = \Theta(N)$ makes its QNF term quadratic in rows; the favorable linear row law applies for fixed Q and F.

Table 17: Analytical forward compute for the same $N = 1{,}024$, $Q = 100$, $F = 100$ classification episode. Values count the dominant contractions described above.

Model	Forward TFLOPs
QuTab	3.44
TabFM	7.71

For the direct comparison in Table 17, we set $N = 1{,}024$, $Q = 100$, and $F = 100$ for both models. QuTab uses the dimensions in Table 9 and the ten-logit classification decoder. For TabFM, we count the released classification preset: two three-block induced-column stages with 256 inducing points, two three-block row-attention stages with eight summary tokens, and 24 in-context-learning blocks

of width 2,048 (Google Research, 2026a;b). Its cell width is 256 and all feed-forward widths are four times their input width. The TabFM estimate comprises 2.178 TFLOPs in the column stages, 1.624 in the row stages, 3.847 in the in-context-learning stage, and 0.061 in input and decoder projections; inducing queries attend only to context rows. We include dense projections and attention products in both models and exclude normalization, pointwise Fourier functions, nonlinearities, rotary transforms, softmax, data movement, and optimizer work. Figure 21 evaluates the corresponding forward-compute estimates while holding $Q = 200$ and $F = 100$ fixed. QuTab's forward count rises from 5.219 TFLOPs at $R = 1{,}000$ to 270.468 TFLOPs at $R = 50{,}000$, where TabICLv2 remains cheaper in absolute arithmetic. Continuing the same fixed-shape FLOP ledgers gives a crossover near $R = 212{,}000$ and, at $R = 300{,}000$, about 1,624 TFLOPs for QuTab versus 2,272 for TabICLv2. This is only an asymptotic compute comparison as both row counts exceed QuTab's current 10,240-row codebook.