MECHANISMS AND SYMMETRIES OF TABULAR FOUNDATION MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Tabular foundation models with different architectures converge in accuracy across many classification and regression tasks. This raises questions a leaderboard cannot answer: (i) whether the models execute the same in-context algorithm, and (ii) where row, column, and class-permutation invariances originate. We study released checkpoints from three families through readout rules, interventions, and controlled pretraining comparisons. We can characterize the model behavior with simple similarity-based readouts: TabPFNv2 is better matched by a label vote than a prototype; TabICLv2 shows the reverse pattern. We show a kernel-based model is able to preserve row, column and class symmetries without incurring representation collapse. Motivated by this, we perform full pretraining for different architectural choices and demonstrate that a fully invariant model can match or outperform the default reference model. Together, these results give a mechanistic account of contemporary tabular foundation models and examine how architectural choices relate to their predictions and symmetries.

1 INTRODUCTION

Tabular foundation models (TFMs) make zero-shot predictions for new rows, using a labeled table as model input. TFMs achieve this through in-context learning (ICL) (Qu et al., 2025; 2026) by pretraining on diverse synthetic data. Two key design questions are (i) how we define the model and (ii) how we generate synthetic data for pretraining. In this paper, we investigate the former.

The TFMs studied here share the same setup, which resembles traditional training and prediction. A context table is the equivalent of a training set and contains labeled data (rows), with features and known targets (columns). A query denotes a row whose target we want to predict, given the context.

TabPFNv1 (Hollmann et al., 2023) is the starting point of the modern TFM family. For each row in both context and query, it projects the row to a fixed-size vector, adding a label embedding for context rows. It then applies attention across the rows, allowing queries to attend to context rows and context rows to each other. TabPFNv2 (Hollmann et al., 2025) doesn't immediately collapse the rows, but alternates attention across rows and columns. TabICL (Qu et al., 2025; 2026) encodes columns, then pools features within each row, collapsing them before row attention. Mitra returns to alternating row and column attention without per-row feature pooling (Zhang et al., 2025). TabPFN-3 returns to TabICL's column encoding, per-row feature pooling, and row attention (Grinsztajn et al., 2026); TabFM shares a similar approach (Google Research, 2026).

Performance on benchmarks improves over releases, while shared patterns of row and column attention suggest partial convergence rather than a universally best architecture. A useful contrast is TabuLa-8B (Gardner et al., 2024), a Llama-3 8B autoregressive model over serialized tables. Instead of using a generic architecture, modern TFMs incorporate inductive biases directly into their architectures. This motivates our questions: which architectural choices matter for predictions, and which ensure consistency under equivalent table encodings?

We study released TabPFNv2, TabICLv2, and Mitra checkpoints, then train TabICLv2 variants that change readouts, depth, and symmetries while holding the synthetic task generator and training budget fixed. TabICLv2 releases the data generator, model implementation, and pretraining code needed for this control. A comparison of newer pretrained models would change architecture and training together, so it cannot answer the same controlled question.

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.

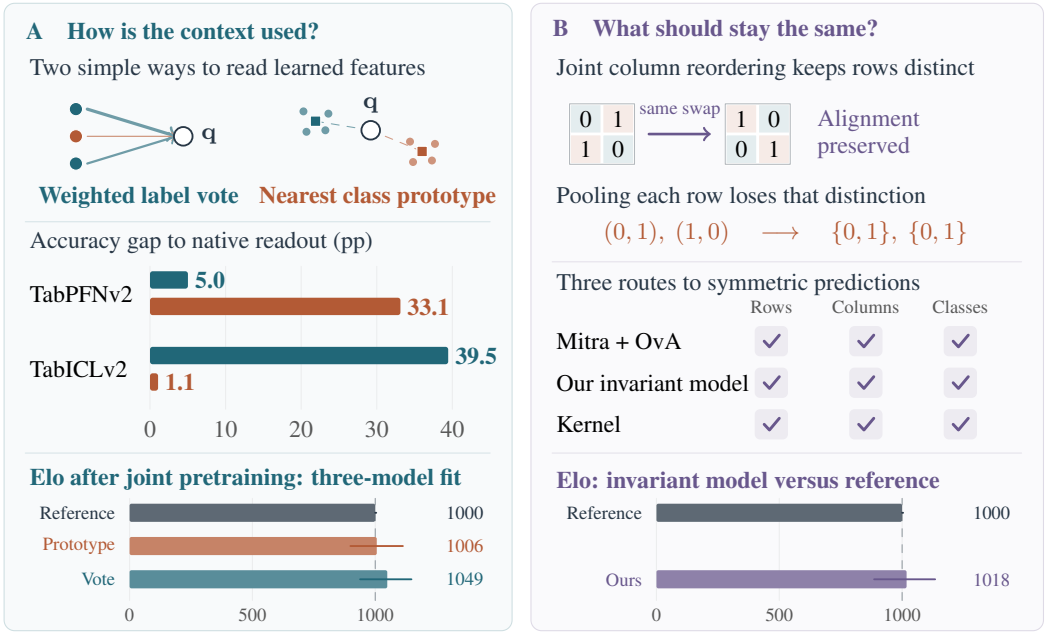


Figure 1: (A): Frozen readout accuracy gaps and a three-model Elo comparison of the jointly pretrained reference, prototype, and vote heads. (B): Column alignment and pooling, three symmetry routes, and a two-model Elo comparison of the fully invariant and reference models.

We summarize our main results in Figure 1. First, the released models use labeled context differently: TabPFNv2 is closer to a label vote, TabICLv2 to prototypes and local neighbors, while Mitra's vote-like pattern changes across versions. We then test whether these readouts can also be learned directly during pretraining by replacing the final output head.

Second, we ask what happens when an input table is encoded differently, even though we expect the result to be identical. We distinguish reordering context rows, reordering feature columns together, relabeling classes, and recoding categorical columns. All of these changes should produce the same predictions because the model should not depend on a particular ordering or encoding.

We find that TFMs usually only implement row-permutation invariance through a deliberate design choice and they have to learn to implement others during pretraining. However, most can be converted to fully invariant models post hoc, without a large drop in performance. For example, we can run One-vs-All (OvA) inference to implement class-permutation invariance exactly. To test this further, we run controlled training comparisons that show that it is possible to train a fully invariant model from scratch without sacrificing performance.

1.1 Notation

We fix notation here for the rest of the paper. A context is a labeled table $D = \{(\mathbf{x}_i, y_i)\}_{i=1}^{n}$ of n rows, where each row's features $\mathbf{x}_i$ span the d feature columns and its target y_i is a label among C classes (or a real-valued target for regression). A query $\mathbf{q}$ is a held-out row whose target we predict given D. We write $\widehat{\mathbf{p}}(\mathbf{q}, D)$ for a model's predicted class-probability vector, and $\mathbf{h}_i, \mathbf{h}_q$ for its internal, per-row representations of a context row or the query at a given layer.

Rows, columns, and classes are the three axes along which D can be re-encoded without changing the underlying prediction problem: reordering the n context rows, reordering the d feature columns consistently for every row, and relabeling the C classes. Section 3 states these as formal invariance and equivariance properties. TabPFNv1/v2, TabICL, Mitra, TabPFN-3, and TabFM are each an instance of the same $D, \mathbf{q} \mapsto \widehat{\mathbf{p}}$ mapping, described briefly above; Appendix A.1 gives block counts, widths, and parameter budgets for the three families we study.

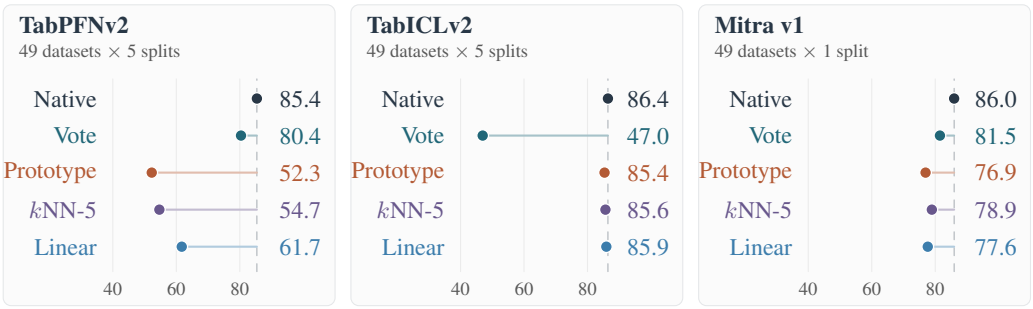


Figure 2: Simple readouts reveal different checkpoint preferences. Mean held-out accuracy (%). Dashed lines mark each checkpoint's native readout.

2 How models turn context into predictions

How well can simple context-based rules decode a frozen TFM representation? Given vector representations $\mathbf{h}_i$ for context rows and $\mathbf{h}_q$ for a query, we define a prototype rule as

$$\boldsymbol{\mu}_c = \frac{1}{|I_c|} \sum_{i \in I_c} \mathbf{h}_i, \qquad \widehat{y}_q = \arg\min_c d(\mathbf{h}_q, \boldsymbol{\mu}_c), \tag{1}$$

where $I_c = \{i : y_i = c\}$ and d is a distance function. This first finds the mean representation for each class and then assigns the query to the nearest class prototype.

A vote rule instead assigns each context row i a nonnegative weight a_{qi} for query $\mathbf{q}$:

$$\widehat{p}_c(\mathbf{q}) = \sum_{i=1}^{n} a_{qi}\, \mathbf{1}[y_i = c]. \tag{2}$$

The weights measure each row's contribution and sum to one. For TabPFNv2, for example, a_{qi} is a query-to-context attention weight at a chosen layer, averaged across heads. We also use local kNN and fit a linear logistic classifier on context representations and labels only. *Frozen* means that checkpoint weights remain unchanged, and *native* is the checkpoint's original trained prediction head.

Each rule uses the same labeled context as the checkpoint and is evaluated only on held-out queries. The layer and rule choices define a diagnostic comparison, rather than a separate model-selection benchmark; the sharp-vote layer is selected per table without query labels as described below. Full extraction and evaluation details are in Appendix A.2.

2.1 What the released models have learned

We start with TabPFNv2. On the 49-dataset, five-split released-checkpoint grid, its label vote is much closer to its native predictions than a prototype rule (Figure 2). In a separate intervention, replacing the corresponding late attention output by the same value for every row reduces accuracy sharply (Table 6). Together these suggest that late, selective access to context contributes to this checkpoint's predictions: a fixed, row-independent summary cannot substitute for it, and a rule that reads out individual rows comes far closer than one that reads out a class average. This does not imply a literal label vote, we test that directly later, with full pretraining.

A second check tells a similar story from the representation's side. Ordinary kNN falls from 77.2% accuracy in input space to 54.9% on TabPFNv2's final representation, while the native model reaches 85.5% (Table 7). The final representation is therefore not a cleaned feature space in which nearby points share a label. With the intervention above, this points to a readout that depends on learned, query-specific attention rather than local geometry.

For Mitra v1, the late attention vote is closer to the native prediction than the prototype or kNN rules, but still has a local gap of about 4.5 percentage points (Figure 2, Table 3). A separate v1.1 check gives a larger gap, so neither result establishes an exact label-voting mechanism (Appendix A.2).

Mitra is vote-like in the same qualitative sense as TabPFNv2, but the fit is looser, so we treat it as a weaker instance of the same pattern rather than a second confirmation of it.

TabICLv2 shows the opposite pattern, and this contrast is the main finding of this subsection. In the 49-dataset, five-split comparison, class prototypes and local kNN on its final representation approach native top-1 accuracy, whereas the TabPFNv2-style vote does not (Figure 2). The sharp vote uses the block whose attention concentrates most strongly on individual context rows, selected without query labels. Prototype and local-neighbor predictions agree with the native predictor on more than 92% of queries, while this vote agrees on about half, and the context-fitted linear head also comes close to native accuracy (Tables 3 and 5). Unlike TabPFNv2, TabICLv2 arranges context rows so that a class average or a few nearby neighbors recovers most of what the native head knows.

This pattern is not unique to a particular release, which suggests it reflects a property of the architecture rather than of a training run. In the same 49-dataset, five-split protocol, the prototype is within about two percentage points of native accuracy for TabICL v1 and v1.1 and within about one point for v2. Nearest neighbors are also close in all three releases, while the vote remains much weaker (Table 4).

Matching predicted classes does not ensure matching probabilities, which limits how far the readouts above can be trusted. Measuring expected calibration error, the mismatch between confidence and observed accuracy, the high-accuracy TabICL prototype has 46.3% against 4.2% for the native predictor (Table 11). Its probabilities are an untuned unit-temperature softmax of cosine similarities to class centroids. The representation supports accurate class choices, but this conversion of distances to probabilities does not recover the model's confidence.

For regression, distance-weighted kNN with ten neighbors improves from $R^2 = .433$ in the input to $R^2 = .566$ on the final TabICLv2 representation, while the native regressor reaches $R^2 = .805$ (Table 10). The learned representation helps this local rule, but the classification prototype pattern does not extend automatically to continuous targets.

2.2 How the representation develops

Given that TabICLv2's final representation already supports simple, accurate readouts, we ask whether that property is a late-layer effect or builds up gradually across the stack. We examine the released TabICLv2 checkpoint by fitting cosine kNN on context representations and evaluating it on held-out queries at the output of blocks 1 through 12. Mean accuracy rises overall through the stack (Figure 3; Appendix A.2), suggesting that the representation becomes progressively more useful for this local kNN readout rather than only becoming useful in the last one or two blocks.

In a separate intervention, all learned weights remain fixed while query-to-context attention is made uniform over context rows in selected blocks. Changing either tested end block has a modest median loss, whereas changing the latter six blocks or the whole stack has a larger one. This supports distributed refinement in the trained checkpoint—no single block carries the readout on its own. That raises a question about necessity rather than redundancy, whether the model needs all twelve blocks or a shallower one could learn the same job, which we answer next by retraining rather than probing the backbone at different depths.

2.3 Training with different readouts and depths

The frozen probes above diagnose the checkpoint TabICLv2 happens to release, but they cannot decide whether a vote head could be learned instead: the backbone was never trained to make voting work. To ask the causal version, we replace TabICLv2's output head and train the backbone with the replacement for 500,000 steps on the same synthetic data. The prototype head is implemented to score squared Euclidean distances to context class centroids. The vote head learns a temperature-scaled dot-product affinity over context rows and averages their one-hot labels. Unlike the frozen probes, both can change the representation during training. The evaluation uses TabArena tasks with the complete protocol in Appendix A.4.

Figure 1 compares the reference, prototype, and vote heads directly. Both alternative heads perform close to the reference, with a modest advantage for the vote and task outcomes of 21–0–17 against the prototype (wins–ties–losses). This resolves the tension left by the frozen probe: a frozen vote fails not because voting is unsuitable for TabICLv2, but because it is unsuitable for the representation

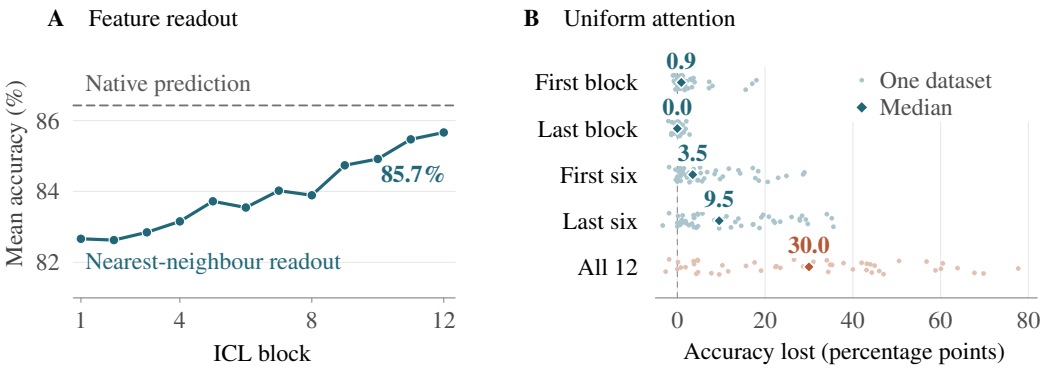


Figure 3: TabICLv2 depth and attention interventions. Left: mean cosine-kNN accuracy by ICL block. Dashed is native accuracy. Right: Enforcing uniform attention in various layers. Dots are dataset means across splits, diamonds are medians over datasets.

the reference happened to learn. Trained together, representation and vote head co-adapt and recover reference-level performance. The output head is one of several readouts the same backbone can be trained to support, not privileged information it must discover.

To test the influence of depth we train models with 1, 3, or 6 ICL blocks instead of 12, keeping everything else fixed. All three rank below the 12-block reference (Figure 4, which shows Elo across all variants and each variant against the unchanged TabICLv2 model). This qualifies the depth intervention above: since the deeper model retains an advantage when trained, the small loss from intervening on a single block does not imply blocks can be dropped during training for free. Mean ranks are in Appendix A.4.

2.4 A CONNECTION TO KERNEL REGRESSION

The vote rule in Eq. (2) has a classical interpretation that will let us build a fully symmetric baseline in the next section. If $a_{qi} = k(\mathbf{q}, \mathbf{x}_i) / \sum_j k(\mathbf{q}, \mathbf{x}_j)$ for a specified strictly positive symmetric kernel k, Eq. (2) is normalized kernel regression, or Nadaraya–Watson label averaging (Nadaraya, 1964; Watson, 1964). Learned attention still differs, since it can use contextual, label-dependent, asymmetric scores and need not be a fixed positive-semidefinite kernel, while our prototype rule aggregates context rows within each class before comparison. The next section uses this distinction to construct an exactly symmetry-preserving kernel predictor (Appendix A.5).

From readouts to symmetries. Both rules make label entries explicit—a prototype aggregates context members within a class, a vote transports labels from individual rows. Neither makes a whole predictor class equivariant on its own, since representations, class channels, and output normalization may still break symmetry. Therefore, we track every stage of the modeling procedure in §3.

Conclusion. Taken together, the results show TFMs do not have a single mechanism how to turn context into predictions. We also find that the readout preference is not fixed by the backbone since retraining the same architecture with a different head closes most of the gap. In contrast, depth appears to be important during training which tracks with the probing results.

3 SYMMETRIES IN TABULAR MODELS

Symmetry means model's predictions stay the same when the same table is written down differently. This includes rows in a different order, columns in a different order, or classes given different numeric codes. This is a separate goal from accuracy, and a property of the whole prediction procedure rather than the backbone alone, since preprocessing and inference choices can break a symmetry the backbone has, or supply one it lacks. Figure 1 previews three routes to full row, column, and class

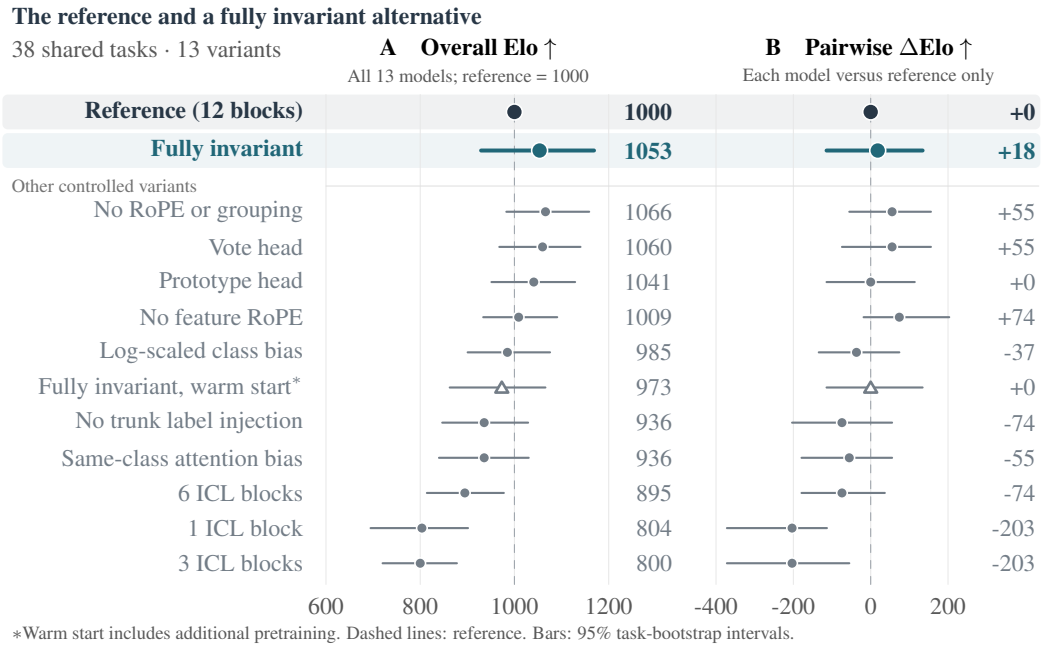


Figure 4: Overall Elo and direct comparisons with the reference. The highlighted model is invariant to row and column permutations and equivariant to class relabeling, not nominal-category recoding. Intervals reflect task variation, not retraining uncertainty.

symmetry, which we take in turn: wrapping Mitra in a one-vs-all inference procedure, a hand-built kernel predictor, and a model trained end to end to be invariant.

3.1 Invariance for free in existing models

All three released backbones already ignore context-row order at the representation level, but diverge on the feature and class axes. TabPFNv2 uses feature grouping and feature-position parameters; TabICLv2 uses RoPE on column positions and, in v2, circular grouping; Mitra uses no feature positions at all, so its column treatment is already compatible with a joint feature permutation.

In principle, both approaches can be switched off in the released checkpoints without retraining (Appendix A.3). Zeroing TabPFNv2's per-feature positional weight matrix, keeping its bias, makes it exactly invariant to permutations of and within its feature groups at unchanged accuracy, .854. Removing feature RoPE from TabICLv2 gives exact circular-shift invariance for .1 pp, with no dataset losing over 5 pp, while the same edit costs TabICL v1 15.4 pp, which has no circular grouping to fall back on. Exact joint-column symmetry needs grouping gone too, which §3.3 does in pretraining.

We pretrain three TabICLv2 models, 1) without RoPE, 2) without grouping and 3) without both. The final model is verified to be fully column-permutation invariant while retaining the accuracy of the original model (§3.3, Table 19).

None of the released TFM models are symmetric under class relabeling. This can have real practical implications as changing the class labels flips 3.3% TabPFNv2's prediction and 2.3% for TabICLv2 (Appendix A.2). Whether we assign two target labels values '0' and '1' or '1' and '0' should have no influence on the final prediction. Because of this, we propose an overlay method for the existing checkpoints which makes them invariant to these changes by default.

Relabeling classes with OvA. One-vs-all (OvA) inference encodes each of C classes against its complement, evaluates the same binary predictor C times, and maps the scores back. Because the binary scores are deterministic, relabeling classes only relabels these subproblems, so the score vector

is equivariant. The reason this might work on pretrained models is because they were trained to perform any kind of classification, including binary, which was present in the training data.

The same wrapper works for all three families (Table 9). On a 21-dataset multiclass cohort, OvA reaches .877 accuracy for TabPFNv2 and .880/.890 for TabICL v1/TabICLv2, each with exact $(21/21)$ label agreement under class recoding, and Mitra v1 holds accuracy at .841 with 99.6% agreement. OvA removes essentially all of the class-relabeling sensitivity above, for any checkpoint, at the cost of C binary evaluations instead of one multiclass call.

3.2 Symmetry without collapse

Mitra's column invariance comes from pooling across features with no positional encoding. Qu et al. (2025) call a failure mode of such pooling *representation collapse* where distinct rows pool to the same representation and become indistinguishable downstream. Column symmetry is easy to confuse with this much stronger invariance, so we separate the two.

A joint feature-column permutation acts on a whole table, $(\sigma\mathbf{X})_{ij} = X_{i,\sigma(j)}$ for every row i, preserving which value belongs to which column. An independently pooled row map $\mathbf{r}(\mathbf{x}) = \boldsymbol{\rho}(\sum_{j=1}^{d} \boldsymbol{\phi}(x_j))$, the set-pooling map behind Deep Sets (Zaheer et al., 2017), is instead blind to which feature supplied which value. It is invariant to a *separate* permutation of every row's features, a larger group than joint column permutation, and exactly the mechanism behind collapse.

Let feature names be a, b, rows $(0, 1)$ and $(1, 0)$, and labels that depend on a. Swapping a, b jointly preserves a valid problem, but a row-wise multiset representation maps both rows to $\{0, 1\}$ and cannot tell them apart. We test collapse empirically on *balance-scale* and a twelve-feature analogue, where columns share the marginal distributions but play different roles in the target (Appendix A.2, Table 12). Both Mitra versions solve these tasks with no feature-position layers, so joint column invariance does not imply collapse.

A kernel example. Now we ask a more general question. Can all four symmetries, row, column, class, and categorical recoding, coexist with an accurate predictor? Let $\mathcal{N}$ and $\mathcal{C}$ denote the numerical and nominal column index sets. For $h > 0$ and a nonempty context D, define

$$\begin{aligned} \rho(\mathbf{x}, \mathbf{x}') &= \sum_{j \in \mathcal{N}} (x_j - x'_j)^2 + \sum_{j \in \mathcal{C}} \mathbf{1}\{x_j \neq x'_j\}, \\ k_h(\mathbf{x}, \mathbf{x}') &= \exp[-\rho(\mathbf{x}, \mathbf{x}')/h^2], \\ \widehat{p}(c \mid \mathbf{q}, D) &= \frac{\sum_i k_h(\mathbf{q}, \mathbf{x}_i)\mathbf{1}\{y_i = c\}}{\sum_i k_h(\mathbf{q}, \mathbf{x}_i)}. \end{aligned} \tag{3}$$

Proposition 1 (Symmetry and consistency)**.** *For every $h > 0$ and nonempty context D, with missingness represented by a distinct symbol, the predictor in Eq. (3) is invariant to context-row and joint feature-column permutations, equivariant to class relabeling, and invariant to bijective recodings. It self-classifies a finite context of distinct rows as $h \to 0$. For i.i.d. contexts and an independent iid query, compact numerical support without numerical missingness, finite categorical alphabets, and a fixed finite label set, the associated Nadaraya–Watson classifier with $h_n = n^{-1/[2(p+1)]}$ has expected risk converging to Bayes risk, where p is the number of numerical coordinates.*

Appendix A.5 realizes this as an attention-style network and proves its symmetries. Appendix A.6 proves consistency. The example says that these symmetries are compatible with learning, but nothing on if the construction is a practical TFM. This is still a useful result since symmetry itself is not the obstacle to an accurate predictor.

3.3 Training symmetric procedures

The third route trains a model with the symmetries incorporated in the model. We pretrain 13 TabICLv2 checkpoints ourselves: a reference and twelve variants, each changing one named architectural choice with the data and the optimization held fixed. Every run uses the same cached graph-SCM task generator, 500,000 updates at global batch size 64, the same width-128 backbone, and the same seed, and each is a single pretraining run, so the comparisons carry no seed variance. Some of the results have already been presented in the previous sections.

The variants fall into four families. *Depth* shortens the ICL trunk to 1, 3, or 6 blocks and *readout* swaps the learned MLP head for the prototype or vote twin, both already used in §2. *Feature axis* removes TabICLv2's two feature-position devices, first RoPE over the feature and CLS slots within a row, then circular feature grouping as well. *Label pathway* changes how label identity enters the trunk, in five variants that all carry the feature-axis removals and the vote head. Dropping the trunk's class-indexed label embedding, replacing it with a bias on same-class attention scores, that bias rescaled with sequence length, the fully invariant model below, and its warm start. We score 38 TabArena tasks (Erickson et al., 2025) by Elo over pairwise task wins, anchoring the reference at 1,000 (Figure 4, Appendix A.4).

Partial invariant models. Dropping RoPE and grouping together buys exact joint-column equivariance at no measured cost, as quoted in §3.1. Class symmetry does not yield to the same treatment. The reference model injects label identity in two places through a class-indexed embedding, 1) in the target-aware column embedder and 2) in the ICL trunk. The three label-pathway variants replace only the trunk copy and all three use the vote head, which is class equivariant by construction, yet all three still move their predictions under a class relabeling.

The fully invariant model. The last variant replaces *both* class-indexed embeddings with a content injection that depends only on which context rows share a label, never on the label value. Each row receives the leave-one-out mean representation of the other context rows of its class, through an MLP shared across classes. With the vote head and the feature-axis removals, it is the one checkpoint in the set that is invariant to context-row order, invariant to joint column permutation, and equivariant to class relabeling at once. This is also confirmed empirically on real tables and their permutations, resulting in identical predictions. We do not cover categorical feature recoding, which the kernel of §3.2 does, because the default TabICLv2 pretraining does not distinguish categorical columns.

Its accuracy does not pay for this: it reaches Elo 1053 against the reference's 1000, mean rank 5.50 against 6.42, and wins 20 of 38 tasks (Table 16). The bootstrap intervals overlap the reference, so the honest reading is a match rather than an improvement. Note that the all the hyperparameters and the full training procedure in the model are identical to the reference which is why we didn not expect larger differences.

We also test if there is any benefit to the warm start—an invariant model that starts training from an already pretrained non-invariant checkpoint. Compared to the random initialization, performing 500,000 additional updates lands at Elo 973, below its from-scratch twin. We conclude that the invariant model is sufficiently different to the reference in terms of the underlying mechanisms its weights learn.

With all three routes demonstrated in this section we can positively confirm that an inference-time fix, an exact kernel, and end-to-end training all reach row, column, and class symmetry. At the same time, models do not pay a cost that would justify leaving them out by default.

4 Related work

From prior fitting to tabular models. Prior-fitted networks (Müller et al., 2022) first showed that a network can learn to predict from labeled examples purely by pretraining on synthetic tasks. TabPFN (Hollmann et al., 2023) adapted this approach to small tabular classification, representing each row as one token and applying attention over rows. TabPFNv2 (Hollmann et al., 2025) kept cell-level representations instead, alternating attention along the row and feature axes to support regression and variable-width tables, and TabPFN-2.5 (Grinsztajn et al., 2025) kept that design while adding depth, larger feature groups, and learned extra rows. TabICL (Qu et al., 2025) takes a different route: it models column distributions first, pools features within each row, and only then applies row-level ICL, and TabICLv2 (Qu et al., 2026) broadens its prior and updates attention and optimization for scale. Mitra (Zhang et al., 2025) returns to a persistent cell grid with alternating row and column processing, changing the synthetic prior along with the architecture. TabPFN-3 (Grinsztajn et al., 2026) and TabFM (Google Research, 2026) both go back to staged column processing, per-row feature pooling, and row-level interaction, with TabPFN-3 adding an attention-weighted label readout on top. Two branches differ more sharply: TabDPT (Ma et al., 2025) uses real-table self-supervision and retrieved context rather than a synthetic prior, and TabuLa (Gardner et al., 2024) serializes a table

as text for an autoregressive language model. These releases keep returning to the same three choices we study here: whether columns are processed before rows, whether features are pooled per row or kept as a persistent grid, and how the label enters the computation.

Understanding tabular foundation models. A smaller line of work asks how these models arrive at their predictions rather than how well they score. McCarter (2025) probes TabPFN's input–output behavior on synthetic decision functions and under input changes. Ye et al. (2025) randomize TabPFNv2's feature tokens and present held-out context folds as queries to derive discriminative embeddings; those matched query-role embeddings differ from the context-role features our readout probes use. Hu & Ghelichi (2026) relate TabPFN's behavior under irrelevant features and label noise to attention concentration, while Gupta et al. (2026) decode linear coefficients and arithmetic intermediates from TabPFNv2 on synthetic functions. Closest to our own setup, Miftachov et al. (2026) fine-tune a TabICL trunk with explicit dot-product and Gaussian kernel heads, using the same embedding function for context and query rows. That symmetry is distinct from the invariance to row order, column order, class labels, and recoding we study in Section 3. None of this work compares readout behavior across model families, or asks whether a model's invariance properties are a deliberate design choice or an accident of training, which is the gap our readout and symmetry results fill.

Mechanisms and invariance in ICL. A broader line of work asks which algorithms transformers can learn in context at all. Garg et al. (2022) show that transformers trained from scratch learn simple function classes from context alone, and in linear settings Akyürek et al. (2023) and Von Oswald et al. (2023) connect this to ridge regression, least squares, and iterative updates. Han et al. (2025) go further and connect language-model ICL to normalized kernel regression through attention and hidden features. All of it studies models trained from scratch on tractable function classes, not released checkpoints on real tabular tasks, so it cannot identify the computation inside a deployed TFM; our probes and interventions target that gap. Separately, Deep Sets (Zaheer et al., 2017) shows that pooling a set gives permutation invariance, but pooling features separately within each row can discard the feature alignment across rows, the same failure our two-row example in Section 3 makes concrete. Our mixed-kernel construction shows that row and joint-column invariance can coexist with that alignment instead of trading it away.

5 Discussion and Limitations

We asked whether tabular foundation models that converge in accuracy also converge in mechanism, and whether the row, column, and class symmetries a practitioner might take for granted actually hold. Neither does by default. TabPFNv2 leans on late, row-selective attention in a way that resembles a label vote, TabICLv2 arrives at a representation where class prototypes and local neighbors already match its native head, and none of the three released families is fully symmetric out of the box.

Both gaps turn out to be fixable at little cost. Retraining TabICLv2 with a different readout closes most of the gap to its reference head, and OvA inference, the mixed kernel, and a fully invariant trained model each reach full symmetry without a large drop in accuracy. The common thread is that readouts and symmetries are properties of the whole prediction procedure, learned weights, table representation, preprocessing, and inference together, not of the backbone in isolation, and that a symmetry-preserving procedure need not reduce rows to uninformative summaries.

Limitations. We hold the synthetic pretraining mixture fixed, so we cannot separate the architecture's contribution to the mechanisms we audit from the data generator's. To do that we would need the same comparisons repeated across priors.

AI use statement

In this work, we used generative AI tools for coding and writing assistance. We have reviewed all AI-assisted work. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REFERENCES

Ekin Akyürek, Dale Schuurmans, Jacob Andreas, Tengyu Ma, and Denny Zhou. What learning algorithm is in-context learning? Investigations with linear models. In *International Conference on Learning Representations*, 2023.

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 *Thirty-ninth Conference on Neural Information Processing Systems Track on Datasets and Benchmarks*, 2025.

J. Gardner, J. C. Perdomo, and L. Schmidt. Large scale transfer learning for tabular data via language modeling. In *Advances in Neural Information Processing Systems*, 2024.

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

Google Research. TabFM. GitHub repository, 2026. URL https://github.com/google-research/tabfm.

Léo Grinsztajn, Klemens Flöge, Oscar Key, Felix Birkel, Philipp Jund, Brendan Roof, Benjamin Jäger, Dominik Safaric, Simone Alessi, Adrian Hayler, Mihir Manium, Rosen Yu, Felix Jablonski, Shi Bin Hoo, Anurag Garg, Jake Robertson, Magnus Bühler, Vladyslav Moroshan, Lennart Purucker, Clara Cornu, Lilly Charlotte Wehrhahn, Alessandro Bonetto, Bernhard Schölkopf, Sauraj Gambhir, Noah Hollmann, and Frank Hutter. TabPFN-2.5: Advancing the state of the art in tabular foundation models, 2025. URL https://arxiv.org/abs/2511.08667.

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, 2026.

Aviral Gupta, Armaan Sethi, and Dhruv Kumar. Tabpfn through the looking glass: An interpretability study of tabpfn and its internal representations. *arXiv:2601.08181*, 2026.

Chi Han, Ziqi Wang, Han Zhao, and Heng Ji. Understanding emergent in-context learning from a kernel regression perspective. *Transactions on Machine Learning Research*, 2025.

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

Noah Hollmann, Samuel Müller, Lennart Purucker, Arjun Krishnakumar, Max Körfer, Shi Bin Hoo, Robin Tibor Schirrmeister, and Frank Hutter. Accurate predictions on small data with a tabular foundation model. *Nature*, 2025.

James Hu and Mahdi Ghelichi. Noise immunity in in-context tabular learning: An empirical robustness analysis of TabPFN's attention mechanisms, 2026. URL https://arxiv.org/abs/2604.04868.

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

Calvin McCarter. What exactly has TabPFN learned to do?, 2025. URL https://arxiv.org/abs/2502.08978.

Ratmir Miftachov, Bruno Charron, and Simon Valentin. Interpretable tabular foundation models via in-context kernel regression. *arXiv preprint arXiv:2602.02162*, 2026.

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

Elizbar A. Nadaraya. On estimating regression. *Theory of Probability & Its Applications*, 9(1): 141–142, 1964.

Jingang Qu, David Holzmüller, Gaël Varoquaux, and Marine Le Morvan. TabICL: A tabular foundation model for in-context learning on large data. In *International Conference on Machine Learning*, 2025.

Jingang Qu, David Holzmüller, Gaël Varoquaux, and Marine Le Morvan. TabICLv2: A better, faster, scalable, and open tabular foundation model. *arXiv preprint arXiv:2602.11139*, 2026.

Johannes Von Oswald, Eyvind Niklasson, Ettore Randazzo, João Sacramento, Alexander Mordvintsev, Andrey Zhmoginov, and Max Vladymyrov. Transformers learn in-context by gradient descent. In *International Conference on Machine Learning*, 2023.

Geoffrey S. Watson. Smooth regression analysis. *Sankhyā: The Indian Journal of Statistics, Series A*, pp. 359–372, 1964.

Han-Jia Ye, Si-Yang Liu, and Wei-Lun Chao. A closer look at TabPFN v2: Understanding its strengths and extending its capabilities, 2025. URL https://arxiv.org/abs/2502.17361.

Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Russ R Salakhutdinov, and Alexander J Smola. Deep sets. In *Advances in Neural Information Processing Systems*, 2017.

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 (NeurIPS)*, 2025.

A SUPPLEMENTARY MATERIAL

A.1 MODEL ARCHITECTURES

This subsection gives the architectural detail behind the one-paragraph summary in §1: block counts, widths, and parameter budgets for the three released families we probe directly, TabPFNv2, TabICLv2, and Mitra.

TabPFNv2 (Hollmann et al., 2025) is a per-cell transformer (hidden width 192, 12 blocks) operating on the $(n_{\text{rows}}+1) \times n_{\text{cols}}$ table formed by the context set plus the query row. Each block alternates row-wise attention, mixing across context rows for a fixed column, with column-wise attention, mixing across columns for a fixed row, interleaved with MLPs. Inputs pass through a fixed three-copy preprocessing pipeline (raw, scaled, quantile-rotated). Numerical features are encoded via a per-feature positional matrix and bias, and class identity is read off the query cell at the output head. Of its 7.24M parameters, 7.08M sit in the 12-block stack.

TabICLv2 (Qu et al., 2025) is a row-then-ICL transformer with three stages: three column-embedding blocks, three pre-label row-interaction blocks, and then the 12-block row-level ICL transformer that mixes context rows with the query row after labels are injected. The first two stages process the table column by column and emit one vector per row; the third column-embedding block is a SetTransformer readout that compresses per-feature into per-sample vectors before the row stage. RoPE is applied along the feature dimension and circular feature grouping is applied to inputs. The class prediction is read off the final ICL block's output for the query row. Its 27.55M parameters split roughly 0.88M / 0.40M / 26.28M across the column, row, and ICL stages.

Mitra (Zhang et al., 2025) is a 12-block transformer trained with mixed synthetic priors and released as part of AutoGluon. It differs from TabPFNv2 and TabICLv2 along three axes that our main analysis singles out: labels live in a separate per-row token slot rather than being added to feature tokens, the per-feature embedding is shared across columns, and the architecture has no positional encoding on either axis. Each block is an interleaved row/feature layer, where a row-attention call lets each query token attend to the context tokens for the same feature and a column-attention call mixes across features within a row. Both released checkpoints (v1 and v1.1) share this structure and differ only in their training mixtures. We load Mitra as a single model with AutoGluon's ensembling and stacking disabled, so we measure the model in isolation rather than the meta-ensemble it ships with by default. Of its 75.67M parameters, 75.66M sit in the 12-block interleaved stack; the remainder is the per-feature embedding, the label embedding, and the final classification head.

The released TabPFNv2 checkpoint used here is the 12-block PerFeatureTransformer, not the newer 24-block v2.5/v2.6 architectures, which are released under licenses incompatible with this analysis pipeline and are therefore out of scope.

Table 1: Architecture accounting for the three main classification models, derived from the released model files used throughout this paper.

Model	Params	Core stack	Heads	FFN hidden	Normalization
TabPFNv2	7.24M	12 PerFeatureTransformer blocks, d=192, no thinking rows	6 feature + 6 item	768	3 bias-free LN / block
TabICLv2	27.55M	ColEmb 3 + Row 3 + ICL 12, d_{row}=128, d_{ICL}=512	8 / 8 / 8	256 / 256 / 1024	pre-norm LN with bias
Mitra	75.67M	12 interleaved row/column blocks, d=512, separate per-row label slot, no positional encoding	4 row + 4 col	2048	4 LN with bias / block

A.2 EVALUATION DETAILS AND SUPPORTING RESULTS

Each task consists of a labeled context $D = \{(\mathbf{x}_i, y_i)\}_{i=1}^{n}$ and held-out queries. Query labels are not available to preprocessing, a backbone, or a post-hoc readout. A surrogate head is fitted on context representations and labels only, then evaluated on held-out query representations.

Datasets. The study inventory is 49 classification datasets (four synthetic, three from `sklearn`, and 42 OpenML tasks) and ten OpenML regression datasets. The classification names are

Table 2: Shape and accounting proxy for the three audited classification models on a single-query context set with n context rows and d raw features. The "attention-work proxy" counts attention-matrix elements and ignores heads, MLPs, and constant factors; it is a scaling summary, not an exact FLOPs count.

Model	Serialized state	Attention-work proxy	Measured cost
TabPFNv2	$3(n+1)d$ cell tokens throughout the 12-block stack	$3d(n+1)^2$ from row attention + $(n+1)(3d)^2$ from column attention, per block	~ 1.0 s, ~ 3 GiB
TabICLv2	d feature tokens / row until ColEmb-2, then $n+1$ row tokens through Row 3 + ICL 12	$3(n+1)d^2$ in the ColEmb stage + $15(n+1)^2$ across Row 3 and ICL 12	~ 1.5 s, ~ 4 GiB
Mitra	$(n+1)(d+1)$ tokens per layer (d feature cells + 1 label slot per row), through the 12-block stack	$d(n+1)^2$ from row attention + $(n+1)(d+1)^2$ from column attention, per block	—

`Fitness_Club`, `Is-this-a-good-customer`, `LED-display-domain-7digit`, `MIC`, `MiceProtein`, `analcatdata_authorship`, `analcatdata_dmft`, `anneal`, `balance-scale`, `banknote-authentication`, `blood-transfusion-service-center`, `breast-w`, `breast_cancer`, `car`, `climate-model-simulation-crashes`, `cmc`, `credit-approval`, `credit-g`, `cylinder-bands`, `diabetes`, `dresses-sales`, `eucalyptus`, `hill-valley`, `ilpd`, `iris`, `kc2`, `maternal_health_risk`, `mfeat-factors`, `mfeat-fourier`, `mfeat-karhunen`, `mfeat-morphological`, `mfeat-pixel`, `mfeat-zernike`, `pc1`, `pc3`, `pc4`, `qsar-biodeg`, `quadrant_2d`, `random_labels`, `red_wine`, `semeion`, `sign_1d`, `steel-plates-fault`, `tic-tac-toe`, `vehicle`, `wdbc`, `website_phishing`, `wine`, `xor_2d`.

The regression names are `Another-Dataset-on-used-Fiat-500`, `Moneyball`, `QSAR_fish_toxicity`, `airfoil_self_noise`, `cars`, `concrete_compressive_strength`, `energy_efficiency`, `forest_fires`, `healthcare_insurance_expenses`, and `socmob`.

Readout evaluation. The common TabICLv2 classification grid (Qu et al., 2025; 2026) uses 49 datasets and five seeds. For every seed, the released train and test parts were concatenated and repartitioned into a fresh stratified 80/20 context/query split. Table 4 gives raw aggregate means. The TabPFNv2 portion (Hollmann et al., 2025) uses an evaluation configuration without fingerprint augmentation or feature-shift ensembling.

Readout rules. For TabPFNv2, the vote averages layer 9 (L9) query-to-context scaled-dot-product attention over heads, uses the weights to average one-hot context labels, and then normalizes each query's class scores. It is a reconstructed readout, not the native output head. Prototype and kNN readouts use the final-layer L11 representation at the target slot; the linear comparison uses the same L11 representation. For TabICLv2, these readouts use the row token after block 12. The Mitra-v1 (Zhang et al., 2025) main readouts are taken at L9. In every case the representation is split into context and held-out query rows before fitting a readout.

The TabICLv2 sharp-vote layer is chosen separately for each fitted table: among the ICL blocks, it maximizes the mean largest query-to-context attention weight, averaged over heads and query rows. No query labels enter this selection. The selected attention is averaged over heads and used to average one-hot context labels. Prototypes are context class centroids compared by cosine similarity, with a unit-temperature softmax for reported probabilities. The kNN rules use Euclidean distance, distance weighting, and $k = 5$. The linear comparison is an unstandardized but regularized logistic classifier with $C = 1$. Regression uses the analogous distance-weighted kNN regressor fit on context rows. For the uniform-attention intervention, selected TabICLv2 layers replace a query's attention output with the uniform average of the projected context values within each head. These procedures are fixed before query-side evaluation.

Mitra vote across implementations. The Mitra v1 comparison in Table 3 uses 49 datasets and one split per dataset. A separate Mitra v1.1 evaluation reconstructs layer-9 query-to-context attention

on 42 OpenML datasets with three seeds. Across its 126 evaluations, the original predictor has 84.4% mean accuracy and the label vote has 73.3%; their predicted labels agree on 79.1% of queries, averaged over evaluations. The larger gap than in the v1 comparison cannot be attributed to the checkpoint version alone because the datasets and vote implementation also differ. Both checks support an imperfect correspondence with label voting, not an exact account of Mitra's predictions.

Depth probes and attention interventions. Figure 3 uses 49 classification datasets with five stratified 80/20 context/query splits. At each of the 12 ICL block outputs, we fit a distance-weighted five-neighbor classifier on context representations using cosine distance and evaluate held-out queries. Mean accuracy rises overall from 82.7% after the first block to 85.7% after the twelfth, compared with 86.4% for the original predictor. This is a probe of intermediate representations, not early-exit inference with the model's trained output head. In the separate attention intervention, we keep the checkpoint weights fixed and replace query-to-context attention by uniform averaging in the first block, last block, first six blocks, last six blocks, or all 12 blocks. We first average each dataset's accuracy loss over its five splits, then take the median across datasets: the losses are 0.9, 0.0, 3.5, 9.5, and 30.0 percentage points, respectively. The figure also shows every dataset's mean loss so that these medians do not hide the variation across tasks.

Table 3: Released-checkpoint classification readouts.

Family	Native	Vote	Prototype	kNN-5	Linear
TabPFNv2	.854	.804	.523	.547	.617
TabICLv2	.864	.470	.854	.856	.859
Mitra v1	.860	.815	.769	.789	.776

Table 4: TabICL classification readouts.

Release	Native	Prototype	kNN-5	Sharp vote
v1	.852	.828	.836	.590
v1.1	.852	.830	.842	.626
v2	.864	.854	.856	.470

Table 5 records how often the surrogate and native hard predictions agree on the common grid. These agreements describe prediction overlap, not equality of probabilities.

Table 5: Agreement with native hard predictions.

Backbone	Readout	Agreement (%)
TabPFNv2	L9 vote	**87.4**
TabPFNv2	L11 prototype	51.8
TabICLv2	Block 12 prototype	**92.2**
TabICLv2	Block 12 kNN-5	**93.6**
TabICLv2	Sharpness vote	49.6

For the TabPFNv2 intervention in Table 6, the late L9 attention output is replaced by a constant shared across rows rather than by uniform attention.

Table 7 covers 49 datasets and ten splits; it uses context-fitted, query-evaluated Euclidean kNN with $k = 5$ and distance weights.

Native class-relabeling sensitivity. Before wrapping any checkpoint in OvA, Table 8 reports how far each native head already is from class equivariance: on the 49-dataset, five-seed, three-trial grid, relabeling classes changes TabPFNv2's hard-label predictions on 3.3% of queries and TabICLv2's on 2.3%, even though column agreement is high for both (98.2%). This is the gap OvA is meant to close.

OvA results across all three families. Table 9 gives OvA results for all three released families. TabPFNv2 and both TabICLv2 releases are evaluated on 21 multiclass datasets with five seeds and

Table 6: Late TabPFNv2 L9 attention intervention.

Scope	Native accuracy	Constant-output accuracy
10 datasets × five splits	.874	.488

Table 7: TabPFNv2 representation comparison.

Input-space kNN	L9 kNN	L11 kNN	Native
.772	.745	.549	.855

three column- and three class-permutation trials each; wrapping either backbone in OvA leaves accuracy at or above its native value and reaches exact hard-label agreement under class recoding on every dataset ($21/21$). Mitra v1 is evaluated separately on a 24-dataset multiclass cohort with the same trial protocol: OvA leaves accuracy essentially unchanged (.841 versus .841 for a separately run native baseline, unrounded .841321 versus .840909), with hard-label agreement of .998731 under feature-column permutation and .996039 under class recoding. Mitra falls just short of the other two on label agreement, which we attribute to tie-breaking and floating-point ties in the wrapper rather than to a symmetry failure, since these numbers are empirical agreement on this cohort, not a tie-safe hard-label guarantee.

Table 10 uses ten datasets with five random 80/20 splits. Representation-based kNN readouts are fit on context block 12 representations and evaluated on held-out queries.

Table 11 uses the same 49-dataset × five-seed split grid as the classification readouts. We compute expected calibration error (ECE) with 15 bins for each dataset and seed, then average across seeds and datasets.

Table 12 reports accuracy with row or column attention disabled separately; all settings use three fixed 70/30 splits.

Table 13 reports one fixed evaluation on each of 49 datasets; entries after clean are percentage-point accuracy changes.

A.3 Column-invariance edits on the released checkpoints

Each released classifier's feature-position device can be switched off at inference, without retraining, and the result measured on the same 49-dataset grid with five seeds and three column-permutation trials (Table 14). TabPFNv2 maps each feature index through a learned weight matrix $W \in \mathbb{R}^{192\times 48}$ and adds a learned bias $b \in \mathbb{R}^{192}$. Setting $W{=}0$ while retaining b shifts predictions by at most .001 and leaves mean accuracy at .854, and the edited model is exactly invariant to permutations within and among its feature groups; replacing b by $\mathrm{sign}(b)$ is equally lossless, whereas setting $b{=}0$ collapses accuracy to .338, so b acts as an architectural prior rather than a positional code.

Removing feature RoPE from TabICLv2 gives exact invariance to circular column shifts at a mean cost of .1 pp, with no dataset losing more than 5 pp. The same edit on TabICL v1 gives exact arbitrary-column invariance but costs 15.4 pp, and 27 of 49 datasets lose more than 5 pp; v1 has no circular grouping to fall back on, so RoPE is its only symmetry-breaking route. The worst individual losses are on the *mfeat* family, *semeion*, *MiceProtein*, and *balance-scale*, each between 49 and 58 pp. The gap between the two versions upper-bounds what RoPE buys when it is the sole such route.

Table 8: Native (non-OvA) permutation and class-recoding agreement, 49 datasets × five seeds × three trials.

Model	Accuracy	Column agree	Label agree
TabPFNv2	.854	98.2%	96.7%
TabICLv2	.865	98.2%	97.7%

Table 9: One-vs-all wrapper results across all three released families. Column and label agree are hard-label agreement rates under column permutation and class recoding, respectively; TabPFNv2 and TabICLv2 use a 21-dataset multiclass cohort, Mitra a 24-dataset cohort (Appendix A.2).

Model+OvA	Accuracy	Column agree	Label agree
TabPFNv2+OvA	.877	98.4%	100% (21/21)
TabICL v1+OvA	.880	97.2%	100% (21/21)
TabICLv2+OvA	.890	98.3%	100% (21/21)
Mitra v1+OvA	.841	99.9%	99.6%

Table 10: TabICL regression readout checks.

Method	R^2	Pearson with native prediction
Native TabICL-reg	.805	—
Block 12 representation kNN-5	.577	.859
Block 12 representation kNN-10	.566	.871
Input-space kNN-10	.433	—
Block 10 reconstructed vote	.064	.732

Table 11: Readout accuracy and calibration.

Backbone	Readout	Accuracy	ECE	NLL
TabPFNv2	native	.854	.043	.329
TabPFNv2	vote	.804	.201	.625
TabPFNv2	prototype	.523	.277	1.178
TabPFNv2	kNN-5	.547	.371	9.715
TabICLv2	native	.864	.042	.303
TabICLv2	prototype	.854	.463	1.058
TabICLv2	kNN-5	.856	.115	2.840
TabICLv2	attention-sharpness vote	.470	.125	1.094

Table 12: Targeted collapse checks.

Check	Full	Row off	Column off	Majority
Mitra v1, balance	.954	.080	.459	.463
Mitra v1, balance-like d=12	.893	.074	.459	.474
Mitra v1.1, balance	.963	.080	.461	.463
Mitra v1.1, balance-like d=12	.969	.074	.467	.474

Table 13: Mitra perturbation checks.

Release	Clean	Hub poison (Δ pp)	Cube (Δ pp)	Rank (Δ pp)
v1	.848	-2.07	-0.01	-0.02
v1.1	.849	-2.20	-0.13	-0.25

Table 14: Column-invariance edits applied to the released checkpoints, 49 datasets $\times$ five seeds $\times$ three column-permutation trials.

Model	Edit	Accuracy	Δ pp	Residual symmetry
TabPFNv2	W=0, keep b	.854	0.0	Exact within/among feature groups
TabPFNv2	W=0, b=0	.338	-51.6	Exact, but the model breaks
TabICL v1	No feature RoPE	.697	-15.4	Exact arbitrary column
TabICLv2	No feature RoPE	.862	-0.1	Exact circular shift

A.4 STAGE-1 PRETRAINING AND EVALUATION

The stage-1 study covers 13 locally trained TabICLv2 variants (Qu et al., 2026), including a reference trained from scratch. The reference run uses a graph-SCM prior, 500,000 updates, global batch size 64, and one pretraining seed (both NumPy and Torch seed 42). The twelve-block reference is accompanied by 1-, 3-, and 6-block ICL trunks; two feature-axis variants; two jointly trained readout twins; and five label-pathway variants.

Table 15: Stage-1 training recipe.

Setting	Reference and variants
Synthetic tasks	Cached graph-SCM mixture
Tables	Generator ranges: at most 1,024 total context-and-query rows, 1–100 features, 2–10 classes, and context fraction .3–.9
Architecture	Width 128; 3 column and 3 row blocks, 4 CLS tokens, and 12 ICL blocks
Optimizer	Muon ($\beta_1 = .9$)
Learning rate	Peak $8\mathrm{e}{-4}$, cosine decay to 10^{-7}; weight decay .01
Schedule	500,000 updates; batch 64; 1% warmup; gradient clip 10. The warm start adds 500,000 updates after reference initialization
Arithmetic and hardware	float32 on four H100 GPUs

The common evaluation contains 38 TabArena datasets (Erickson et al., 2025): 30 binary tasks summarized by AUC and 8 multiclass tasks summarized by log loss. The cohort is defined by common dataset–model records in the final exports. Each checkpoint uses one estimator ($n_{\text{estimators}} = 1$), and fold–repeat evaluations are averaged within each task. The metrics are not pooled. Uncertainty and win/tie/loss counts use matched within-dataset test differences, they describe dataset variation.

Across all 13 models, the analysis retains 7722 held-out evaluation rows. Table 16 reports macro AUC over the 30 binary tasks, macro log loss over the 8 multiclass tasks, and mean within-task ranks over all 38 tasks, with tied ranks averaged. Its Elo column fits Bradley–Terry ratings to pairwise task wins across all 13 models: lower error wins and exact ties count as half a win. The reference is anchored at 1,000 with base-10 scale 400. Rank and Elo intervals use 200 resamples of datasets, not pretraining seeds.

Figure 1A uses a three-way Elo fit restricted to the reference, prototype, and vote heads; Figure 1B uses only the reference and fully invariant row–column–class model. Figure 4's left panel is a global 13-model Elo fit. Its right panel compares each candidate directly with the reference: with task win fraction p (ties count as half), it reports $400 \log_{10}[p/(1-p)]$. Its 95% interval takes the corresponding quantiles of p across 200 dataset bootstraps before the monotone transformation, rather than subtracting global ratings. Mean ranks appear in Table 16.

The trained vote twin computes learned-temperature dot-product attention over context representations and averages their one-hot labels. The prototype twin uses negative squared Euclidean distance to context class prototypes. Both are jointly trained with the encoder, unlike the frozen native-attention probe in §2. Positional ablations concern feature and CLS slots within each row, not context-row order; feature grouping is a separate feature-position choice.

For the separate Stage-1 TabICL permutation checks, the wrapper uses one estimator, no numerical normalization methods, and no feature or class shuffles. The three earlier label-pathway variants alter the ICL-trunk label path but retain a class-indexed target-aware column channel, so they are not end-to-end class equivariant. Table 19 reports substantial maximum probability discrepancies under class recoding despite small tested-column discrepancies; the largest checked context-row difference is $3.6\mathrm{e}{-6}$.

The full invariant and warm-start variants use the revised label path. The warm start begins from the reference's 500,000-step weights and then trains for a further 500,000 steps, so it is not compute matched with the reference.

Figure 5 displays all 13 training curves. They show training-batch cross-entropy, not validation or held-out loss, and are not directly comparable when heads use different objectives.

Table 16: Stage-1 local-checkpoint evaluation.

Variant	Elo ↑	Rank ↓	AUC ↑	Log loss ↓	W/T/L
No RoPE or grouping	1066.0	5.29	0.8482	0.2499	22/0/16
Vote head	1059.6	5.39	0.8490	0.2498	22/0/16
Fully invariant	1053.4	5.50	0.8477	0.2492	20/0/18
Prototype head	1040.9	5.71	0.8466	0.2481	19/0/19
No feature RoPE	1009.0	6.26	0.8487	0.2498	23/0/15
Reference (12 blocks)	1000.0	6.42	0.8455	0.2491	—
Log-scaled class bias	985.1	6.68	0.8465	0.2494	17/0/21
Fully invariant, warm start*	973.1	6.89	0.8468	0.2526	19/0/19
No trunk label injection	935.8	7.55	0.8438	0.2611	15/0/23
Same-class attention bias	935.8	7.55	0.8448	0.2528	16/0/22
6 ICL blocks	894.7	8.26	0.8451	0.2550	15/0/23
1 ICL block	803.6	9.71	0.8424	0.2596	9/0/29
3 ICL blocks	800.0	9.76	0.8335	0.2588	9/0/29

* *Warm start receives an additional 500,000 pretraining updates after initialization from the reference.*

A.5 A RETAINED-CELL KERNEL CONSTRUCTION

The construction retains a grid of aligned cells rather than first compressing a table into row tokens. Shared attention exchanges information within feature columns and across the cells of a row; context rows and class summaries do not read query rows, and queries do not read one another. The result is one vector $\mathbf{Z}_{rj}$ for each cell, with r ranging over context rows, query rows, and class summaries. Numerical cells retain their raw values, while categorical information enters only through equality and missingness. Class summaries begin identically and are linked to their labeled context members. All transformations are shared and position-free.

A single final label-attention head uses

$$\begin{aligned} \Delta(\mathbf{q}, \mathbf{x}_i) &= \sum_{j=1}^{d} \|\mathbf{Z}_{qj} - \mathbf{Z}_{ij}\|_2^2 + \sum_{j \in \mathcal{C}} \mathbf{1}\{q_j \neq x_{ij}\}, \\ W_i(\mathbf{q}) &= \mathrm{softmax}_i[-\Delta(\mathbf{q}, \mathbf{x}_i)/h^2], \qquad \mathbf{P}(\mathbf{q}) = \mathbf{W}(\mathbf{q})\mathbf{Y}. \end{aligned} \tag{4}$$

Here $\mathbf{Y}$ is the context one-hot label matrix. Thus the only final readout is a weighted sum of context labels: there is neither an extra classifier nor a second softmax. Keeping aligned cells makes the feature comparison in (4) well defined.

In a fixed parameter setting, the learned cell and contextual updates and the categorical, type, and role vectors can all be zeroed. Numerical cells then retain their identity coordinate and categorical cells contribute only their equality tests, so the same network becomes the normalized mixed kernel average

$$\begin{aligned} k_h(\mathbf{x}, \mathbf{z}) &= \exp\left[-\frac{\sum_{j \in \mathcal{N}} (x_j - z_j)^2 + \sum_{j \in \mathcal{C}} \mathbf{1}\{x_j \neq z_j\}}{h^2} \right], \\ p_c(\mathbf{q}; D) &= \frac{\sum_{i=1}^{n} k_h(\mathbf{q}, \mathbf{x}_i) \mathbf{1}\{y_i = c\}}{\sum_{i=1}^{n} k_h(\mathbf{q}, \mathbf{x}_i)}. \end{aligned} \tag{5}$$

This is Nadaraya–Watson label averaging (Nadaraya, 1964; Watson, 1964). The same attention template admits learned contextual features away from this setting, but Proposition 1 certifies only the stated setting. Arbitrary learned query–key scores need not be positive semidefinite (Han et al., 2025; Miftachov et al., 2026).

For every parameter setting of this shared, position-free construction, permuting context rows, joint feature columns, class names, or nominal codes permutes the intermediate cell states, class summaries, and equality-based terms in the same way. Induction over the attention blocks therefore gives the corresponding permutation of $\mathbf{Z}$. The final sum is invariant to row, column, and code permutations and has a permuted probability vector under a class renaming. If a hard class label is required, uniform randomized tie breaking preserves class equivariance in distribution.

Table 17: Dataset-level scores: depth and readouts.

Dataset	Task ID	Ref.	D1	D3	D6	Proto.	Vote	NoPE
Binary: AUC ↑								
Amazon employee access	363613	0.8465	0.8363	0.8127	0.8358	0.8403	0.8388	0.8403
APSFailure	363616	0.9931	0.9910	0.9919	0.9928	0.9931	0.9936	0.9931
bank-marketing	363618	0.7556	0.7611	0.7557	0.7157	0.7538	0.7626	0.7584
Bank Customer Churn	363619	0.8713	0.8643	0.8671	0.8673	0.8687	0.8671	0.8679
Bioresponse	363620	0.8569	0.8224	0.7512	0.8589	0.8670	0.8388	0.8746
Blood transfusion	363621	0.7580	0.7535	0.7560	0.7584	0.7548	0.7552	0.7541
churn	363623	0.9279	0.9168	0.9237	0.9282	0.9271	0.9325	0.9292
Coil2000 insurance	363624	0.7522	0.7722	0.6810	0.7565	0.7576	0.7580	0.7672
credit-g	363626	0.7945	0.7930	0.7926	0.7964	0.7962	0.7944	0.7942
Credit-card default	363627	0.7860	0.7805	0.7844	0.7862	0.7872	0.7860	0.7878
customer satisfaction in airline	363628	0.9940	0.9916	0.9924	0.9932	0.9935	0.9944	0.9941
diabetes	363629	0.8385	0.8381	0.8385	0.8379	0.8397	0.8396	0.8388
Diabetes130US	363630	0.6483	0.6493	0.5740	0.6229	0.6241	0.6390	0.6374
E-CommereShippingData	363632	0.7421	0.7404	0.7407	0.7454	0.7412	0.7412	0.7443
Fitness Club	363671	0.8208	0.8193	0.8199	0.8199	0.8205	0.8210	0.8212
GiveMeSomeCredit	363673	0.8492	0.8626	0.8576	0.8604	0.8634	0.8643	0.8626
Hazelnut contamination	363674	0.9941	0.9815	0.9898	0.9934	0.9945	0.9939	0.9945
heloc	363676	0.8014	0.7987	0.7987	0.7981	0.8017	0.8001	0.7985
HR analytics / job change	363679	0.8033	0.8016	0.8037	0.8041	0.8048	0.8030	0.8019
Vehicle coupon recommendation	363681	0.8131	0.8038	0.8057	0.8004	0.7965	0.8151	0.8052
Is-this-a-good-customer	363682	0.7495	0.7478	0.7498	0.7465	0.7465	0.7467	0.7497
jm1	363712	0.7291	0.7331	0.7447	0.7579	0.7213	0.7684	0.7542
kddcup09 appetency	363683	0.8154	0.7898	0.7361	0.7992	0.8100	0.8180	0.8177
Marketing Campaign	363684	0.9214	0.9089	0.9204	0.9189	0.9207	0.9236	0.9178
NATICUSdroid	363689	0.9870	0.9852	0.9705	0.9868	0.9871	0.9874	0.9874
online shoppers intention	363691	0.9349	0.9326	0.9331	0.9354	0.9357	0.9352	0.9344
Polish company bankruptcy	363694	0.9728	0.9365	0.9602	0.9645	0.9776	0.9799	0.9774
qsar-biodeg	363696	0.9398	0.9392	0.9416	0.9397	0.9416	0.9426	0.9405
seismic-bumps	363700	0.7229	0.7785	0.7687	0.7859	0.7866	0.7814	0.7726
Taiwan bankruptcy	363706	0.9463	0.9434	0.9425	0.9463	0.9442	0.9479	0.9442
Multiclass: log loss ↓								
anneal	363614	0.0192	0.0106	0.0209	0.0212	0.0123	0.0161	0.0135
hiva agnostic	363677	0.1946	0.1832	0.1938	0.2139	0.1931	0.1847	0.2029
maternal health risk	363685	0.3992	0.4544	0.4312	0.4141	0.4017	0.3978	0.3986
MIC	363711	0.4417	0.4335	0.4495	0.4411	0.4383	0.4411	0.4443
SDSS17	363699	0.0771	0.0868	0.0819	0.0769	0.0746	0.0865	0.0735
splice	363702	0.0999	0.1063	0.1007	0.1031	0.1000	0.1008	0.1020
Student dropout / success	363704	0.5377	0.5525	0.5441	0.5442	0.5406	0.5423	0.5366
website phishing	363707	0.2237	0.2499	0.2480	0.2255	0.2245	0.2290	0.2270

Ref.: reference; D1/D3/D6: ICL depth; Proto./Vote: jointly trained heads; NoPE: no feature RoPE.

Table 18: Dataset-level scores: symmetry variants.

Dataset	Task ID	Ref.	NoPE/G	None	Bias	Log	Inv.	Warm*
Binary: AUC ↑								
Amazon employee access	363613	0.8465	0.8397	0.8302	0.8343	0.8423	0.8282	0.8499
APSFailure	363616	0.9931	0.9942	0.9935	0.9927	0.9937	0.9910	0.9933
bank-marketing	363618	0.7556	0.7634	0.6891	0.7533	0.7424	0.7620	0.7617
Bank Customer Churn	363619	0.8713	0.8716	0.8626	0.8664	0.8677	0.8706	0.8641
Bioresponse	363620	0.8569	0.8696	0.8707	0.8747	0.8713	0.8340	0.8156
Blood transfusion	363621	0.7580	0.7552	0.7552	0.7524	0.7554	0.7530	0.7543
churn	363623	0.9279	0.9275	0.9312	0.9332	0.9237	0.9328	0.9309
Coil2000 insurance	363624	0.7522	0.7465	0.7628	0.7477	0.7567	0.7606	0.7667
credit-g	363626	0.7945	0.7971	0.7991	0.7971	0.7954	0.7984	0.7940
Credit-card default	363627	0.7860	0.7866	0.7853	0.7861	0.7835	0.7869	0.7868
customer satisfaction in airline	363628	0.9940	0.9944	0.9930	0.9943	0.9930	0.9945	0.9940
diabetes	363629	0.8385	0.8386	0.8405	0.8395	0.8396	0.8410	0.8384
Diabetes130US	363630	0.6483	0.6274	0.6419	0.6072	0.6103	0.6218	0.6272
E-CommereShippingData	363632	0.7421	0.7439	0.7378	0.7425	0.7452	0.7460	0.7402
Fitness Club	363671	0.8208	0.8216	0.8211	0.8211	0.8214	0.8212	0.8209
GiveMeSomeCredit	363673	0.8492	0.8633	0.8590	0.8455	0.8609	0.8631	0.8628
Hazelnut contamination	363674	0.9941	0.9939	0.9945	0.9947	0.9943	0.9947	0.9945
heloc	363676	0.8014	0.8013	0.8017	0.8006	0.8000	0.7983	0.8009
HR analytics / job change	363679	0.8033	0.8046	0.8025	0.8025	0.8021	0.8025	0.8041
Vehicle coupon recommendation	363681	0.8131	0.7987	0.8058	0.8023	0.8006	0.8026	0.7887
Is-this-a-good-customer	363682	0.7495	0.7468	0.7461	0.7437	0.7438	0.7486	0.7471
jm1	363712	0.7291	0.7623	0.7546	0.7475	0.7623	0.7735	0.7741
kddcup09 appetency	363683	0.8154	0.8030	0.8148	0.8008	0.7988	0.8178	0.8118
Marketing Campaign	363684	0.9214	0.9210	0.9197	0.9189	0.9201	0.9208	0.9162
NATICUSdroid	363689	0.9870	0.9872	0.9869	0.9874	0.9872	0.9869	0.9869
online shoppers intention	363691	0.9349	0.9353	0.8762	0.9348	0.9346	0.9365	0.9361
Polish company bankruptcy	363694	0.9728	0.9794	0.9798	0.9794	0.9803	0.9745	0.9785
qsar-biodeg	363696	0.9398	0.9408	0.9405	0.9399	0.9411	0.9410	0.9397
seismic-bumps	363700	0.7229	0.7847	0.7758	0.7595	0.7823	0.7856	0.7834
Taiwan bankruptcy	363706	0.9463	0.9464	0.9434	0.9443	0.9458	0.9429	0.9424
Multiclass: log loss ↓								
anneal	363614	0.0192	0.0131	0.0138	0.0138	0.0146	0.0120	0.0117
hiva agnostic	363677	0.1946	0.1913	0.2769	0.2001	0.2009	0.1840	0.2243
maternal health risk	363685	0.3992	0.4082	0.4021	0.4016	0.3948	0.4015	0.3925
MIC	363711	0.4417	0.4417	0.4421	0.4404	0.4376	0.4507	0.4487
SDSS17	363699	0.0771	0.0746	0.0831	0.0973	0.0791	0.0731	0.0740
splice	363702	0.0999	0.1031	0.1011	0.1000	0.1005	0.1069	0.1080
Student dropout / success	363704	0.5377	0.5415	0.5436	0.5421	0.5392	0.5414	0.5357
website phishing	363707	0.2237	0.2255	0.2261	0.2268	0.2289	0.2244	0.2257

NoPE/G: no RoPE or grouping; None/Bias/Log: trunk-label variants; Inv.: fully invariant (rows, columns, classes); Warm: its warm start. *Additional pretraining.

Table 19: Probability changes under permutations.

Variant	Column maximum	Class maximum
Reference (12 blocks)	0.0029	0.0042
No feature RoPE	0.0027	0.0096
No RoPE or grouping	3.9×10^{-7}	0.065
No trunk label injection	1×10^{-6}	0.17
Same-class attention bias	5.7×10^{-7}	0.074
Log-scaled class bias	3.9×10^{-7}	0.031
Fully invariant	$\leq 1 \times 10^{-6}$	$\leq 1 \times 10^{-6}$

The fully invariant bound is the worst case of the joint row, column, and class check run before its pretraining launch, not a row of the post-hoc sweep above.

Proposition 2 (Four symmetries and finite-context separation)**.** *The predictor in* (5) *is invariant to context-row permutations and joint feature-column permutations, equivariant to a permutation of class names, and invariant to independent bijective recoding of each categorical column. If context rows are distinct, sufficiently small h makes its argmax classify every context row as its observed label.*

Proof. The transformations respectively reorder the sums, reorder the terms in the kernel distance, permute the one-hot output coordinates, and preserve every categorical equality test. Let δ be the smallest positive mixed squared distance between two context rows. At $\mathbf{q} = \mathbf{x}_i$, the self weight is one and all other unnormalized weights total at most $(n-1)e^{-\delta/h^2}$. For sufficiently small h, the correct class weight strictly exceeds the combined weight of all other classes. The one-row case is immediate. □

Proposition 2 gives self-classification on a finite distinct context, not a generalization result. It does not transfer to an arbitrary compressed representation merely because that model has the same named symmetries.

A.6 Consistency of the exact kernel

Proof of Proposition 1. Write μ for the input distribution, $\boldsymbol{\eta}(\mathbf{x}) = \mathbb{E}[\mathbf{Y} \mid \mathbf{X} = \mathbf{x}]$ for the posterior with one-hot label vector $\mathbf{Y}$, and $a_h(\mathbf{x}) = \int k_h(\mathbf{x}, \mathbf{z}) d\mu(\mathbf{z})$. Split the compact numerical box into cubes of side $h/\sqrt{p}$ and categorical tuples. There are $N_h = O(Mh^{-p})$ cells, where M is the finite number of categorical tuples; within a cell $k_h \geq e^{-1}$. Hence $\int a_h^{-1} d\mu \leq eN_h$. At a fixed query put $D = \sum_i k_h(\mathbf{x}, \mathbf{X}_i)$, $\mathbf{t} = T_h\boldsymbol{\eta}(\mathbf{x})$, and $\mathbf{S} = \sum_i k_h(\mathbf{x}, \mathbf{X}_i)(\mathbf{Y}_i - \mathbf{t})$. Then $\mathbb{E}\|\mathbf{S}\|_2^2 \leq 4na_h(\mathbf{x})$. On $D \geq na_h(\mathbf{x})/2$, the normalized squared error contributes at most $16/[na_h(\mathbf{x})]$; Chebyshev bounds the complement by $4/[na_h(\mathbf{x})]$, where the squared error is at most two. Integrating gives

$$\mathbb{E}\|\widehat{\boldsymbol{\eta}}_{n,h} - T_h\boldsymbol{\eta}\|_{L^2(\mu)}^2 \leq \frac{24eN_h}{n} = O\left(\frac{M}{nh^p}\right), \qquad T_h f(\mathbf{x}) = \frac{\int k_h(\mathbf{x}, \mathbf{z}) f(\mathbf{z}) d\mu(\mathbf{z})}{a_h(\mathbf{x})}.$$

For the bias, each positive-mass bin contributes at most $e \sup_{\mathbf{x} \in B} k_h(\mathbf{x}, \mathbf{z})$ to $\int k_h(\mathbf{x}, \mathbf{z})/a_h(\mathbf{x}) d\mu(\mathbf{x})$. Summing the Gaussian lattice yields a finite constant $K_{p,M}$, independent of h and $\mathbf{z}$. Jensen therefore makes T_h uniformly bounded in $L^2(\mu)$. Uniform continuity and the Gaussian tail give convergence to the identity on continuous functions; their density in $L^2(\mu)$ gives $\|T_h\boldsymbol{\eta} - \boldsymbol{\eta}\|_{L^2} \to 0$ for every measurable posterior. The stated schedule has $h_n \to 0$ and $nh_n^p \to \infty$, so the sampling and bias terms vanish. Finally, conditional excess classification error is at most $2\|\widehat{\boldsymbol{\eta}}_{n,h}(\mathbf{x}) - \boldsymbol{\eta}(\mathbf{x})\|_2$; integration and Cauchy–Schwarz prove the result. When $p = 0$, the finite categorical tuples give the same argument directly. □

Proposition 1 concerns growing iid context and the exact kernel setting. It neither guarantees every finite query nor says that pretraining finds this setting. It establishes that aligned-cell attention, the four symmetries, and statistical consistency are compatible in one shared forward model.

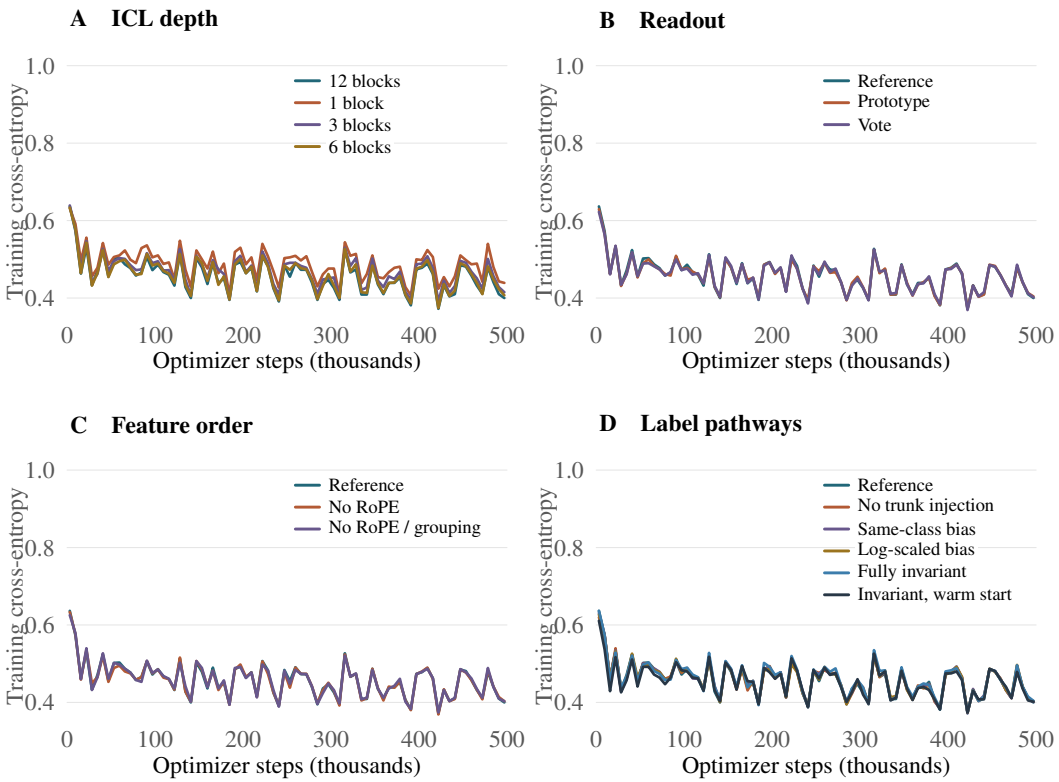


Figure 5: Stage-1 training-batch cross-entropy. Curves are medians of bins of about 25 retained log samples; the .35–1.0 vertical range omits earlier, higher loss. Panels A–D use the same 0–500k optimizer-step range.