TABULAR SELECTIVE ATTENTION

Anonymous authors
Paper under double-blind review

ABSTRACT

Tabular foundation models (TFMs) predict each test row by attending over a context of labelled training rows, so a prediction is only as good as the set of rows attention averages over. While these models perform well in low-data regimes, current state-of-the-art models employ context curation to generalize over larger datasets. Existing curation methods either sharpen attention with a temperature or shrink the context with nearest-neighbour retrieval. We prove that temperature scaling faces an inherent tradeoff: a single scalar must balance suppressing the uninformative tail against preserving the attention mass over the entire informative set. Retrieval avoids this tradeoff by explicitly selecting *which* samples enter the context in the first place. Yet in doing so, it uses row similarity as a proxy for the attention that the model would assign to each row. We address the limitations of both curation types with a new model-agnostic mechanism: **Tabular Selective Attention (TSA)**. TSA learns a non-negative score for each test/training-row pair that reshapes the attention distribution into one that favours *all* informative samples rather than just a subset. Across different architectures, such as TabICLv2 and TabDPT, TSA outperforms existing curation methods on the TabArena benchmark. Leveraging these TSA scores to prune uninformative rows from context, we can speed up its inference while largely preserving accuracy.

1 INTRODUCTION

Tabular data is the most common modality in applied machine learning, but it is also heterogeneous (Borisov et al., 2024). Gradient-boosted decision trees (Chen & Guestrin, 2016; Ke et al., 2017; Prokhorenkova et al., 2018) remained the default for tabular prediction even as neural networks surpassed classical methods in other modalities (Shwartz-Ziv & Armon, 2022). Yet with any of these models, every new dataset requires its own training run and hyperparameter search. Tabular foundation models (TFMs) (Hollmann et al., 2023) eliminate this per-dataset cost by pre-training over a distribution of tables and then leveraging in-context learning (ICL): they take a labelled support set as context and predict in a single forward pass, without gradient updates. Despite requiring no per-dataset optimization, recent TFMs achieve state-of-the-art performance on small- to medium-sized tables (Grinsztajn et al., 2026; Qu et al., 2026; Zhang et al., 2026).

At inference, within each attention head, a test row scores the n training rows and aggregates their values with softmax-normalized weights $\mathrm{Attn}(Q, K, V) = \mathrm{softmax}\big(QK^\top/(\tau\sqrt{d_k})\big)V$. The temperature τ divides every logit before normalization, and so determines how sharply the weights concentrate. Standard attention fixes $\tau = 1$, and the $\sqrt{d_k}$ factor stops dot-product magnitudes from growing with the key dimension (Vaswani et al., 2017) but does not correct for context length. As the number of training rows n grows, the softmax denominator accumulates over more samples and hence the distribution flattens, an effect known as *attention fading* (Nakanishi, 2025). This matters because tabular ICL appears to be largely retrieval-based: a TFM learns to identify the rows relevant to a test row and aggregate their labels (Shaheen et al., 2026). For a given test row, we call the rows a head retrieves its *informative* rows and the rest its *uninformative* tail. This split depends on both the test row and the head, and is graded rather than binary. Fading shifts mass from the informative rows onto the tail, increasingly so as n grows. Qu et al. (2026) identifies this as the length-generalization failure in tabular ICL. Consequently, for TFMs, running inference on datasets larger than the context they were pre-trained on required some form of *context curation*: a mechanism deciding how much each training row can influence a given test prediction. We identify two context curation approaches: (1) temperature scaling, and (2) nearest-neighbour retrieval.

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.

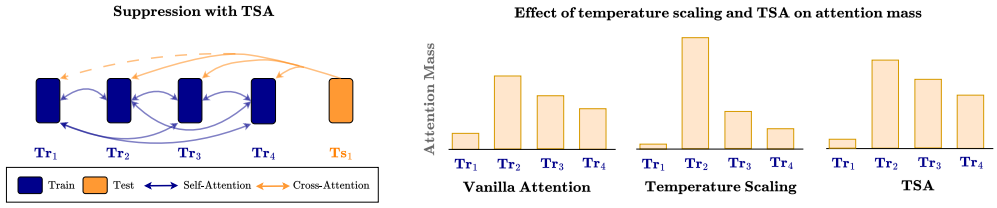

Figure 1: TSA suppresses uninformative rows. **Left:** Training rows attend to each other (*self-attention*), and the test row attends only to training rows (*cross-attention*). TSA allows Ts_1 to suppress the uninformative row Tr_1 (dashed). **Right:** Temperature scaling suppresses Tr_1 but also reshapes the relative weights betweenTr_2, Tr_3 and Tr_4. TSA suppresses only Tr_1 and keeps the relative weights of the informative rows.

Temperature scaling includes every row in the context and lowers τ as n grows, so that the attention distribution stays sharp, as in SSMAX (Nakanishi, 2025) and its query-dependent tabular extension QASSMAX (Qu et al., 2026). Ideally, however, a head would do two things at once: keep attention weight off the uninformative rows, and read from the *whole* informative set rather than only its top-scoring row. Tabular labels are noisy, so a head that collapses onto its top-scoring rows inherits these rows' label noise, whereas averaging over the informative set reduces it. Because lowering the temperature divides every logit by the same number, it cannot suppress the uninformative rows without also stripping mass from the weakest informative ones. Section 3 shows that, in an ideal setting, this trade-off is controlled by a ratio of logit gaps γ that temperature cannot change. This trade-off vanishes in the needle-in-a-haystack setting ($\gamma \to 0$), where QASSMAX was shown to work, but is expensive when tabular predictions must aggregate over many informative rows.

Nearest-neighbour retrieval instead limits the context to the k training rows nearest to each test row, for a fixed k (Thomas et al., 2024; Ma et al., 2026). This requires a per-dataset index and a separate forward pass for every test row, which makes it at least $100\times$ slower than full-context inference (Hosseinzadeh et al., 2026). Moreover, retrieval uses row *similarity* as a proxy for *informativeness*, and since the ranking is computed outside the model, it need not match the rows the model's attention would select. It also provides a single neighbour set for each test row that every head must share, although each head has its own informative set in the full context. We therefore ask for a **learnable notion of informativeness**, one that leverages the attention maps the model actually learns.

In this work, we present **Tabular Selective Attention (TSA)**, a context curation mechanism applied as a drop-in modification of the attention blocks in TFMs. In each head, TSA learns a non-negative score for every test/training-row pair and subtracts it from the pre-softmax attention logits. Thus, a test row suppresses individual training rows instead of reweighting all of them at once (Figure 1, left). Unlike a temperature, this score is key-dependent and can therefore distinguish a weakly informative row from an uninformative one (Figure 1, right). When the score is zero on the informative set, suppression acts on the tail alone and leaves the relative weights of the informative rows unchanged. Unlike retrieval, the scores are learned inside each head's attention and applied in a single forward pass over the full context. The suppression scores also identify rows that test queries consistently mark as uninformative. Ranking training rows by their mean suppression across heads and test rows lets the highest-scoring ones be dropped from the context outright, at a cost controlled by the attention they were receiving.

Our contributions are as follows:

- We show that, under idealized separability of informative and uninformative rows, key-independent temperature scaling, SSMAX and QASSMAX included, cannot suppress the uninformative rows without also stripping attention from weakly informative ones.
- We introduce **Tabular Selective Attention (TSA)**, an implicit context curation mechanism whose key-dependent suppression leaves the relative weights of the informative rows unchanged however strongly the tail is suppressed.
- We show that the same scores rank training rows by droppability, giving a pruning rule that shrinks the context during the forward pass and speeds up inference on large tables.

- On TabArena classification, TSA achieves the best improvability, average rank and Elo among attention variants in an identical TabICLv2 backbone, with a 66.1% win rate against vanilla attention and outperforming QASSMAX, TabICLv2's default. It also transfers to TabDPT, where it ranks first on datasets with more than 10K training samples.

2 RELATED WORK

Tabular Foundation Models (TFMs). TFMs, based on prior-data fitted networks (PFNs) (Müller et al., 2022), emerged as a new paradigm in tabular deep learning. Pre-trained on synthetic datasets sampled from a prior $p(\mathcal{D})$, a TFM q_θ predicts $q_\theta(y_{\text{test}} \mid x_{\text{test}}, \mathcal{D}_{\text{train}})$ with a single forward pass through in-context learning (ICL) without gradient updates. TabPFN (Hollmann et al., 2023) treats each row as a separate token and performs ICL over rows, while TabPFNv2 (Hollmann et al., 2025) and TabPFN-2.5 (Grinsztajn et al., 2025) treat cells as tokens with alternating row and column attention layers. To improve computational efficiency, TabICL models (Qu et al., 2025; 2026) propose a three-stage approach: a column-then-row attention module first constructs fixed-dimensional row embeddings, after which ICL is performed over these embeddings. This design is now also adopted in TabPFN-3 (Grinsztajn et al., 2026). Other recent TFMs include LimiX (Zhang et al., 2025; Wang et al., 2026), TabDPT (Ma et al., 2026), TabFM (Kong & Das, 2026) and TabSwift (Liu & Ye, 2026).

Addressing *attention fading*. Veličković et al. (2025) show that, with bounded attention logits, softmax attention in Transformers (Vaswani et al., 2017) must disperse as the number of context items grows, even if it is sharp at the lengths seen during pre-training. Nakanishi (2025) calls this *attention fading* and counters it with Scalable-Softmax (SSMAX), which multiplies the logits by $s \log n$, where s is a learnable parameter per-head. Chen et al. (2026) also proves that $\log n$ scaling rate is critical to prevent attention collapse. TabICLv2 (Qu et al., 2026) adopts this to TFMs with QASSMax by combining a learned function of $\log n$ with a query-aware gate. Adaptive-Scalable Entmax (ASEntmax) (Vasylenko et al., 2026; Peters et al., 2019) extends sparse projections with content-aware logarithmic scaling to adapt sparsity thresholds dynamically. Closest to our method is Selective Attention (Leviathan et al., 2025), a parameter-free method for language models in which a token can mark an earlier token as no longer needed once it has absorbed that token's content. Future tokens then pay less attention to the marked token, which reduces redundancy. Tokens deemed redundant can also be dropped from the context, saving memory at inference.

Scaling TFMs for larger tables. When scaling to large tables, Transformer-based TFMs face, in addition to attention fading, the quadratic $\mathcal{O}(n^2)$ cost of attention in the number of rows n. Compression methods shrink what is being attended to: MotherNet (Mueller et al., 2025) and HyperFast (Bonet et al., 2024) map the training set directly to the parameter space of a compact child network in a single forward pass. Retrieval methods (Thomas et al., 2024; Ma et al., 2026) use the nearest training rows to each test row as context, but their cost grows with the number of queries (Hosseinzadeh et al., 2026). Other work modifies attention itself: TabFlex (Zeng et al., 2025) uses non-causal linear attention (Katharopoulos et al., 2020), TabSwift (Liu & Ye, 2026) adds gated attention and learnable register tokens, similar to the "thinking rows" of TabPFN-2.5 (Grinsztajn et al., 2025), and TabICLv2 (Qu et al., 2026) adapts SSMAX to TFMs as QASSMAX, combining a learned function of $\log n$ with a query-aware gate. Unlike compression and retrieval, our method does not pre-filter the context. Instead, the model's own attention decides which rows to suppress and drop.

3 LIMITATIONS OF TEMPERATURE SCALING IN TFMS

In tabular ICL models, as the number of training rows n grows, softmax attention spreads and the distribution flattens (Veličković et al., 2025). Temperature scaling counters this attention fading by multiplying the logits by a positive scalar $1/\tau$ before normalization:

$$\text{Attn}_\tau(Q, K, V) \;=\; \text{softmax}\left(\frac{1}{\tau} \cdot \frac{QK^\top}{\sqrt{d_k}}\right) V, \qquad \tau > 0, \tag{1}$$

where $1/\tau$ may depend on the query, the head and n, but not on the keys. Letting $1/\tau$ grow with n amplifies every logit gap and keeps the distribution sharp as n increases. This is the idea behind SSMAX (Nakanishi, 2025), which sets $1/\tau = s \log n$ with a learnable per-head

scalar s, and QASSMAX (Qu et al., 2026), which rescales the query as $\tilde{q} = D(q,n)\,q$ with $D(q,n) = \mathrm{diag}\big(\mathrm{base}(n) \odot \mathrm{gate}(q)\big)$. Here $\mathrm{base}(n)$ is a learned length-dependent scaling and $\mathrm{gate}(q)$ is a learned query-dependent gate. At fixed n, $\mathrm{base}(n)$ is absorbed into W_Q and $\mathrm{gate}(q)$ approximately stretches q (Figure 2), so QASSMAX acts as a query-dependent $1/\tau$ (Appendix A).

Setup. For a given head and test row, let $z_1, \ldots, z_n$ be its attention logits over the n training rows, with attention mass $a_j \propto e^{z_j}$. We shift so that $\max_i z_i = 0$, which leaves the softmax output unchanged. The training rows split into an *informative set* R of m rows containing the top-scoring one, with $\min_{i\in R} z_i = -\delta$, and a *tail* T of the remaining $n-m$ uninformative rows **at the same depth**, each with $z_j = -\Delta$, where $\Delta > \delta > 0$. We define the *share gap* as:

$$\gamma \;=\; \frac{\delta}{\Delta} \;\in\; (0,1), \tag{2}$$

which is the spread inside R as a fraction of the drop to the tail: it measures how sharply informative rows separate from uninformative ones.

Importantly, **the sets R and T are head-specific**: each head scores the training rows differently, so R, T, δ, Δ and γ are defined per head and per test row. For a set of rows A, write $M_A = \sum_{j\in A} a_j$ for the mass it receives. We consider that a head aggregates well when it does two things: it should distribute little mass outside R, so M_T/M_R should be *small*, and it should distribute its mass across all of R, although not necessarily uniformly. We measure this by

$$\rho_R \;=\; \frac{\min_{j\in R} a_j}{\max_{j\in R} a_j}. \tag{3}$$

With $t = e^{-\Delta/\tau}$ and $\delta = \gamma\Delta$, scaling leaves the top row at unnormalized mass 1, the weakest informative row at $e^{-\delta/\tau} = t^\gamma$, and every tail row at t (at the same depth). With $1 \le M_R \le m$,

$$\rho_R = t^\gamma, \qquad \tfrac{n-m}{m}\, t \;\le\; \frac{M_T}{M_R} \;\le\; (n-m)\, t. \tag{4}$$

Sharpening ($1/\tau \uparrow$, $t \downarrow$) lowers both ρ_R and $\frac{M_T}{M_R}$, so the tail shrinks fast while coverage falls at a rate set by γ. After eliminating t in Equation 4, we formalize the following proposition:

Proposition 1 (Temperature trades coverage for suppression)**.** *For any $1/\tau > 0$ that may depend on the query, the head and n, but not on the keys,*

$$\rho_R \;\le\; \left(\frac{m}{n-m}\cdot\frac{M_T}{M_R}\right)^{\gamma}. \tag{5}$$

Hence driving $M_T/M_R \to 0$ forces $\rho_R \to 0$, at a rate set by γ.

Intuitively, $1/\tau$ acts as a single knob: raising it pushes *every* row away from the top-scoring one in proportion to how far below it already sits, so the tail and the lower-scoring informative rows are suppressed together and γ stays constant. Removing tail mass by a factor c therefore costs a factor c^γ in coverage, and the trade-off worsens with context length, since the tail outgrows R linearly in n. QASSMAX is no exception (Figure 2): of its two query factors, only $\mathrm{base}(n)$ turns q towards R (above the dashed line), and as it is independent of the query and keys it folds into W_Q; the query-dependent gate only stretches q, i.e., acts as a temperature. In the following section we introduce **Tabular Selective Attention (TSA)**, which mitigates this trade-off with a key-dependent term.

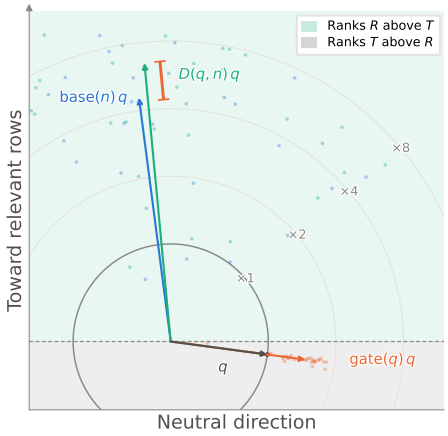

Figure 2: In QASSMAX, only $\mathrm{base}(n)$ turns q towards the informative samples R; the query-aware gate only stretches it. (Each dot is one retrieval head's median query after applying each factor, projected onto the plane shown.)

4 TABULAR SELECTIVE ATTENTION

Tabular Selective Attention (TSA) is a modification on top of standard attention for tabular in-context learning

(ICL). Inspired by Leviathan et al. (2025), the core idea is that test rows can *mask* training rows, i.e., the amount of attention a test row i pays a training row j can be reduced by an amount learned by the pair (i, j). In standard tabular ICL models, training rows attend to each other while test rows only attend to the training rows. TSA suppresses only the attention between test-to-training rows, while training rows continue to attend to each other without suppression (see Appendix D for an exception in TabDPT).

Concretely, for each head h we introduce a learned matrix $W^{(h)} \in \mathbb{R}^{d_k \times d_k}$, which parametrizes a suppression score using the test queries Q_{test} and training keys K_{train}:

$$F^{(h)} = \mathrm{ReLU}\big(Z^{(h)}\big) = \mathrm{ReLU}\left(\frac{Q_{\text{test}}\, W^{(h)} K_{\text{train}}^{\top}}{\sqrt{d_k}}\right) \in \mathbb{R}^{n_{\text{test}} \times n_{\text{train}}}. \tag{6}$$

where Z_{ij} denotes the suppression score applied by test row i to training row j. We further constrain the score to be non-negative via $F^{(h)} = \mathrm{ReLU}(Z^{(h)})$ and apply it as follows:

$$\mathrm{TSA}_h(Q, K, V) = \mathrm{softmax}\left(\frac{QK^{\top}}{\sqrt{d_k}} - F^{(h)}\right) V. \tag{7}$$

The mechanism can only lower the pre-softmax scores, thereby reallocating attention mass from the suppressed training rows towards the unsuppressed ones. TSA requires $d_k \times d_k$ additional parameters per head, comparable to other methods such as QASSMAX.

4.1 Theoretical Analysis of Tabular Selective Attention

In this analysis, we rely on the assumption that each head h has its own tail set T and the informative set R. Assume further that $\forall j \in R, F_{ij} = 0$ and let $\sigma = \min_{j \in T} F_{ij}$ be the minimum penalty it applies within the tail set T. Then, the spread inside the set R stays at δ while the tail is suppressed to at least $\Delta + \sigma$. Given the setup in the previous section for temperature scaling, we now have the share gap γ' given by:

$$\gamma' \;=\; \frac{\delta}{\Delta + \sigma} \;\le\; \gamma, \tag{8}$$

with strict inequality for $\sigma > 0$. This is precisely what a (uniform) temperature cannot do: scaling stretches δ and Δ by the same factor and leaves their ratio fixed, while TSA deepens the gap to the tail without widening the spread inside the R. As a result, suppressing the tail no longer comes at the expense of coverage. We formalize this result in the following proposition (Details in Appendix A.3):

Proposition 2 (Tail suppression does not cost coverage)**.** *Under the assumptions above, TSA has two properties:* (1) *Coverage is unaffected by suppression: for every depth,* $\sigma \ge 0$, $\rho_R = \min_{j \in R} a_j / \max_{j \in R} a_j = e^{-\delta}$. (2) *Logarithmic depth suffices to suppress the tail: the tail mass stays below a fraction* ϵ *of the informative mass,* $M_T / M_R \le \epsilon$*, whenever*

$$\sigma \;\ge\; \log\!\left(\frac{n-m}{m\,\epsilon}\right) \;-\; (\Delta - \delta), \tag{9}$$

and, if every tail row is suppressed by exactly σ*, this is possible only if* $\sigma \ge \log\!\left(\frac{n-m}{m\,\epsilon}\right) - \Delta$.

This result highlights two ways in which TSA differs from temperature scaling. First, coverage and suppression are now controlled by different terms: coverage by the attention logits on their own, suppression by F. **Increasing penalty on the tail therefore leaves ρ_R precisely where it was**, whereas under Proposition 1 every gain in suppression was paid with coverage on R. Second, by equation 9, the penalty that F subtracts from each tail logit grows as $\Theta(\log n)$. Since $\log n$ grows slowly, a bounded suppression score suffices at practical context lengths.

4.2 Suppression-Guided Row Pruning

By the non-negativity property of TSA suppression scores, we can leverage the learned score matrices to drop the most suppressed rows. At any given layer, we sort the training rows decreasingly on $F_{ij}^{(h)} \ge 0$, so we always evict the most "uninformative" training samples from context first. Removing a row from the context renormalizes the remaining weights, so the change in the output is

controlled by the attention the row was already receiving. Since $F_{im} = 0$ for any row m that TSA does not suppress,

$$\frac{a_{ij}}{a_{im}} = e^{-F_{ij}}\, e^{-(z_{im}-z_{ij})} \quad \Longrightarrow \quad a_{ij} \leq e^{-F_{ij}}\, e^{-(z_{im}-z_{ij})}, \tag{10}$$

where z are the attention logits before suppression. At a fixed logit gap, larger F_{ij} tightens this bound, making such rows natural candidates for pruning, and TSA already computes their scores, so selecting them adds only a small tracking cost.

We recognize two main caveats: (1) each head has its own representation and thus its own scores, and (2) a row dropped in one layer may still be informative to a head in a subsequent layer. To address (1), we simply aggregate each layer's scores into a single score per training row,

$$s_j = \frac{1}{n_h\, n_{\text{test}}} \sum_{h=1}^{n_h} \sum_{i=1}^{n_{\text{test}}} F_{ij}^{(h)}, \tag{11}$$

and sort on s_j in decreasing order. To address (2), we combine this with schedules that decide (i) *where* to prune context (i.e., after which layer), (ii) *how much* to drop at each pruning layer, and (iii) *how confident* the score s_j must be before a row is dropped (see Appendix E for more details).

5 Experiments

TSA is an architecture-agnostic addition to standard attention in the ICL layers of TFMs. However, in this work we focus mainly on TabICLv2's architecture for downstream performance and interpretability results, and add TSA to TabDPT to test transferability to different architectures.

Benchmarks. We use TabArena (Erickson et al., 2025) in the same setting as in the TabICL model series (Qu et al., 2025; 2026): predictions are ensembled using random column or class shuffles and different preprocessors. In this work, we evaluate only the 38 classification datasets using repeated cross-validation with ROC-AUC as a metric for binary classification and log-loss for multiclass tables. We primarily focus on improvability, which measures the average relative error gap relative to the best method on the dataset, allowing the best method to vary across datasets. Additionally, we also report the rank-based metrics average rank and Elo, as common in tabular benchmarks (Liu et al., 2025; Erickson et al., 2025).

Synthetic dataset. Comparing TSA and QASSMax requires a ground-truth *informative set* for each query. Deriving it from the attention logits would make it depend on the quantity under evaluation, so we fix it *a priori*. For each test row, the informative set R consists of training rows planted in its input-space neighbourhood, and all remaining rows form the tail (Figure 3). We restrict the analysis to *retrieval heads*, i.e., heads that already rank informative rows above tail rows before either mechanism is applied ($\text{AUC}_{RT} \geq 0.9$, the probability that an informative row receives higher attention than a tail row is at least 0.9). The partition and the head-selection criterion are independent of the mechanisms and identical for both models. Geometric similarity serves here only as a **controlled proxy** for informativeness; in general, *the two do not necessarily coincide*.

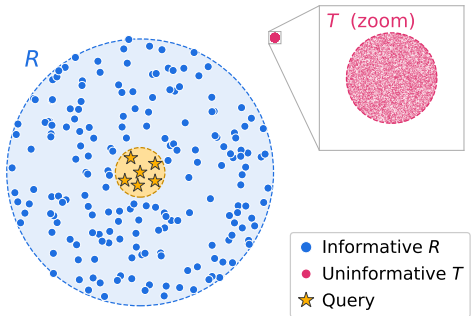

Figure 3: Synthetic dataset (2D sketch). Informative rows R are planted in the neighbourhood of each test row (star); all other training rows form the tail T. See Appendix B for details.

Baselines. While downstream performance is a core component of this evaluation, we only compare to the different attention mechanisms applied to the chosen architecture. For that reason, TSA we compare against **(1)** QASSMax, **(2)** SSMax and **(3)** Vanilla Attention, all built on top of TabICLv2's ICL layers. Additionally, both SSMax and QASSMax are also wired into the column embedder layer, as in the original (Qu et al., 2026). All baselines are pre-trained from scratch under identical conditions using TabICLv2's prior. For more details on pre-training and architectures, see

Appendix C. On TabDPT, TSA is compared against two vanilla-attention models pre-trained from scratch: one with training-time nearest-neighbour retrieval (*Vanilla + train. ret.*), and one without (*Vanilla*). For more details on the TabDPT experimental setup, see Appendix D.

5.1 Downstream Results

Table 1: Downstream performance of TabICLv2 attention variants on TabArena. Train/Pred. report wall-clock time in seconds per 1k samples. *vs. Vanilla* is the head-to-head win rate (%) of each method against vanilla attention, computed per dataset over folds and averaged across TabArena. Best per column in **bold**, second underlined. See Appendix C for more details.

Attention type	Impr. (↓)	Rank (↓)	Elo (↑)	Train (s/1k) (↓)	Pred. (s/1k) (↓)	vs. Vanilla (↑)
TSA	**14.6%**	**21.2**	$\mathbf{1540_{-62,+64}}$	8.61	0.49	**66.1**
QASSMax	17.6%	31.8	$1405_{-65,+64}$	9.29	0.37	31.9
SSMax	17.5%	30.6	$1419_{-64,+63}$	**7.46**	0.37	37.9
Vanilla	15.3%	23.7	$1505_{-66,+57}$	8.83	**0.36**	–

TSA achieves the best improvability on TabArena across attention methods, reducing the average relative error gap to the per-dataset best method to 14.6% (Table 1). This advantage holds in the rank-based metrics, where TSA attains the lowest average rank (21.2) and the highest Elo (1540), and in head-to-head comparisons: TSA has a 66.1% win rate against vanilla attention, whereas SSMax and QASSMax reach only 37.9% and 31.9%. The gap to vanilla attention is smaller than to the scalable-softmax variants, and the Elo confidence intervals of TSA and vanilla overlap. Both SSMax and QASSMax fall behind, which is consistent with their degradation in the later stages of pre-training. At inference, TSA adds overhead of roughly 0.13 s per 1k samples over vanilla attention (0.49 vs. 0.36 s/1k). This overhead comes from computing the suppression score, which is the only additional operation TSA introduces on top of the standard attention computation.

Table 2: Downstream performance of TabDPT attention variants on the TabArena classification datasets with more than 10K training rows. *Context*: the full training set, or the 4,096 nearest training rows of each test row (retrieval). *vs. Vanilla*: win rate (%) against Vanilla with full context. Best in **bold**, second underlined.

Attention type	Context	Impr. (↓)	Rank (↓)	Elo (↑)	vs. Vanilla (↑)
TSA	full	**3.55%**	**1.90**	$\mathbf{1271_{-116,+197}}$	**80.0**
Vanilla	full	4.40%	3.00	$1125_{-128,+136}$	–
Vanilla + train. ret.	full	6.96%	3.80	$1026_{-83,+76}$	20.0
Vanilla	retrieval	4.18%	3.10	$1100_{-123,+105}$	50.0
Vanilla + train. ret.	retrieval	4.24%	3.20	$1088_{-123,+114}$	50.0

TSA is transferable to other architectures. We add TSA to TabDPT v1.1, which, unlike TabICLv2, embeds each row with a single linear encoder, is pre-trained on real tables, and processes rows in one attention call. We pre-train with contexts up to 4,096 rows, and then evaluate on the 10 TabArena classification datasets with more than 10K training samples, the limit up to which TabArena runs TabPFNv2 (Erickson et al., 2025). TSA uses the full training set as context, and the baselines are evaluated both with the full training set and with retrieval of the 4,096 nearest training rows per test row. Details regarding the full evaluation setup are in Appendix D. TSA achieves the best improvability, the best average rank, and the highest Elo (Table 2).

5.2 Understanding TSA

On synthetic data, TSA maintains coverage while QASSMax concentrates. To empirically validate the theory, we use our synthetic dataset, an idealized setting in which R and T are separable and distinguishable. For each test row, we compute the tail-to-informative mass ratio M_T/M_R and the coverage ρ_R twice: once with the mechanism switched off ($F = 0$ in every layer for TSA,

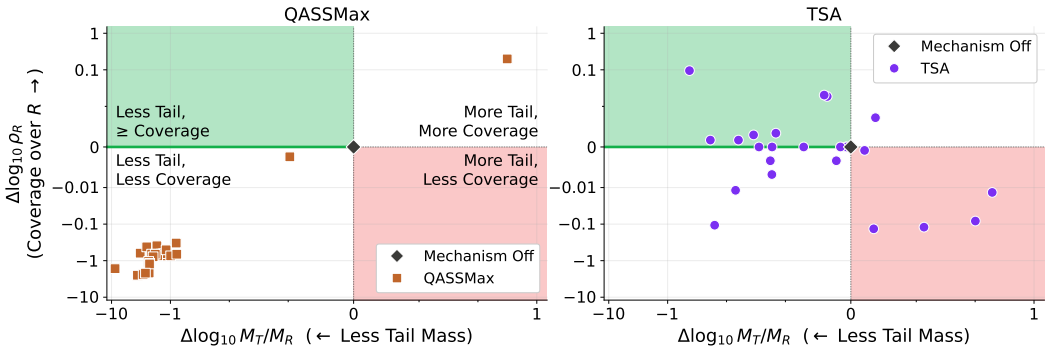

Figure 4: Effect of each mechanism on tail mass and coverage. Each marker is one retrieval head and shows how its attention changes when the mechanism is switched on, relative to the same checkpoint with the mechanism off (⧫, the origin). **Left:** QASSMAX suppresses the tail mass while also reducing coverage over R. **Right:** TSA suppresses the tail mass while effectively maintaining/increasing coverage over R on 15 of the 28 active heads. On the remaining heads, the mask leaks onto R rather than trade coverage for suppression.

$\tau = 1$ for QASSMAX) and once with it on. Figure 4 plots the change in both quantities on a logarithmic scale, taking the median over each head's test rows. The origin is the head without the mechanism, and the top-left quadrant is where both improve: less tail mass and more coverage. As Proposition 1 predicts, a scalar temperature trades coverage over R for tail suppression, so 23/24 QASSMAX heads move into the bottom-left quadrant. Proposition 2 instead predicts that a key-dependent suppression which leaves R unsuppressed removes tail mass without any trade-off in coverage. This holds for 15 of the 28 active TSA heads, which move left while staying on or above the horizontal axis. Most of the remaining 13 lose coverage because masks in earlier layers change their input, not because of their own mask: comparing a head's logits before and after its own $\mathrm{ReLU}(Z_h)$ is subtracted, 9 of them leave coverage exactly unchanged. Only four heads lose coverage through their own mask (10–27% of ρ_R). These heads leak onto R rather than trade coverage for suppression.

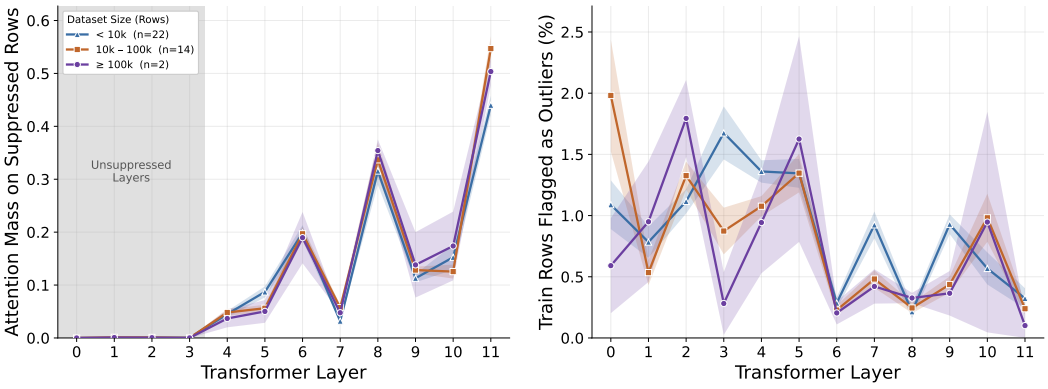

Figure 5: Layer-wise behaviour of TSA on TabArena, grouped by dataset size. **Left:** fraction of attention mass assigned to suppressed rows in each ICL layer. In layers 0–3 (grey), suppressed rows receive almost no attention mass. **Right:** percentage of training rows flagged as outliers in each layer. At most about 2% of rows are flagged in any layer, yet in deeper layers suppressed rows carry a large share of the attention mass. The trend is consistent across dataset sizes. Shaded bands show the standard error across datasets.

TSA operates sparsely, becomes more active with depth, and does not depend on table size. Figure 5 plots, per ICL layer, the fraction of training rows flagged as outliers and the share of attention mass assigned to suppressed rows. In the early layers, suppression rarely triggers and suppressed rows receive almost no attention mass. From the middle layers onwards, the attention

mass on suppressed rows grows non-monotonically and peaks in the final layer, while the fraction of rows flagged as outliers stays at or below about 2% in every layer (a row is flagged if its suppression score exceeds $\mu + 3\sigma$, with μ and σ computed over that layer's positive scores). The bilinear matrix is initialized with a zero-mean Gaussian; therefore, roughly half of all pairs are initially suppressed in every layer, whereas after training the early suppression scores remain non-degenerate but mostly fall just below zero. Finally, the layer-wise profiles of small, medium and large datasets closely overlap despite large differences in context length, so TSA is not a generic context-pruning mechanism: *it targets a sparse, depth-specific set of rows, independent of table size*.

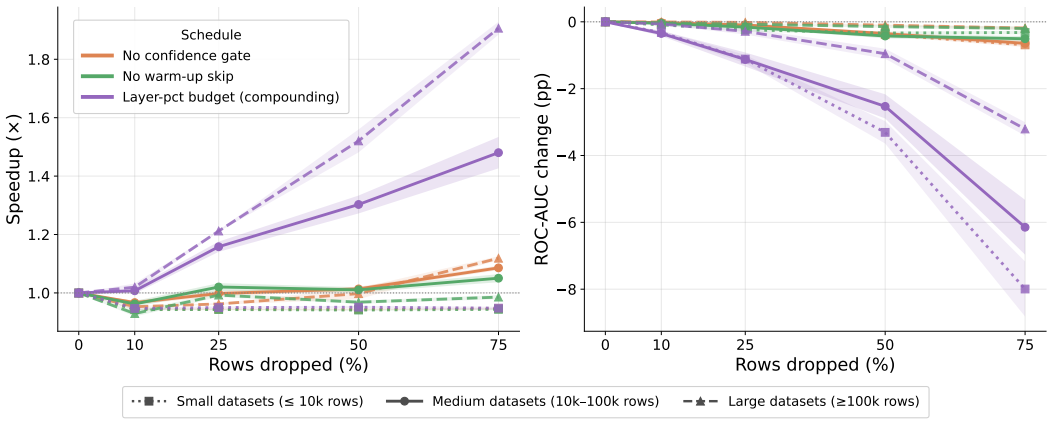

Figure 6: Inference time speedup (left) and **ROC-AUC degradation** (right) with respect to the no row-dropping schedule, across different schedules and different dropping %. Averaged across seeds in TabArena small ($<$ 10k rows), medium ($10 - -100$k) and large datasets ($\geq$ 100k).

Row dropping becomes suitable on larger tables. We evaluate row dropping on the 38 classification datasets of TabArena (5 seeds each) with drop rates $p \in \{10, 25, 50, 75\}\%$, comparing three schedules: (a) dropping rows from early layers, (b) dropping rows without a confidence gate, and (c) a naive compounding scheme that reapplies the $p\%$ budget after each layer to the remaining context (see Figure 6). The results yield two conclusions: **(i)** Smaller tables have few redundant rows; therefore, degradation grows quickly with p, and the shorter context leaves little opportunity to amortize the cost of scoring rows. **(ii)** Larger tables contain more redundant rows, so the same drop rate removes mainly redundant information and degrades performance more gracefully, while the longer context lets speedups grow with both table size and p. The compounding scheme (c) shows the trade-off most clearly: it achieves the largest speedups, most pronounced on large tables, at the cost of a steeper performance loss.

6 Conclusion

In TFMs, long-context generalization is typically approached either with retrieval, which is expensive and uses similarity as a proxy for informativeness, or with temperature scaling, which is key-independent and cannot suppress uninformative rows without also removing mass from weakly informative ones. As an alternative, we introduce **Tabular Selective Attention (TSA)**, a model-agnostic modification of the attention block that learns a non-negative, key-dependent suppression score for each test/training-row pair. TSA can therefore lower the logits of uninformative rows while leaving those of informative rows unchanged and, under the assumptions of Proposition 2, redistributes the removed mass over the informative set without altering its relative weights. Ranking training rows by these scores also yields a pruning rule that shrinks the context layer by layer. However, calculating the scores add inference overhead (0.49 vs. 0.36 s per 1k samples), and pruning pays off only on larger tables to amortize the scoring cost. It also aggregates head- and query-specific scores that need not agree with each other, and reconciling them is a natural next step. Nonetheless, in an identical TabICLv2 backbone, TSA achieves the best improvability, average rank and Elo among attention variants on TabArena classification, with a 66.1% win rate against vanilla attention. When TabDPT is pre-trained with TSA, it also ranks first on TabArena datasets with more than 10K training rows using the full training set as context, ahead of nearest-neighbour retrieval baselines.

AI USE STATEMENT

In this work, we used generative AI tools to aid or polish writing. This includes checking for spelling, grammar and sentence structure. We also used generative AI tools for research execution. This includes coding the experiments to test the implemented methods. However, TSA was implemented personally and no AI tools were used. All code generated using these tools was manually verified. AI tools were also used to compile the supplementary code attached with the submission. We used generative AI tools for assistance in the writing of proofs. The idea for the mathematical proofs in Section 3 and 4 was inspired from previous literature around temperature scaling. Generative AI tools were only used to formalize the language in the proof. All mathematical claims were manually verified. We did not use these tools for generating synthetic data sets, help develop theoretical models or conceptual frameworks, provide critical ingredients for proving mathematical claims, propose or refine hypotheses, formulate mathematical claims, design or provide feedback on research, methodology or experiments, assist with translation, clean and reformat dataset, support qualitative and thematic data analysis, interpret results. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REFERENCES

David Bonet, Daniel Mas Montserrat, Xavier Giró-i Nieto, and Alexander G Ioannidis. HyperFast: Instant classification for tabular data. In *Proceedings of the AAAI conference on artificial intelligence*, volume 38, pp. 11114–11123, 2024.

Vadim Borisov, Tobias Leemann, Kathrin Seßler, Johannes Haug, Martin Pawelczyk, and Gjergji Kasneci. Deep neural networks and tabular data: A survey. *IEEE Transactions on Neural Networks and Learning Systems*, 35(6):7499–7519, June 2024. ISSN 2162-2388. doi: 10.1109/tnnls.2022.3229161. URL http://dx.doi.org/10.1109/TNNLS.2022.3229161.

Shi Chen, Zhengjiang Lin, Yury Polyanskiy, and Philippe Rigollet. Critical attention scaling in long-context transformers. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=7SLtElfqCW.

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

Aaron Defazio, Xingyu Yang, Harsh Mehta, Konstantin Mishchenko, Ahmed Khaled, and Ashok Cutkosky. The road less scheduled. *Advances in Neural Information Processing Systems*, 37: 9974–10007, 2024.

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 *The Thirty-ninth Annual Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2025. URL https://openreview.net/forum?id=jZqCqpCLdU.

Léo Grinsztajn, Klemens Flöge, Oscar Key, Felix Birkel, Philipp Jund, Brendan Roof, Benjamin Jäger, Dominik Safaric, Simone Alessi, Adrian Hayler, et al. TabPFN-2.5: Advancing the state of the art in tabular foundation models. *arXiv preprint arXiv:2511.08667*, 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. URL https://arxiv.org/abs/2605.13986.

Noah Hollmann, Samuel Müller, Katharina Eggensperger, and Frank Hutter. TabPFN: A transformer that solves small tabular classification problems in a second. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=cp5PvcI6w8_.

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

Rasa Hosseinzadeh, Alex Labach, Zexin Xue, Shuyi Han, Valentin Thomas, and Anthony L. Caterini. TabDPT-turbo: Efficient in-context learning for tabular prediction, 2026. URL https://arxiv.org/abs/2608.01400.

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

Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers are RNNs: Fast autoregressive transformers with linear attention. In *Proceedings of the 37th International Conference on Machine Learning (ICML)*, volume 119 of *Proceedings of Machine Learning Research*, pp. 5156–5165. PMLR, 2020.

Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. LightGBM: A highly efficient gradient boosting decision tree. In *Advances in Neural Information Processing Systems 30*, 2017. URL https://proceedings.neurips.cc/paper/2017/file/6449f44a102fde848669bdd9eb6b76fa-Paper.pdf.

Weihao Kong and Abhimanyu Das. Introducing TabFM: A zero-shot foundation model for tabular data. *Google Research Blog*, 2026. URL https://research.google/blog/introducing-tabfm-a-zero-shot-foundation-model-for-tabular-data/. Code and weights available at https://github.com/google-research/tabfm.

Yaniv Leviathan, Matan Kalman, and Yossi Matias. Selective attention improves transformer. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=v0FzmPCd1e.

Si-Yang Liu and Han-Jia Ye. TabSwift: An efficient tabular foundation model with row-wise attention. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=ottLtChHp2.

Si-Yang Liu, Hao-Run Cai, Qi-Le Zhou, Huai-Hong Yin, Tao Zhou, Jun-Peng Jiang, and Han-Jia Ye. Talent: A tabular analytics and learning toolbox. *J. Mach. Learn. Res.*, 26:226:1–226:16, 2025. URL https://jmlr.org/papers/v26/25-0512.html.

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, 2026. URL https://arxiv.org/abs/2410.18164.

Andreas C Mueller, Carlo A Curino, and Raghu Ramakrishnan. MotherNet: Fast training and inference via hyper-network transformers. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=6H4jRWKFc3.

Samuel Müller, Noah Hollmann, Sebastian Pineda Arango, Josif Grabocka, and Frank Hutter. Transformers can do Bayesian inference. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=KSugKcbNf9.

Ken M Nakanishi. Scalable-softmax is superior for attention. *arXiv preprint arXiv:2501.19399*, 2025.

Ben Peters, Vlad Niculae, and André F. T. Martins. Sparse sequence-to-sequence models. In Anna Korhonen, David Traum, and Lluís Màrquez (eds.), *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 1504–1519, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1146. URL https://aclanthology.org/P19-1146/.

Liudmila Prokhorenkova, Gleb Gusev, Aleksandr Vorobev, Anna Veronika Dorogush, and Andrey Gulin. CatBoost: unbiased boosting with categorical features. In *Advances in Neural Information Processing Systems 31*, 2018. URL https://proceedings.neurips.cc/paper/2018/hash/14491b756b3a51daac41c24863285549-Abstract.html.

Jingang Qu, David Holzmüller, Gaël Varoquaux, and Marine Le Morvan. TabICL: A tabular foundation model for in-context learning on large data. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=0VvD1PmNzM.

Jingang Qu, David Holzmüller, Gaël Varoquaux, and Marine Le Morvan. TabICLv2: A better, faster, scalable, and open tabular foundation model. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=SxsyLjIfWB.

Jay Shah, Ganesh Bikshandi, Ying Zhang, Vijay Thakkar, Pradeep Ramani, and Tri Dao. FlashAttention-3: Fast and accurate attention with asynchrony and low-precision. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=tVConYid20.

Nour Shaheen, Junwei Ma, Alex Labach, Frank Hutter, Valentin Thomas, and Anthony L Caterini. Understanding the surprising generalization properties of tabular foundation models. *arXiv preprint arXiv:2608.17957*, 2026.

Ravid Shwartz-Ziv and Amitai Armon. Tabular data: Deep learning is not all you need. *Information fusion*, 81:84–90, 2022.

Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024.

Valentin Thomas, Junwei Ma, Rasa Hosseinzadeh, Keyvan Golestan, Guangwei Yu, Maksims Volkovs, and Anthony L. Caterini. Retrieval & fine-tuning for in-context tabular models. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=337dHOexCM.

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 (NeurIPS)*, volume 30, pp. 5998–6008, 2017.

Pavlo Vasylenko, Hugo Pitorro, Andre Martins, and Marcos Vinicius Treviso. Long-context generalization with sparse attention. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=PsB6Lynznk.

Petar Veličković, Christos Perivolaropoulos, Federico Barbero, and Razvan Pascanu. Softmax is not enough (for sharp size generalisation). In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=S4JmmpnSPy.

Yuanrui Wang, Xingxuan Zhang, Han Yu, Mingchao Hao, Gang Ren, Hao Yuan, Li Mao, Yunjia Zhang, Chun Yuan, and Peng Cui. LimiX-2m: Mitigating low-rank collapse and attention bottlenecks in tabular foundation models. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=YjYPYg112A.

Yuchen Zeng, Tuan Dinh, Wonjun Kang, and Andreas C Mueller. TabFlex: Scaling tabular learning to millions with linear attention. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=d60cmFf89H.

Xingxuan Zhang, Gang Ren, Han Yu, Hao Yuan, Hui Wang, Jiansheng Li, Jiayun Wu, Lang Mo, Li Mao, Mingchao Hao, Ningbo Dai, Renzhe Xu, Shuyang Li, Tianyang Zhang, Yue He, Yuanrui Wang, Yunjia Zhang, Zijing Xu, Dongzhe Li, Fang Gao, Hao Zou, Jiandong Liu, Jiashuo Liu, Jiawei Xu, Kaijie Cheng, Kehan Li, Linjun Zhou, Qing Li, Shaohua Fan, Xiaoyu Lin, Xinyan Han, Xuanyue Li, Yan Lu, Yuan Xue, Yuanyuan Jiang, Zimu Wang, Zhenlei Wang, and Peng Cui. LimiX: Unleashing structured-data modeling capability for generalist intelligence. *CoRR*, abs/2509.03505, September 2025. URL https://doi.org/10.48550/arXiv.2509.03505.

Xingxuan Zhang, Gang Ren, Hao Yuan, Hao Zou, Hongze Tan, Hui Wang, Jianhao Song, Jiansheng Li, Jiayao Zhang, Jinghan Zhang, Kaifang Li, Lang Mo, Li Mao, Mingchao Hao, Nuo Xu, Rui Ding, Ruiji Zhang, Shuyang Li, Siyu Mei, Tianyang Zhang, Weiyang Mu, Yancheng Dong, Yongxian Wei, Yuan Xue, Yuanrui Wang, Yue He, Zijia Yang, Ziyun Li, Dongzhe Li, Fuqiang Wang, Jiandong Liu, Jiawei Chen, Jiaxin Du, Kaijie Cheng, Kehan Li, Lei Sun, Linjun Zhou, Ningbo Dai, Qi Wang, Renzhe Xu, Shaoxing Du, Shumeng Yang, Wang Lu, Wenjing Chu, Xiannan Huang, Xiaoyu Lin, Xing Ai, Xinyan Han, Xuanyue Li, Xuanyue Su, Xukun Zhang, Yan Lu, Yaxin Zhang, Yi Qin, Yifei Huang, Yihan Xu, Yongle Lv, Yuanyuan Jiang, Yushan Han, and Peng Cui. LimiX-2: A contextual mechanism network towards general structured-data intelligence, 2026. URL https://arxiv.org/abs/2609.17488.

Appendices

Contents

A Theory 15
A.1 Proof of Proposition 1 . . . 15
A.2 QASSMAX as a query-direction temperature . . . 15
A.3 Proof of Proposition 2 . . . 16
B Synthetic Dataset 17
B.1 Geometry . . . 17
B.2 Rationale behind geometry . . . 17
B.3 Labels . . . 17
B.4 Configuration . . . 17
C Architecture and Pre-training Details on TabICLv2 Experiments 18
C.1 Shared Backbone . . . 18
C.2 Tabular Selective Attention (TSA) . . . 18
C.3 Baselines . . . 18
C.4 Pre-training . . . 19
D Pre-training and Evaluation Details on TabDPT Experiments 20
D.1 Architecture . . . 20
D.2 TSA in TabDPT . . . 20
D.3 Pre-training . . . 20
D.4 Evaluation . . . 21
E Details on Row Dropping 22
E.1 Implementation . . . 22
E.2 Default schedule . . . 22
E.3 Ablated schedules . . . 23

A THEORY

A.1 PROOF OF PROPOSITION 1

We restate the proposition for convenience.

Proposition (Temperature trades coverage for suppression)**.** *For any $1/\tau > 0$ that may depend on the query, the head and n, but not on the keys,*

$$\rho_R \;\le\; \left(\frac{m}{n-m}\cdot\frac{M_T}{M_R}\right)^{\gamma}. \tag{12}$$

Hence driving $M_T/M_R \to 0$ forces $\rho_R \to 0$, at a rate set by γ.

Proof. Scaling maps each logit $z_i \mapsto z_i/\tau$, so that $a_j \propto e^{z_j/\tau}$. Since $1/\tau > 0$ does not depend on the keys, it preserves the ordering of the rows and keeps the top row at 0. Hence R and T are unchanged, the gaps become δ/τ and Δ/τ, and $\gamma = \delta/\Delta$ is unchanged. Both ρ_R and M_T/M_R are ratios of masses, so the softmax normalization cancels and we may work with the unnormalized masses $w_j = e^{z_j/\tau}$. Write

$$t \;=\; e^{-\Delta/\tau} \in (0,1).$$

Within R, the top row has the largest mass, $w = 1$, and the weakest informative row has the smallest, $w = e^{-\delta/\tau}$. Since $\delta = \gamma\Delta$ by equation 2,

$$\rho_R \;=\; e^{-\delta/\tau} \;=\; e^{-\gamma\Delta/\tau} \;=\; t^{\gamma}. \tag{13}$$

Every tail row has mass t, so $M_T = (n-m)\,t$. Every informative row has mass at most 1, and the top row has mass exactly 1, so $1 \le M_R \le m$. Therefore

$$\frac{n-m}{m}\,t \;\le\; \frac{M_T}{M_R} \;\le\; (n-m)\,t. \tag{14}$$

By equation 13 and equation 14, both sides of equation 12 are controlled by $t = e^{-\Delta/\tau}$ alone. Since $1/\tau$ enters only through t, it cannot move them independently.

The left inequality in equation 14 gives $t \le \frac{m}{n-m}\cdot\frac{M_T}{M_R}$. Raising both sides to the power $\gamma > 0$ and substituting equation 13 yields equation 12. The right inequality, treated in the same way, gives the matching lower bound $\rho_R \ge \left(\frac{1}{n-m}\cdot\frac{M_T}{M_R}\right)^{\gamma}$, so equation 12 is tight up to a factor m^{γ}.

Finally, suppose $M_T/M_R \to 0$ with n and m fixed. The left inequality in equation 14 then forces $t \to 0$, and so $\rho_R = t^{\gamma} \to 0$. For the rate, equation 14 gives $\log(M_T/M_R) = \log t + c$, where $c \in [\log\frac{n-m}{m}, \log(n-m)]$ is bounded. By equation 13, $\log\rho_R = \gamma\log t$. Hence

$$\frac{\log\rho_R}{\log(M_T/M_R)} \;=\; \frac{\gamma\log t}{\log t + c} \;\longrightarrow\; \gamma \qquad (t\to 0),$$

and this holds for any choice of τ. □

A.2 QASSMAX AS A QUERY-DIRECTION TEMPERATURE

QASSMAX is a query-scaling mechanism that rescales every query element as:

$$\begin{aligned} \tilde{q} &= q\cdot D(q,n) \\ \text{where}\quad D(q,n) &= \operatorname{diag}\big(\text{base}(n)\odot\text{gate}(q)\big) \\ \text{base}(n) &= \text{MLP}_{\text{base}}(\log n) \\ \text{gate}(q) &= 1+\tanh\big(\text{MLP}_{\text{gate}}(q)\big) \end{aligned} \tag{15}$$

The TabICLv2 model with QASSMAX is trained only with the rescaled query $\tilde{q}$, never with the raw query q. Writing $\tilde{q} = u/\tau$ with $\tau = 1/\|\tilde{q}\|$ and $\|u\| = 1$, the logits are $z_j = \frac{1}{\tau}\,u^{\top}k_j/\sqrt{d_k}$, i.e., Eq. (1) with query u. Any vector admits this decomposition, so it is not by itself an argument; what matters is which factor the mechanism varies. Proposition 1 applies with temperature τ and with δ, Δ measured on the unit-scale logits $u^{\top}k_j/\sqrt{d_k}$, provided u does not change as τ varies.

Two observations reduce QASSMAX to this case. First, at fixed n, $\mathrm{base}(n)$ is a constant diagonal map and can be absorbed into W_Q, so it does not act as a knob. Second, if $\mathrm{gate}(q)$ is approximately proportional to the all-ones vector, it rescales $\tilde{q}$ without rotating it, so it changes only τ. QASSMAX then acts as a query-dependent temperature and inherits the frontier of Proposition 1. The second condition is not guaranteed by the architecture; it is an empirical observation (Figure 2). To the extent that the gate rotates u, it changes γ, but only through a key-independent, per-coordinate reweighting of the query, not through a penalty on individual rows as in TSA.

A.3 PROOF OF PROPOSITION 2

We restate the proposition for convenience.

Proposition (Tail suppression does not cost coverage)**.** *Under the assumptions above, TSA has two properties:* (1) *Coverage is unaffected by suppression: for every depth $\sigma \geq 0$, $\rho_R = \min_{j\in R} a_j / \max_{j\in R} a_j = e^{-\delta}$.* (2) *Logarithmic depth suffices to suppress the tail: the tail mass stays below a fraction ϵ of the informative mass, $M_T/M_R \leq \epsilon$, whenever*

$$\sigma \;\geq\; \log\Big(\frac{n-m}{m\,\epsilon}\Big) \;-\; (\Delta - \delta), \tag{9}$$

and this is possible only if $\sigma \geq \log\big(\frac{n-m}{m\,\epsilon}\big) - \Delta$.

Proof. Under TSA the head attends with $a_j \propto e^{z_j - F_{ij}}$, where $F_{ij} = 0$ for $j \in R$ and $F_{ij} \geq \sigma$ for $j \in T$. Both ρ_R and M_T/M_R are ratios of masses, so softmax normalization cancels and we may work with the unnormalized masses $w_j = e^{z_j - F_{ij}}$.

Since $F_{ij} = 0$ on R, the informative rows keep their original logits. The top row has mass $w = 1$, and the weakest informative row has mass $w = e^{-\delta}$, so

$$\rho_R \;=\; \frac{e^{-\delta}}{e^{0}} \;=\; e^{-\delta}$$

for every $\sigma \geq 0$. This proves (1). Unlike the temperature case equation 13, ρ_R does not depend on the quantity being tuned.

For (2), every tail row has logit $-\Delta - F_{ij} \leq -(\Delta + \sigma)$, and every informative row has mass in $[e^{-\delta}, 1]$. Hence

$$M_T \;\leq\; (n-m)\,e^{-(\Delta+\sigma)}, \qquad m\,e^{-\delta} \;\leq\; M_R \;\leq\; m. \tag{16}$$

Dividing the bound on M_T by the lower bound on M_R gives

$$\frac{M_T}{M_R} \;\leq\; \frac{(n-m)\,e^{-(\Delta+\sigma)}}{m\,e^{-\delta}} \;=\; \frac{n-m}{m}\,e^{-(\Delta+\sigma-\delta)}.$$

The right-hand side is at most ϵ exactly when $\Delta + \sigma - \delta \geq \log\big(\frac{n-m}{m\,\epsilon}\big)$, which is equation 9.

For the converse, suppose every tail row is suppressed by exactly σ, i.e., $F_{ij} = \sigma$ for all $j \in T$. Then $M_T = (n-m)\,e^{-(\Delta+\sigma)}$. Combined with $M_R \leq m$ from equation 16, this gives

$$\frac{M_T}{M_R} \;\geq\; \frac{n-m}{m}\,e^{-(\Delta+\sigma)},$$

so $M_T/M_R \leq \epsilon$ forces $\sigma \geq \log\big(\frac{n-m}{m\,\epsilon}\big) - \Delta$.

The sufficient and necessary thresholds differ only by δ, the spread inside R, which does not grow with n. For fixed m and ϵ, both are $\log n + O(1)$, so the worst-case required depth is $\Theta(\log n)$. □

Remark 3. The analysis assumes $F_{ij} = 0$ on R. A test row far from all training rows still has a relative best key, but a suppression score that is not computed relative to that key can leave $F_{ij} > 0$ for some $j \in R$. Write $\phi = \max_{j\in R} F_{ij}$ for the largest such value. The effective spread inside R is then at most $\delta + \phi$ instead of δ. Coverage degrades to $\rho_R \geq e^{-(\delta+\phi)}$, and equation 9 picks up the same additive term ϕ. The guarantee therefore degrades gracefully rather than failing. It is exact when the mechanism is inactive on R, a state the ReLU in equation 6 makes reachable: any $Z_{ij} \leq 0$ gives $F_{ij} = 0$.

B SYNTHETIC DATASET

The synthetic dataset of section 5 provides a context in which the informative set R and the uninformative set T are known *a priori* for every query. This lets us compute δ, Δ, γ, ρ_R and M_T/M_R from a head's logits without inferring informativeness from those same logits. Informativeness is defined by input-space geometry where every training row belongs to one of two disjoint clusters, designated informative and uninformative in advance and independently of any model's output.

B.1 GEOMETRY

The training rows are sampled from a 2D input space where the informative cluster R is a disc of radius r_R centred at the origin and the uninformative cluster T is a disc of radius r_T centred at $(D/\sqrt{2}, D/\sqrt{2})$. Therefore, the centres of the two clusters are separated by a distance D along the diagonal. R has m rows and T has $n - m$ rows. Rows are sampled uniformly by area inside each disc ($\rho = r\sqrt{u}$ with $u \sim \text{Uniform}[0, 1]$), making them uniform in squared radius. Test rows (queries) are sampled in the same manner from a small disc of radius r_q at the centre of R.

B.2 RATIONALE BEHIND GEOMETRY

Proposition 1 bounds the coverage ρ_R a temperature τ can retain at a given uninformative set mass M_T/M_R, and the exchange rate between the two depends on the logits through $\gamma = \frac{\delta}{\Delta}$. Using insights from Shaheen et al. (2026), which proposes that tabular ICL models generalize by retrieving and aggregating, we use a Euclidean proxy in which a head scores a key by its squared distance to the query, $\ell_j \propto -\|q - k_j\|^2$.

Under this assumption, for a query at the centre of R, the spread δ inside the informative set scales as r_R^2 and the drop Δ to the nearest tail row as $(D - r_T)^2$. The uninformative cluster disc is kept tiny so that the set is compact in input space, and the query radius is set so that the *gap constraint* $r_R + 2q_r < D - r_T$ holds, which guarantees that every informative row is closer to every query than any tail row is. The default $r_R/D = 0.7$ satisfies the gap constraint with margin at the default r_T and q_r.

B.3 LABELS

The informative set R is assigned label $y = 0$, and the uninformative set T is assigned label $y = 1$. Every training label, in both clusters, is then flipped independently with probability $p = 0.1$. All queries have label $y = 0$. We introduce label noise in both clusters so that 1-nearest-neighbour is not the optimal predictor. With clean labels, the single closest row already gives the right answer. Therefore, by introducing noisy labels, the head has to spread its mass over the informative set to predict reliably.

B.4 CONFIGURATION

Table 3 lists the default configuration used to generate the synthetic dataset.

Table 3: Default configuration used to generate the synthetic dataset.

Parameter	Symbol	Default	Role
Centre distance	D	1	sets the scale
Informative radius	r_R/D	0.7	spread of R relative to D (input space)
Tail radius	r_T/D	0.02	keeps the tail compact in input space
Query radius	q_r/r_R	0.05	keeps the gap constraint satisfied
Informative rows	m	200	size of R
Context rows	n	10^4	$n - m$ tail rows
Queries	—	64	test rows, all with $y = 0$
Label flip rate (both clusters)	p	0.10	makes 1-NN suboptimal
Seeds	—	$0, \ldots, 4$	one dataset per seed

C ARCHITECTURE AND PRE-TRAINING DETAILS ON TABICLV2 EXPERIMENTS

All models share the TabICLv2 classification architecture and pre-training pipeline (Qu et al., 2026). They differ only in how attention logits are modulated in the ICL predictor and, where noted, the column embedder. We build on the official implementation (`https://github.com/soda-inria/tabicl`). SSMAX, QASSMAX and the vanilla baseline use the code already in that repository unchanged. TSA is our only addition (subsection C.2). The shared backbone is described in subsection C.1, the baselines in subsection C.3, and the pre-training procedure in subsection C.4.

C.1 SHARED BACKBONE

We use the TabICLv2 architecture with its default hyperparameters (Qu et al., 2026, Table A.1). It has three stages:

- a *column embedder*: 3 induced set attention blocks, 128 inducing points, target-aware and with feature grouping (group size 3);
- a *row interactor*: 3 blocks with RoPE (Su et al., 2024) and 4 `[CLS]` tokens;
- an *ICL predictor*: 12 pre-norm transformer blocks with 8 heads, where training rows attend to each other and test rows attend only to training rows.

The embedding dimension is 128 (512 in the ICL predictor, after concatenating the `[CLS]` tokens), the feed-forward expansion factor is 2, and we use GELU activations, LayerNorm with biases and standard (non-zero) residual initialization. Every variant in this work keeps these settings. The only change is the attention-modulation component described below.

C.2 TABULAR SELECTIVE ATTENTION (TSA)

TSA adds a learned, non-positive bias to the *test-to-train* attention logits in every ICL block. For head h, with test queries $Q_h \in \mathbb{R}^{m\times d_k}$ and training keys $K_h \in \mathbb{R}^{n\times d_k}$,

$$A_h = \mathrm{softmax}\left(\frac{Q_h K_h^\top}{\sqrt{d_k}} \; - \; \mathrm{ReLU}\left(\frac{Q_h W_h K_h^\top}{\sqrt{d_k}}\right)\right), \tag{17}$$

where $W_h \in \mathbb{R}^{d_k\times d_k}$ is a learnable matrix for each head, initialized as $W_h \sim \mathcal{N}(0, 1/d_k)$. Because of the ReLU, TSA can only *suppress* a training row relative to standard attention and never boost it. Training-to-training attention is left unchanged.

Implementation details:

- TSA adds $12 \times 8 \times 64^2 \approx 0.39$M parameters ($d_k = 64$).
- The bias is added through the attention mask of PyTorch's SDPA.
- To bound the memory of the $(m \times n)$ bias, test queries are processed in chunks of 2,048. During training, chunks are recomputed in the backward pass (activation checkpointing).
- At inference, the bias is computed by a fused, forward-only Triton kernel.

TSA is applied *only* in the ICL predictor. The column embedder's induced set attention has no test-to-train structure, so the TSA model uses plain attention there.

C.3 BASELINES

All baselines use the attention-modulation modules already implemented in the TabICLv2 codebase, without modification.

SSMAX. Scalable-Softmax (Nakanishi, 2025) rescales queries by the log of the context length n: $\tilde{q} = q \cdot s_h \log n$, with one learnable scale s_h per head.

QASSMAX. Query-Aware Scalable-Softmax is the default mechanism in TabICLv2 (Qu et al., 2026). We use its element-wise form:

$$\tilde{q} = q \odot \mathrm{MLP}_{\text{base}}(\log n) \odot \big(1 + \tanh(\mathrm{MLP}_{\text{query}}(q))\big), \tag{18}$$

where both MLPs have 64 hidden units and the last layer of $\mathrm{MLP}_{\text{query}}$ is zero-initialized, so the query-dependent modulation starts out as the identity. This baseline is therefore a from-scratch reproduction of TabICLv2 itself.

Vanilla attention. Standard scaled dot-product attention with no modulation, in both the column embedder and the ICL predictor.

Following Qu et al. (2026), SSMAX and QASSMAX are applied in *both* the column embedder and the ICL predictor. Table 4 summarizes the configurations.

Table 4: Attention modulation per component for each pre-trained variant.

Variant	Column embedder	ICL predictor
Vanilla	–	–
SSMAX	SSMAX	SSMAX
QASSMAX	QASSMAX	QASSMAX
TSA (ours)	–	TSA

C.4 PRE-TRAINING

All variants are pre-trained from scratch with identical data, seeds and optimization settings, following TabICLv2's classification recipe (Qu et al., 2026, Sec. 4.1, App. B.1), with the two deviations noted below (cautious weight decay and the Stage 2–3 train fraction). The runs differ only in which attention modulation is used in the column embedder and the ICL predictor (Table 4).

Prior. Synthetic datasets are sampled from TabICLv2's graph-based SCM prior (2–32 nodes, no graph noise or activation warping, filtering out unpredictable graphs and datasets). Each dataset has 1–100 features and up to 10 classes. For more details, see Qu et al. (2026).

Optimization. We use Muon (Jordan et al., 2024) with $\beta_1 = 0.9$ and weight decay 0.01. As in the released TabICLv2 checkpoints, cautious weight decay is not enabled. The learning rate follows a cosine schedule with 1% linear warm-up, decaying to 10^{-7}. The effective batch size is 64 datasets, and training uses mixed precision.

Curriculum. Pre-training has three stages that progressively increase dataset size (Table 5). Each stage starts from the final checkpoint of the previous stage, loading model weights only, so the optimizer and scheduler state are reset. Train/test sizes are sampled independently for each dataset. Following Qu et al. (2026), FlashAttention-3 (Shah et al., 2024) is enabled in Stages 2 and 3. In Stage 3, both SSMAX and QASSMAX degrade relative to their Stage 2 checkpoints; we hypothesize that this is due to their $\log n$ scaling, which sharpens the logits beyond the range seen in earlier stages as N grows to 60K.

Table 5: Pre-training curriculum, following Qu et al. (2026).. N: samples per dataset; log-uniform where a range is given.

Stage	Steps	N	Train fraction	Max LR	Grad. clip
1	500K	1,024	30–90%	8×10^{-4}	10
2	40K	400–10,240	79–81%	1×10^{-4}	10
3	10K	400–60,000	79–81%	2×10^{-5}	1

Compute. Training runs on NVIDIA H100 GPUs (80 GB): four GPUs in all three stages. Gradient accumulation keeps the effective batch size at 64 throughout, with micro-batches of 4 datasets per GPU in Stage 1 and 1 dataset in Stages 2–3. Prior sampling runs on the CPU with 12 worker processes. In Stage 3, when the largest datasets did not fit in memory (TSA, SSMAX and QASSMAX), we enabled activation recomputation. This lowers memory use without changing the computed gradients. *Wall-clock training time* is approximately 5 days for Stage 1, 1.2 days for Stage 2 and 2.9 days for Stage 3. Stage 3 is the slowest per step because of its larger datasets (up to 60K samples) and the extra forward passes from activation recomputation.

D PRE-TRAINING AND EVALUATION DETAILS ON TABDPT EXPERIMENTS

We pre-train three TabDPT v1.1 (Ma et al., 2026) checkpoints from scratch with the public TabDPT training code (https://github.com/layer6ai-labs/TabDPT-training), adding TSA as the only change to the model (Table 6). *TSA* and *Vanilla* differ only in their attention. *Vanilla* and *Vanilla + train. ret.* differ only in how the rows of a training task are selected. Everything else is shared: architecture (apart from the TSA matrices), pre-training corpus, task construction, context-size distribution, objective, optimizer, batch size, number of updates, and evaluation protocol. TSA is evaluated with the full training set in context only. The two vanilla baselines are evaluated both with full context and with retrieval.

Table 6: The three TabDPT checkpoints.

Model	Attention	Training-time retrieval	Parameters	Inference
TSA	TSA in all 16 layers	no	78.05M	full
Vanilla	standard	no	76.87M	full, ret.
Vanilla + train. ret.	standard	yes	76.87M	full, ret.

D.1 ARCHITECTURE

All checkpoints use the TabDPT v1.1 architecture: a pre-norm transformer encoder with 16 layers, width 768, 8 heads ($d_k = 96$), a feed-forward width of 1,536 with GELU, per-head LayerNorm on queries and keys, and no dropout. Each row has up to 100 features, zero-padded to 100. Features are clipped at 4σ, standardized and clipped again, missing values are set to 0, and a single linear encoder followed by RMS normalization maps the row to 768 dimensions. A linear label encoder adds the label embedding to the context rows. Keys and values come only from the context rows, while queries come from all rows, so target rows attend to context rows and never to each other. An MLP head ($768 \to 1{,}536 \to 11$) outputs 10 class logits and one regression value.

D.2 TSA IN TABDPT

In each layer and head, TSA adds a learned $W^{(h)} \in \mathbb{R}^{96\times 96}$, initialized as $W^{(h)} \sim \mathcal{N}(0, 1/d_k)$, and computes

$$F^{(h)} = \mathrm{ReLU}\left(\frac{Q\,W^{(h)}K^\top}{\sqrt{d_k}}\right), \qquad \mathrm{TSA}_h(Q,K,V) = \mathrm{softmax}\left(\frac{QK^\top}{\sqrt{d_k}} - F^{(h)}\right)V, \tag{19}$$

where K and V are the keys and values of the context rows. Because TabDPT computes the attention of all rows against the context keys in a single call, Q holds the queries of *all* rows, context and target alike. TSA therefore suppresses context keys for context queries as well as for test queries, unlike the TabICLv2 implementation, where only test queries are suppressed (subsection C.2). This adds $16 \times 8 \times 96^2 = 1{,}179{,}648$ parameters ($+1.53\%$). At inference, the bias is computed in blocks of query rows.

D.3 PRE-TRAINING

Data. Pre-training uses only real tables: the 123 OpenML datasets in the leakage-filtered training list released with TabDPT's training code. None of the 51 TabArena datasets appears in this

corpus when matched by OpenML dataset ID. Each table has its default target appended as an ordinary column, categorical columns integer-coded, missing values mean-imputed and all columns standardized.

Tasks. Each task is a sequence of 4,596 rows from one table, built as follows:

1. A table is drawn uniformly at random, and a target column is drawn uniformly among all its columns, so the original target has no special role. A column with 2–10 distinct values becomes a classification target. A column with more values becomes a regression target with probability 0.5, and otherwise a classification target with $k \in \{2, \ldots, 9\}$ classes, using $k - 1$ random cut points.
2. The features are a random subset of the remaining columns, of size uniform in $[\min(50, F/2), \min(100, F)]$ for F available columns.
3. The rows are selected differently per checkpoint. For TSA and Vanilla, 4,596 rows are drawn uniformly without replacement (tables with fewer rows are topped up by resampling). For Vanilla + train. ret., a random anchor row is drawn and the task consists of its 4,596 nearest neighbours in the standardized table (exact L_2 search), as in the original TabDPT recipe (Ma et al., 2026).
4. Rows are shuffled and class labels are randomly permuted. Regression targets are standardized, passed through two random nonlinear maps and re-standardized. A regression task is redrawn if any $|y| > 10$ or if a single value covers more than 95% of the rows.

For each forward batch, a context size m is drawn uniformly from $\{100, \ldots, 4{,}096\}$. The first m rows form the context, with labels visible, and the remaining $4{,}596 - m$ rows are targets. Pre-training contexts therefore never exceed 4,096 rows.

Optimization. The loss is computed on target rows only: cross-entropy with label smoothing 0.1 for classification and MSE for regression, weighted by the fraction of regression tasks in the batch. We use Schedule-Free AdamW (Defazio et al., 2024) with learning rate 10^{-3}, $\beta = (0.98, 0.999)$, 1,000 warm-up steps, weight decay 0.05, gradient-norm clipping at 1.0 and bf16 mixed precision. Each update uses 128 tasks, accumulated over 4 steps of 32 tasks, and each accumulation step draws its own context size. All three checkpoints are taken after 64,000 updates (8.19M tasks), with seed 42. For memory reasons, the TSA run splits each accumulation step into smaller sub-batches and uses activation checkpointing. Both these memory helpers leave the objective and the update mathematically unchanged. Validation during training was used for monitoring only, and no checkpoint or hyperparameter was selected on evaluation data.

D.4 Evaluation

The main text (Table 2) reports the 10 classification datasets of TabArena (Erickson et al., 2025) with more than 10,000 training samples, the limit up to which TabArena runs TabPFNv2. The full context is always longer than any context seen in pre-training (Table 7). We use TabArena's benchmarking code. Each training split holds 2/3 of the rows, between 12,772 and 100,000. Each configuration is a single model fit on the whole training split, without hyperparameter tuning.

At inference, categorical features are label-encoded, missing values mean-imputed and features standardized. Tables with more than 100 features (APSFailure, KDDCup09_appetency) are reduced to 100 by PCA. Predictions average the logits of 8 ensemble members, each with a random feature and class-label permutation, at temperature 1. With *full context*, all training rows form the context, up to 100,000 rows. With *retrieval* (vanilla baselines only), each test row's context is its 4,096 nearest training rows (exact L_2 search on the standardized features). In both settings, test rows attend only to context rows, so predictions do not depend on batching.

The error is $1 -$ ROC-AUC for binary and log-loss for multiclass tasks. Improvability, average rank, Elo, and win rates are computed over the datasets of each table with TabArena's evaluation code, and ranks are among the five evaluated configurations. Elo is calibrated to 1000 for TabArena's default random forest, with bootstrap confidence intervals over 200 rounds. The win rate against Vanilla is the fraction of datasets on which a configuration's error is lower than that of Vanilla with full context.

Table 7: TabArena classification datasets whose training split exceeds 10,000 rows. *Tr. rows* is the size of the training split, i.e., the full context.

Dataset	Rows	Tr. rows	Features	Classes	Metric
GIVEMESOMECREDIT	150,000	100,000	10	2	ROC-AUC
CUSTOMER_SATISFACTION_IN_AIRLINE	129,880	86,586	21	2	ROC-AUC
SDSS17	78,053	52,035	11	3	log-loss
APSFAILURE	76,000	50,666	170	2	ROC-AUC
DIABETES130US	71,518	47,678	47	2	ROC-AUC
KDDCUP09_APPETENCY	50,000	33,333	212	2	ROC-AUC
BANK-MARKETING	45,211	30,140	13	2	ROC-AUC
AMAZON_EMPLOYEE_ACCESS	32,769	21,846	9	2	ROC-AUC
CREDIT_CARD_CLIENTS_DEFAULT	30,000	20,000	23	2	ROC-AUC
HR_ANALYTICS_JOB_CHANGE_OF_DATA_SCIENTISTS	19,158	12,772	12	2	ROC-AUC

E DETAILS ON ROW DROPPING

E.1 IMPLEMENTATION

Rows are dropped by retrieving the indices of the surviving rows rather than by logit masking. Masking the logits of the dropped rows to $-\infty$ would give the test rows the same attention output as removing those rows, but every subsequent layer would still process all n_{train} rows. Instead, at a pruning layer we select the k rows with the largest s_j for eviction, build a boolean keep mask over the remaining training rows, and use `index_select` to materialize a smaller hidden-state tensor. The variable `current_train_size` shrinks accordingly, so all remaining blocks operate on a smaller context. After the final block, the dropped positions are zero-padded back, so the output has its original shape and test predictions are sliced from `original_train_size` exactly as without pruning.

This makes pruning an amortized operation. A pruning layer pays a tracking cost (retaining F, one reduction to s_j, a top-k, and one copy of the hidden states), which is repaid by every layer after it running on a smaller context. Pruning at every layer lets the cost dominate the inference overhead, and pruning only near the end leaves too few layers to amortize over. The schedule controls this trade-off, along with how well-calibrated the scores are at the decision layer.

E.2 DEFAULT SCHEDULE

Let $\mathcal{B} = (B_1, \ldots, B_L)$ be the ordered blocks of the ICL predictor and let n_{train} be the number of training rows entering that block. At a given block, $F_{ij}^{(h)} \geq 0$ is the TSA suppression of training row j by test row i in head h, and s_j is its aggregate equation 11, over which μ_s and σ_s denote the mean and standard deviation across the rows still in context. We write $\mathcal{P} \subset \mathcal{B}$ for the *pruning layers*, the blocks after which rows are evicted, and b_B for the number of rows dropped at $B \in \mathcal{P}$. The schedule is specified by three hyperparameters: the warm-up fraction w, the total budget $p \in [0, 1)$, and the gate width $z \geq 0$. Our default schedule combines three elements:

(i) Warm-up skip. The first 25% of blocks never prune. Early representations are not yet discriminative enough for the bilinear score to be a reliable eviction signal, and a mistake made there propagates through the whole remaining layer stack.

(ii) Shared budget over pruning layers. A single global budget of $p \cdot n_{\text{train}}$ rows is fixed in advance and split across a small set of pruning layers $\mathcal{P}$, evenly spaced over the post-warm-up range and excluding the final block, where dropping provides no speedup. The split uses cumulative rounding of (optionally non-uniform) weights, so that $\sum_{B \in \mathcal{P}} b_B = p \cdot n_{\text{train}}$ exactly. Weighting the later layers more heavily trades some speed for decisions taken on more reliable scores. Crucially, the budget is defined against the *original* n_{train}, so the fraction removed is exactly p regardless of $|\mathcal{P}|$.

(iii) Confidence gating. At every pruning layer but the last, a row is dropped only if its score is a statistical outlier among the current candidates, $s_j > \mu_s + z\,\sigma_s$. Any budget that cannot be justified

this way is *deferred* to the next pruning layer, where the scores are better informed. The last pruning layer is never gated: it spends its own budget plus everything deferred, by plain top-k, so the global target p is always met.

E.3 Ablated schedules

Figure 6 reports only three variants, each removing or replacing one component of the default schedule.

1. **No confidence gate** ($z = 0$): every pruning layer spends its budget by plain top-k, with no deferral to subsequent layers. This isolates the value of postponing uncertain decisions to deeper layers.
2. **No warm-up skip**: pruning layers are spread over the whole stack, so the first eviction happens on the earliest representations. This schedule tests whether early scores are reliable enough to act on.
3. **Layer-pct budget (compounding)**: instead of one shared budget, a fixed rate p_{ℓ_i} is applied independently at every pruning layer to the context *remaining* at that layer. Applying the same rate to a context that is already smaller compounds it, so the total fraction removed is $1 - (1 - p_\ell)^{|\mathcal{P}|}$ and grows with the number of pruning layers, instead of being fixed at p.