ADAPTING LINEAR-TIME ARCHITECTURES FOR TABULAR IN-CONTEXT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Tabular foundation models achieve strong performance by conditioning on labelled examples in context, but softmax attention limits their use on large datasets. Existing linear-time alternatives, however, are mostly causal, and their potential for tabular in-context learning (ICL) remains underexplored. To address this, we (1) revisit causal training setups, (2) compare linear sequence mixers, and (3) investigate their ICL generalisation beyond the pretraining context length. First, we show that the best training setup for causal models resembles next-token prediction. Then, perhaps surprisingly, the most promising linear sequence mixer is causal: DeltaNet outperforms even non-causal linear attention. However, it degrades beyond 2-$4\times$ the pretraining context length, and existing mitigation strategies such as bidirectionality defer the problem at best. A hidden-state oracle shows that this is not a capacity problem. Instead, our analysis points to an instability in the recurrent state, which drifts in deeper layers of causal models. Since DeltaNet's learned write rates overfit to the pretraining regime, we modulate them with a time-dependent decay schedule intervention to stabilise length generalisation. Finally, re-introducing non-causality by reading out from the final state allows us to closely match a controlled softmax attention baseline on OpenML-CC18 and TabArena.

1 INTRODUCTION

Despite the success of deep learning methods on text and images, gradient-boosted decision trees have long remained state-of-the-art for tabular data (Grinsztajn et al., 2022). Prior-data fitted networks (PFNs) (Müller et al., 2022) challenge this by recasting supervised learning as in-context prediction from a labelled context set. Models in this family, including TabPFN (Hollmann et al., 2023; 2025), TabICL (Qu et al., 2025), and TabDPT (Ma et al., 2025), now match or exceed boosted-tree baselines without task-specific training (Erickson et al., 2026).

Most of these so-called tabular foundation models rely on softmax self-attention (Vaswani et al., 2017). Internally, predictions are non-parametric and require pairwise token interactions with all in-context training samples. Particularly for large datasets, linear-time architectures could present an attractive alternative as they compress the context into a fixed-size parametric state. This potential has been confirmed by pioneering works which replaced softmax attention by linear-time sequence mixers (Zeng et al., 2025; Koch et al., 2025; Song et al., 2026; Baur & Kim, 2024), but the design of the core memory mechanism has not been disentangled from other training and architectural decisions. Modern linear recurrences such as DeltaNet update a fast-weight memory from past key-value pairs (Schlag et al., 2021), which can be interpreted as online optimisation to compress the observed context (Wang et al., 2025). For tabular ICL, however, the fundamental question is if and how such potentially ordered updates can aggregate information across exchangeable rows in tables.

To that end, we adapt recurrent sequence mixers to tabular ICL rather than treating them as drop-in replacements for softmax attention. We perform a controlled study to disentangle training design, information flow, and update rules using matched parameter counts, a shared synthetic prior, and evaluation across the benchmarks OpenML-CC18 (Bischl et al., 2021) and TabArena (Erickson et al., 2026). To quantify the generality of the learned ICL algorithm, we analyse models beyond the pretraining length and characterise novel failure mechanisms. Finally, we show how to adapt the ordered delta-rule to improve long-context stability and outperform order-invariant linear-time baselines, even on order-invariant tabular data. Our contributions are:

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.

- **Training design space.** We identify label embedding and target formulation as two training axes used across prior work, and evaluate the resulting four strategies in Figure 1 under a unified protocol. Interleaved Multi-Target (Int-MT) emerges as the strongest training strategy for causal and Combined Single-Target (Comb-ST) for non-causal models.
- **Controlled recurrence comparison.** We compare linear attention, Gated Linear Attention, DeltaNet, Gated DeltaNet, and Mamba-2 at matched scale. Delta-rule updates outperform other causal recurrences as well as non-causal linear attention at moderate sequence lengths.
- **Length-generalisation failure.** All recurrent variants degrade beyond roughly 2-4$\times$ the pretraining context length, while order-invariant models remain stable. Common mitigations, including bidirectionality, state weaving (Moroshan et al., 2025), mimetic initialisation (Trockman et al., 2025), and state passing (Buitrago & Gu, 2025), do not solve the issue.
- **Failure mechanisms and stabilisation.** A hidden-state oracle shows that fixed-size state capacity is not the bottleneck. Instead, causal degradation is accompanied by directional state drift in deeper layers. In DeltaNet we additionally identify overfitting of the write-rate β_t to the sequence length and mitigate it by introducing an explicit decay term to β_t.
- **Final-State DeltaNet.** To remove the readout-level causal bottleneck, we propose a cheap but effective extension where every query reads directly from the stabilised final state.

Code is available at https://anonymous.4open.science/r/Linear-Tabular-ICL.

2 Background

Prior-data fitted networks (PFNs) (Müller et al., 2022) cast supervised learning as conditional prediction from a context set. Given labelled context examples and unlabelled query points, a PFN predicts query labels in a single forward pass, without fitting task-specific parameters at test time. TabPFN (Hollmann et al., 2023) applies this idea to tabular classification.

We consider multi-class classification datasets mapping n inputs of m features to K classes, $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{n}$ with $x_i \in \mathbb{R}^m$ and $y_i \in \mathcal{Y} = \{1, \dots, K\}$, split into a labelled context set $\mathcal{C}$ of n_c rows and a query set $\mathcal{Q}$ of n_q rows. Then, a tabular foundation model f_θ maps the labelled context set $\mathcal{C}$ and an unlabelled query sample x_i to a predictive class distribution $\hat{p}_{\theta,i} = f_\theta(\mathcal{C}, x_i) \in \Delta^{K-1}$.

During pretraining, datasets are sampled from a meta-distribution $p(\mathcal{D})$ over synthetic or real datasets, and the model is trained to predict held-out targets using the cross-entropy (CE) loss, i.e.

$$\mathcal{L}(\theta) = \mathbb{E}_{\mathcal{D}\sim p(\mathcal{D})}\mathbb{E}_{(\mathcal{C},\mathcal{Q})\sim \mathrm{Split}(\mathcal{D})}\left[\frac{1}{n_q}\sum_{(x_i,y_i)\in\mathcal{Q}} \mathrm{CE}(f_\theta(\mathcal{C}, x_i), y_i)\right]. \tag{1}$$

As illustrated in Figure 1, query labels are typically hidden from the input context using a learned placeholder embedding y_{pad}, but causal visibility could also be used as we explore in this paper.

2.1 Scaling Bottleneck of Transformers

Most tabular foundation models use Transformer backbones (Vaswani et al., 2017). This means that the model internally relies on non-parametric estimation using softmax attention to compute pairwise interactions between samples, similar to nearest-neighbour methods. Processing a context dataset $\mathcal{C}$ thus requires quadratic computation in the number of samples n_c. Predicting $\hat{p}_{\theta,i}$ from a single query sample x_i still requires loading and iterating over the KV-cache of all n_c samples. These scaling properties limit the applicability of attention-based models to large datasets. Linear-time architectures, in contrast, promise to compress the context $\mathcal{C}$ into a parametric state $\boldsymbol{S}_{n_c}$ and predict in constant time and memory (see Appendix E), but it remains unclear if a generalising state exists and how to construct it.

2.2 Linear Attention and Recurrent Architectures

Linear attention (Katharopoulos et al., 2020) replaces the softmax kernel with a feature map $\phi(\cdot)$, s.t.

$$\boldsymbol{o}_i = \frac{1}{Z_i}\sum_{j\in\mathcal{V}(i)} \phi(\boldsymbol{q}_i)^\top \phi(\boldsymbol{k}_j)\boldsymbol{v}_j = \frac{1}{Z_i}\boldsymbol{S}_{\mathcal{V}(i)}\phi(\boldsymbol{q}_i), \qquad \boldsymbol{S}_{\mathcal{V}(i)} = \sum_{j\in\mathcal{V}(i)} \boldsymbol{v}_j\phi(\boldsymbol{k}_j)^\top, \tag{2}$$

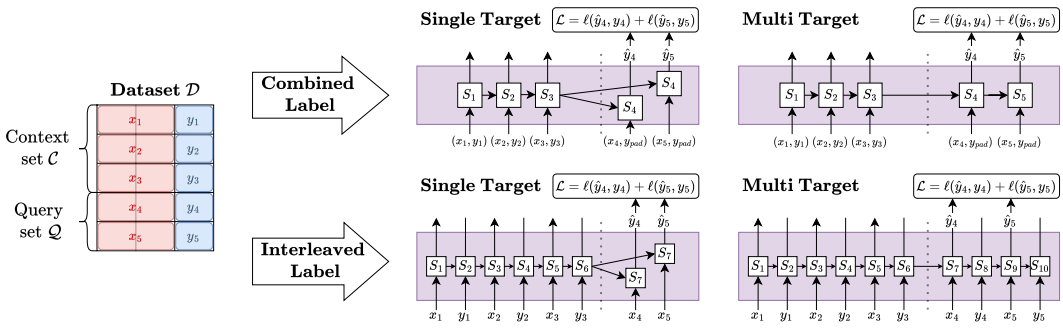


Figure 1: Training strategies given by label embedding and target formulation. Combined-label variants represent features and labels in one token; interleaved variants represent them separately. Single-target variants predict each query independently from the context; multi-target variants process target positions sequentially. Combined-label single-target is the standard (Tab)PFN setup.

where $\boldsymbol{q}_i, \boldsymbol{k}_j \in \mathbb{R}^{d_{qk}}$, $\boldsymbol{v}_j \in \mathbb{R}^{d_v}$ are per-head queries, keys, and values obtained by linear projection of the token embeddings, $\mathcal{V}(i)$ denotes the set of visible tokens, and Z_i is a normalisation term. For non-causal attention, all queries process the same fixed-size state $\boldsymbol{S}_{n_c} = \boldsymbol{V}_{\mathcal{C}}^{\top} \phi(\boldsymbol{K}_{\mathcal{C}}) \in \mathbb{R}^{d_v \times d_{qk}}$ as a compressed key-value cache with $\boldsymbol{V}_{\mathcal{C}} \in \mathbb{R}^{n_c \times d_v}$ and $\boldsymbol{K}_{\mathcal{C}} \in \mathbb{R}^{n_c \times d_{qk}}$ stacking the context values and keys and ϕ applied row-wise. For causal linear attention the state evolves as the linear recurrence

$$\boldsymbol{S}_i = \boldsymbol{S}_{i-1} + \boldsymbol{v}_i \phi(\boldsymbol{k}_i)^{\top}.$$

Modern linear-time recurrences mostly differ in how the memory state is updated. We focus on the most common update rules: Gated Linear Attention (GLA) (Yang et al., 2024a) adds an input-dependent forget gate, DeltaNet (Schlag et al., 2021) uses an overwriting mechanism based on the delta-rule, Gated DeltaNet (Yang et al., 2025) combines delta-rule updates with gating, and Mamba-2 (Dao & Gu, 2024) admits a recurrent update through its state-space dual form.

2.3 Causal Linear Recurrences as Layer-Local Test-Time Regressors

The update rules above admit a common interpretation: the state is updated online from past key-value pairs. More broadly, related test-time regression and test-time training frameworks interpret sequence-model states as associative memories or weights adapted at inference time (Wang et al., 2025; Sun et al., 2025; Behrouz et al., 2025; von Oswald et al., 2026). The test-time regression (TTR) view (Wang et al., 2025) makes this connection explicit by interpreting the recurrent state as an online approximation to a layer-local associative-memory problem:

$$\boldsymbol{S}_t^{\star} = \operatorname*{argmin}_{\boldsymbol{S} \in \mathbb{R}^{d_v \times d_{qk}}} \frac{1}{2} \sum_{i=1}^{t} \|\boldsymbol{v}_i - \boldsymbol{S}\boldsymbol{k}_i\|_2^2. \tag{3}$$

In a related linear-regression analysis, causal in-context learners converge with increasing depth to potentially suboptimal stationary estimators corresponding to online gradient descent (Ding et al., 2024). For order-invariant tabular data, this motivates investigating whether the recurrent state converges to a stationary solution of (3) as more samples are added to the context. We investigate this in Section 4.

2.4 Tabular Foundation Models with Linear Backbones

Recent work replaces the softmax backbone of tabular foundation models with linear-time alternatives. TabFlex (Zeng et al., 2025) swaps non-causal linear attention into TabPFN v1. Koch et al. (2025) replace transformer layers with Mamba (Gu & Dao, 2024) and Hydra (Hwang et al., 2024) in the same backbone. Baur & Kim (2024) explore autoregressive Mamba and Jamba (Lenz et al., 2025) with interleaved feature–label tokens. FEAT (Song et al., 2026) stacks bidirectional Mamba with interleaved convolutional gated linear-attention layers for large structured data. Each work adopts a different combination of sequence mixer, label embedding, and target formulation, making it difficult to attribute gains to any single choice. Our study disentangles these axes and investigates length generalisation in order to quantify the generality of the learned ICL solution. Finally, we propose a principled solution to adapt the best-performing state construction mechanism to the tabular data setting.

3 Disentangling Training Strategies and Sequence Mixers

To capture the order-invariant nature of tabular data better, tabular models are usually non-causal with visibility $\mathcal{V}_{\text{Non-Causal}}(i) = \{1, \dots, n_c\} \cup \{i\}$ and no positional encodings. In contrast, modern linear-time architectures are designed to be causal with visibility $\mathcal{V}_{\text{Causal}}(i) = \{1, \dots, i\}$ and a strong positional bias. Reconciling the two seems almost impossible since even non-causal extensions are not order-invariant. While TabFlex (Zeng et al., 2025) circumvents this by falling back to non-causal vanilla linear attention, we will now question the order-invariance assumption. Specifically, we first ask which training strategy is most effective for causal models (Section 3.2) and then which sequence mixers, including causal ones, perform best under matched parameter counts (Section 3.3).

3.1 Pretraining and Evaluation Protocol

To isolate how linear and recurrent models perform in tabular ICL, we keep the data-generating prior, training budget, evaluation protocol, and model sizes consistent across architectures. All models are pretrained exclusively on synthetic datasets sampled from the TabPFN v1 prior (Hollmann et al., 2023), containing at most $n = 1000$ rows and $m = 20$ input features unless stated otherwise. All architectures are matched to 12.5M parameters to ensure a fair comparison (deviations below 1%; details in Appendix B.1). Most hyperparameters are reused from TabPFN v1, while backbone-specific hyperparameters are selected on validation performance on newly sampled synthetic datasets.

We evaluate on the 30-dataset OpenML-CC18 (Bischl et al., 2021) subset of TabPFN v1, as well as the more modern TabArena classification datasets (Erickson et al., 2026). All datasets are subsampled to at most $m = 20$ features. OpenML-CC18 is further subsampled to at most $n = 1000$ samples to match the pretraining size, while TabArena measures real-world length generalisation on larger datasets. We denote these preprocessed variants as OpenML-CC18* and TabArena*. Following prior work, we report ROC-AUC and accuracy aggregated over 5-fold stratified CV splits and datasets. For pairwise significance, we use paired Wilcoxon signed-rank tests (Wilcoxon, 1945) with Holm correction (Holm, 1979). Further evaluation details are provided in Appendix C.

3.2 Training Strategies

It is not a priori clear whether design decisions made for tabular Transformers also transfer to causal variants. To that end, Baur & Kim (2024) and Koch et al. (2025) modify the training strategy but without a controlled comparison. We identify two axes, the label embedding and the target formulation.

The label embedding determines row serialisation. The combined-label representation embeds features and labels into one token, while the interleaved representation embeds them in separate tokens:

$$\text{combined:} \quad \text{emb}_i = \psi_x(x_i) + \psi_y(\tilde{y}_i), \qquad L = n_c + n_q, \tag{4}$$

$$\text{interleaved:} \quad \text{emb}_{2i-1} = \psi_x(x_i), \quad \text{emb}_{2i} = \psi_y(y_i), \qquad L = 2(n_c + n_q), \tag{5}$$

where ψ_x and ψ_y are embedding functions, $\tilde{y}_i = y_i$ for context rows, and $\tilde{y}_i = y_{\text{pad}}$ for query rows. In the interleaved representation, labels are processed independently from feature tokens and therefore query labels can be held out with appropriate attention masking instead of input padding y_{pad}.

The target formulation determines whether the query positions are treated independently or sequentially during pretraining. At the row level, single-target training isolates queries from each other, whereas multi-target training assumes ordering through causal visibility:

$$\mathcal{V}_{\text{ST}}(i) = \{1, \dots, n_c\} \cup \{i\}, \qquad \mathcal{V}_{\text{MT}}(i) = \{1, \dots, n_c\} \cup \{n_c + 1, \dots, i\}. \tag{6}$$

Recurrent architectures require careful adjustments to support single-target since all queries must condition on the same context state $\boldsymbol{S}_{n_c}$, while their predictions remain independent. We describe this —to the best of our knowledge— novel efficient implementation in Appendix F.

Figure 1 illustrates the resulting four strategies: Comb-ST is typical for PFN-style softmax-attention models and TabFlex (Zeng et al., 2025), Comb-MT is used by Koch et al. (2025), and Int-MT/Int-ST are used by Baur & Kim (2024) for pretraining and prediction, respectively. To investigate the compatibility of causal linear sequence mixers with these training strategies, we aggregate the performance of four common models for each setup. We exclude standard linear attention because it cannot handle interleaved tokens without positional embeddings. To guarantee a fair comparison, we

Table 1: Aggregated training setup performance on OpenML-CC18* and TabArena*, the preprocessed benchmark variants from Section 3.1. Values are averaged across splits, datasets, and the four model families supporting all four setups: Gated Linear Attention, DeltaNet, Gated DeltaNet, and Mamba-2. Len. is the sequence length relative to combined-label models. Bold marks the best setup per metric. The vertical bars denote insignificance classes per benchmark metric; setups sharing a bar are not significantly different under the paired Wilcoxon signed-rank test with Holm correction at $\alpha = 5\%$.

Training setup	Len.	OpenML-CC18* ROC-AUC ↑	OpenML-CC18* ACC ↑	TabArena* ROC-AUC ↑	TabArena* ACC ↑
Interleaved Multi-Target	2×	**0.8837**	**0.8039**	**0.8038**	**0.8398**
Combined Single-Target	1×	0.8824	0.8020	0.8004	0.8370
Interleaved Single-Target	2×	0.8800	0.7980	0.8005	0.8350
Combined Multi-Target	1×	0.8793	0.7952	0.7978	0.8338

evaluate all models non-transductively, i.e. every queried test sample is predicted independently of the others as in single-target training.

The results in Table 1 highlight that the training setup indeed needs to be considered jointly with the sequence mixer. Overall, Interleaved Multi-Target is most compatible with causal linear-time models. We explain the strength of interleaving and multiple targets by a teacher-forcing effect similar to next-token prediction: feature-target pairs from the query set $\mathcal{Q}$ enter the context of later queries and effectively extend their context set $\mathcal{C}$ during pretraining. In comparison, interleaving with only single targets clearly deteriorates. For combined embeddings, however, multiple targets likely underperform due to a distribution shift because the placeholder labels y_{pad} can appear in the context set $\mathcal{C}$ during pretraining but not at inference. Finally, the Combined Single-Target setting remains very competitive considering that interleaving requires doubling the sequence length and thus the compute cost. It is also the only setting where causal models can be fairly compared to non-causal ones, and we therefore adopt the Comb-ST setting in the rest of this paper. We will, however, compare a causal Int-MT to a non-causal Comb-ST model in the final Table 3.

3.3 Sequence Mixers

Linear attention based architectures usually share the readout mechanism $\boldsymbol{o}_i = \boldsymbol{S}_i \boldsymbol{q}_i$ and differ in the state construction. The most common ingredients for recurrent update rules on top of vanilla linear attention $\boldsymbol{S}_t = \boldsymbol{S}_{t-1} + \boldsymbol{v}_t \boldsymbol{k}_t^\top$ (Katharopoulos et al., 2020) are gating $\boldsymbol{S}_t = \boldsymbol{S}_{t-1}\text{Diag}(\alpha_t) + \boldsymbol{v}_t \boldsymbol{k}_t^\top$ with a forgetting strength α_t (Yang et al., 2024a), the delta rule $\boldsymbol{S}_t = \boldsymbol{S}_{t-1}\left(\mathbf{I} - \beta_t \boldsymbol{k}_t \boldsymbol{k}_t^\top\right) + \beta_t \boldsymbol{v}_t \boldsymbol{k}_t^\top$ which reduces memory collisions based on a write strength β_t (Schlag et al., 2021), or combinations thereof (Yang et al., 2025). Note that these updates are not order-invariant. To isolate the effect of this core memory mechanism, we compare these update rules in the Comb-ST setting. We further train a parameter-matched softmax attention baseline and report full-scale tabular baselines for context.

Table 2 shows that non-causal softmax attention is the strongest controlled architecture, confirming that simply replacing it with linear or recurrent updates reduces performance on both benchmarks. Among linear sequence mixers, the causal DeltaNet and its gated variant perform best overall. This indicates that delta-rule updates transfer better to tabular ICL than vanilla linear attention. Gating alone does not meaningfully change performance, and Mamba-2 trails the rest across all metrics. Within linear attention, the non-causal version only outperforms its causal counterpart significantly on OpenML-CC18 ROC-AUC. Together, these results suggest that the update rule of recurrent sequence mixers is more important than causality or order-invariance.

We want to highlight that causal DeltaNet and non-causal softmax attention remain competitive with non-causal TabFlex (Zeng et al., 2025) on OpenML-CC18 despite our far smaller pretraining budgets. On TabArena, however, DeltaNet deteriorates compared to the baselines. Recall that this benchmark includes datasets larger than our 1000-row pretraining cap to measure real-world length generalisation. Figure 4 in the appendix confirms that the gap to softmax attention generally grows with increasing dataset size. In Section 4 we will investigate this length-generalisation failure in more detail, relying on the synthetic prior to allow for finer context-length coverage than TabArena.

Table 2: Model comparison on the preprocessed OpenML-CC18* and TabArena* benchmark variants (Section 3.1). The upper block reports full-scale released baselines, not matched in parameter count or pretraining budget, while the lower block reports our controlled backbones, all using Combined Single-Target. Values are averaged over 5-fold stratified CV splits and datasets. Bold and underline mark the best and second-best controlled models per metric. The vertical bars denote pairwise insignificance classes per metric; models sharing a bar are not significantly different.

	OpenML-CC18*		TabArena*	
	ROC-AUC ↑	ACC ↑	ROC-AUC ↑	ACC ↑
TabPFN v2.5	0.8993	0.8298	0.8383	0.8642
TabICLv2	0.8971	0.8286	0.8375	0.8631
TabFlex	0.8859	0.8093	0.8089	0.8463
CatBoost	0.8833	0.8062	0.8231	0.8562
Softmax Non-Causal	**0.8900**	**0.8166**	**0.8172**	**0.8506**
Delta	<u>0.8860</u>	<u>0.8078</u>	0.8074	0.8388
Delta Gated	0.8857	0.8067	<u>0.8078</u>	<u>0.8414</u>
Linear Non-Causal	0.8833	0.8039	0.8004	0.8396
Linear Gated	0.8827	0.7985	0.8038	0.8388
Linear Causal	0.8817	0.7988	0.8033	0.8390
Mamba-2	0.8784	0.7951	0.7912	0.8344

4 Understanding Generalisation Failure

We now analyse why the competitive short-context performance of causal models does not generalise to larger context sets. To that end, we sample 500 synthetic datasets of size 128k from which we construct query sets with fixed $n_q = 100$ queries and controlled context sets of increasing size n_c.

4.1 Length Generalisation Gap and Hidden-State Capacity

Figure 2(a) compares the causal and non-causal variants of softmax and linear attention beyond the 1000-row pretraining regime. Consistent with Zeng et al. (2025), causal models degrade as the number of samples increases, from 2-4× the pretraining range, with severity varying by update rule: causal additive linear attention degrades only moderately, while more expressive rules such as DeltaNet and GLA degrade more sharply. In contrast, non-causal attention continues to generalise, suggesting that causal prefix access is one important source of the failure as we will analyse further in Section 4.3.

To test whether additional samples could improve prediction, we partition the context into blocks of 3K samples and ensemble predictions for each query across the blocks. This protocol already allows DeltaNet to consistently outperform non-causal linear attention. It shows that DeltaNet could benefit from additional samples, but fails to effectively aggregate them into states $\boldsymbol{S}_{n_c}$ over long sequences.

Still, it remains open whether the fixed-size hidden state even provides the capacity to exploit additional samples compared to softmax attention. To test this hypothesis, we design a hidden-state oracle for the pretrained and frozen DeltaNet checkpoint. We then optimise only its hidden-states $\boldsymbol{S}$ with gradient descent such that the end-to-end model fits the 128k context samples (details in Appendix B.4). In Figure 2(a), the oracle achieves strong long-context performance, demonstrating that limited memory capacity is not the bottleneck: there exists a parametric state $\boldsymbol{S}$ which even outperforms our non-parametric softmax attention baseline, but existing linear models do not construct it.

We use this end-to-end oracle as a diagnostic tool, but it would be valid under the non-transductive evaluation protocol. In Appendix D.2 we show that performance transfers to the five largest TabArena datasets. Since hidden-state optimisation requires careful length-dependent tuning, we leave practical deployment to future work. Approximating the end-to-end test-time objective with the layer-local regression objective of Equation 3 further allows computing an optimal state $\boldsymbol{S}^{\text{ridge}}$ from a permutation-invariant, closed-form ridge regression. In Appendix D.3 we show that such a sequence mixer can be trained and generalises, supporting the hypothesis that stable state construction is the key challenge.

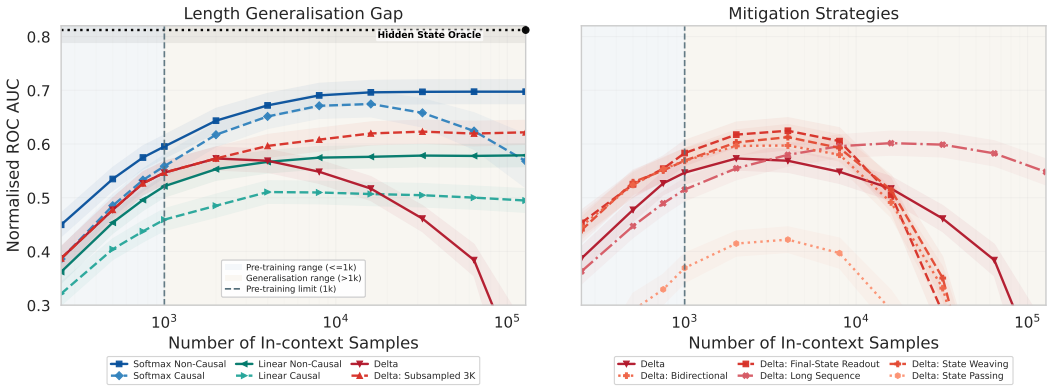


Figure 2: Length generalisation beyond the 1k pretraining length. We report normalised ROC-AUC averaged over 500 synthetic datasets with 95% confidence intervals. **(a) Length Generalisation Gap:** Causal models underperform their non-causal counterparts. DeltaNet is initially competitive but degrades more strongly, while subsampled DeltaNet remains stable; the oracle is evaluated only at 128k and shown as a horizontal reference. **(b) Mitigation Strategies:** None of the applicable strategies eliminates DeltaNet's long-context degradation (Section 4.2).

4.2 Mitigation Strategies

We evaluate several mitigation strategies for causal recurrent models. *Bidirectionality* (Afzal et al., 2025) processes the context in both directions and writes to the same hidden states $\boldsymbol{S}_i$. Our *final-state readout* modification (Section 5) reads from the same final context state $\boldsymbol{S}_{n_c}$ at all positions $\boldsymbol{q}_i$, allowing for non-causal information flow through the aggregated final state. *State weaving* (Moroshan et al., 2025) initialises each layer with the final hidden state of the previous layer to introduce non-causality. *State passing* (Buitrago & Gu, 2025) reuses cached recurrent states from prior batches during pretraining. *Long-sequence pretraining* biases the row sampler toward longer contexts while remaining compute-matched by sampling smaller datasets more frequently.

Figure 2(b) shows all applicable mitigations for DeltaNet; results for GLA are in Appendix D.5. Overall, none of these strategies provides a general solution. Bidirectionality, final-state readout, and state weaving improve performance near the pretraining range, indicating that non-causal information flow is important. However, they do not solve length generalisation. While non-causality stabilises vanilla linear attention, it increases degradation in DeltaNet. Length degradation is therefore not fully explained by causal access alone: a hidden state may itself be an unstable estimator of the task, even when all positions write to or read from it. Long-sequence pretraining shifts degradation to longer contexts at the cost of short-context performance, and state passing is recurrence-dependent: it reduces degradation for GLA, but lowers overall performance for DeltaNet.

4.3 Hidden-State Drift in Causal Linear Attention

We next analyse the recurrent hidden states as a function of sequence length, focusing on vanilla causal linear attention as the simplest example reproducing the failure and comparing it with non-causal linear attention, which generalises in our setting. Neither model uses positional encodings or the normalisation $\frac{1}{Z_i}$ from Equation 2. Visualisations are provided in Appendix D.6. Comparing the Frobenius norm of the final state $\boldsymbol{S}_{n_c}$ for increasing context sizes n_c (Figure 8, second panel), the causal model's hidden-state norms drift substantially relative to one another across layers as the context grows, whereas non-causal linear attention exhibits approximately parallel norm growth. If magnitude drift were the cause, controlling the state norm should restore performance. Training with Frobenius normalisation of each layer's state matrix does not remove the degradation (Figure 8, first panel), so magnitude drift is not sufficient to explain the failure.

We then measure directional drift by comparing each normalised hidden state $\boldsymbol{S}_{n_c}$ to the corresponding state $\boldsymbol{S}_{1000}$ on the same dataset at pretraining length $n_c = 1000$, using cosine similarity after flattening the state matrix. Figure 8 (bottom panel) shows that non-causal linear attention approaches a stable hidden state, whereas causal linear attention drifts increasingly far from $\boldsymbol{S}_{1000}$, with performance degradation occurring alongside the drift. The drift appears in earlier layers and becomes substantially

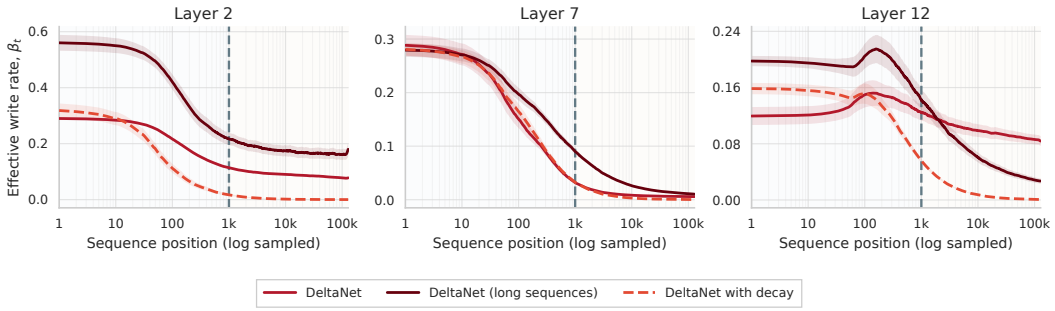


Figure 3: Effective write rates β_t^{eff} over sequence positions across network depth. We report means over 50 datasets, including the 95% CIs. The dashed vertical line marks the 1000-row pretraining limit, with additional layer plots provided in Figure 9 of the appendix. Standard DeltaNet learns a decreasing write-rate profile, but compared to the (not compute-matched) long-sequence pretrained variant, it lowers the write rates less aggressively at very deep layers. The decay variant is our proposed solution, which imposes the decay explicitly within the recurrence relation.

stronger in deeper layers, consistent with accumulating deviations across successive causal layers. Similar drift appears across different data-generating priors whenever the pretraining distribution is sufficiently diverse, the target multi-class, and the model is sufficiently deep. This suggests a failure mode beyond the online-gradient-descent interpretation of causal linear self-attention (Ding et al., 2024), which does not account for the persistent directional drift we observe.[1]

4.4 DeltaNet Length Overfitting and Write-Rate Decay

The hidden-state drift alone does not explain why causal additive linear attention degrades only moderately in Figure 2(a) while DeltaNet degrades much more sharply. Note that the recurrence of additive linear attention forms an order-invariant set aggregate for fixed layer-local keys and values and corresponds to a single gradient step on the objective of Equation 3. The delta-rule, on the other hand, forms an ordered aggregate of first-order updates on per-sample losses (Wang et al., 2025). Its endpoint $\boldsymbol{S}_{n_c}$ is therefore the result of an ordered online optimisation trajectory controlled by the update rule $\boldsymbol{S}_t = \boldsymbol{S}_{t-1}\left(\mathbf{I} - \beta_t \boldsymbol{k}_t \boldsymbol{k}_t^\top\right) + \beta_t \boldsymbol{v}_t \boldsymbol{k}_t^\top$ with its learned write rates β_t.

Classical stochastic approximation motivates decaying the effective step sizes to achieve stable long-horizon convergence (Robbins & Monro, 1951). In contrast, DeltaNet learns its write-rate schedule from pretraining on sequences of at most 1000 rows. Figure 3 analyses the learned write rates: standard DeltaNet learns a decreasing schedule by inferring position from the evolving causal representations. In deeper layers, the long-context pretrained variant lowers its late-sequence write rates substantially, coinciding with better long-context performance. Overall, the figure indicates that DeltaNet overfits its update dynamics to the pretraining length, resulting in excessively large updates at long sequence lengths, which motivates a concrete intervention. We introduce a write-rate decay to explicitly decay the effective write rate over time by setting $\beta_t^{\text{eff}} = \beta_t \eta_t$ with $\eta_t = c/(t+c)$, where $c = 256$ is selected on synthetic pretraining performance. With this decay, DeltaNet continues to improve beyond the pretraining length instead of collapsing and exhibits only minor degradation at long sequence lengths (full analysis in Appendix D.7).

5 Final-State Readout DeltaNet with Write-Rate Decay

Having stabilised DeltaNet for long contexts, we now introduce our non-causal readout extension to the architecture. *Final-state readout* enables non-causal information flow by reading from the final state $\boldsymbol{o}_i = \boldsymbol{S}_{n_c} \boldsymbol{q}_i$ instead of $\boldsymbol{o}_i = \boldsymbol{S}_i \boldsymbol{q}_i$. Unlike the bidirectional approach, this requires only a single forward pass, and it outperforms all other non-causal variants within the stable range in Figure 2(b). The β_t decay again stabilises the final state $\boldsymbol{S}_{n_c}$, enables length generalisation, and improves performance compared to causal DeltaNet on the synthetic tasks as shown in Figure 10 in the appendix.

[1] We provide a minimal notebook for synthetic pretraining and sequence-length evaluation that reproduces this failure mode using a simple MLP prior, Gaussian features, and uniformly distributed class labels.

Table 3: Benchmark performance of the DeltaNet variants and the non-causal attention baselines (protocol of Section 3.1). FS denotes final-state readout, Decay denotes the time-dependent decay intervention, and Int-MT denotes the Interleaved Multi-Target configuration (Appendix B.2). All other models use Combined Single-Target. Bold and underline mark the best and second-best models per metric. Significance bars mark non-significant differences between models within one group.

	OpenML-CC18*		TabArena*	
	ROC-AUC ↑	ACC ↑	ROC-AUC ↑	ACC ↑
Softmax Non-Causal	**0.8900**	**0.8166**	**0.8172**	**0.8506**
Delta-FS + Decay	<u>0.8884</u>	<u>0.8145</u>	0.8139	<u>0.8468</u>
Delta + Decay (Int-MT)	0.8873	0.8120	<u>0.8141</u>	0.8459
Delta + Decay	0.8864	0.8091	0.8100	0.8438
Delta-FS	0.8874	0.8137	0.8022	0.8352
Delta	0.8860	0.8078	0.8074	0.8388
Linear Non-Causal	0.8833	0.8039	0.8004	0.8396

Table 3 evaluates the resulting models on real-world tasks. DeltaNet with final-state readout and write-rate decay is the strongest evaluated linear-time variant: it improves over standard DeltaNet on both benchmarks. It outperforms the causal Delta + Decay in the Interleaved Multi-Target setting on three of four metrics, while processing half as many tokens per dataset. Appendix D.8 further reports an extended comparison and shows that it aggregates information even better than the subsampled 3K and layer-local ridge regression baselines of Section 4.1. Finally, it substantially closes the significance gap to non-causal softmax attention on TabArena, which indicates that the improved length generalisation transfers to real-world datasets.

6 Discussion

In this study, we first disentangled the interaction of training design, information flow, and update rule. We show that Interleaved Multi-Target is the most suitable setting for causal models, but requires $2\times$ sequence length and is not valid for non-causal models. When compared in the more standard Combined Single-Target setting, we observe that the nature of the update rule is more important than non-causal information flow: at moderate context lengths, causal DeltaNet variants outperform our non-causal linear attention baseline and remain competitive with the externally trained TabFlex checkpoint. We then identify length generalisation as the main challenge for such order-dependent update rules even when they are extended to support non-causal information flow.

The hidden-state oracle rules out state capacity as the bottleneck for length generalisation. Instead, the recurrent update dynamics drift when the context is extended beyond the pretraining range. Under the test-time regression view, DeltaNet overfits its write rates β_t to the pretraining length rather than converging to a stable estimator. To mitigate this, we propose a time-dependent β_t decay. The resulting model is the best-performing linear-time model in our evaluation, closing most of the gap to softmax attention, especially in the Int-MT setting or when extended for non-causality via final-state readout.

Limitations and future work. We prioritise a controlled comparison over absolute performance. Larger pretraining budgets and further tuning may disproportionately improve performance. We use row-wise tabular ICL, but expect the results to extend to modern two-axis attention of TabPFN v2 (Hollmann et al., 2025) or TabICL's token preprocessing (Qu et al., 2025). Future work could explain the hidden-state drift, build on the write-rate decay, or develop update rules that are length-stable by design. Another interesting direction could aim to construct the (non-causal) oracle hidden state, for example via end-to-end test-time training (Tandon et al., 2025).

Conclusion. Through the lens of test-time regression, linear architectures compress the rows of tabular datasets into a fixed-size parametric state, either layer-locally or end-to-end. This paper shows across various considered settings that stable state construction is the key challenge to learn a generalising ICL algorithm. To that end, we adapt ordered fast-weight updates in both causal and non-causal models to stably aggregate information across exchangeable rows in tabular data. We hope that these insights pave the way for scalable tabular foundation models based on parametric estimation.

Reproducibility Statement

Code Availability. The code for all experiments is available at https://anonymous.4open.science/r/Linear-Tabular-ICL. Our implementation builds on the PFNs codebase of Müller et al. (2022), available at https://github.com/SamuelGabriel/PFNs. We provide the full codebase, including the pretraining and evaluation pipelines, experiment configs, and notebooks. Most of the linear-time models in the codebase were imported from the Flash Linear Attention Library (Yang & Zhang, 2024) and adapted to the tabular prediction setting with custom patches as described in Appendix F. We additionally provide a minimal notebook reproducing the length-generalisation failure mode of Section 4.3 using a simple MLP prior, Gaussian features, and uniformly distributed class labels.

Data Availability. Synthetic pretraining and length-generalisation datasets are generated on the fly from the TabPFN v1 prior. All real-world datasets used in our experiments, specifically for the OpenML-CC18 (Bischl et al., 2021) and TabArena (Erickson et al., 2026) benchmark evaluation, are freely available at https://www.openml.org (Vanschoren et al., 2014) and are downloaded automatically when running the evaluation in our codebase.

Pretraining and Hyperparameter Details. An overview of the pretraining setup is described in Section 3.1 with additional information in Appendix B.

Evaluation Reproducibility. The evaluation protocol is described in Section 3.1, with additional details about metrics and the significance testing described in Appendix C.

Ethics Statement

This work is methodological and studies scalable architectures for tabular in-context learning. Potential positive impacts include more efficient use of tabular foundation models, reduced inference costs on large context sizes, and longer model lifetimes by enabling pretrained models to use more data at prediction time without task-specific retraining. Such improvements could make tabular foundation models accessible in low-compute or latency-constrained settings, including edge devices.

Potential negative impacts arise from the broader use of more capable tabular prediction systems. Efficient long-context architectures could increase the deployment of tabular prediction systems on large structured data collections, where they may be misused without appropriate validation and oversight. This might reinforce biases through automated decision-making. Additionally, while tabular foundation models can be deployed to new tasks efficiently without task-specific training, they require significant initial compute resources for pretraining. Our focus on linear architectures aims to reduce this cost for long-context pretraining and improve inference efficiency.

AI Use Statement

We used generative artificial intelligence tools, including OpenAI's GPT, Anthropic's Claude, and Google's Gemini, in a supporting role during the writing and implementation phases of this project.

During the writing phase, we used the tools for language editing, grammar checking, and sentence- and sometimes paragraph-level reformulation based on our initial drafts and then continued to refine them manually. The literature search and review were carried out manually by us. We used the tools only for additional checks to find further relevant work and checked every suggested paper before using it in any capacity. During the implementation phase, we used the tools for coding assistance, such as implementing individual functions according to our specifications, wiring new parameters through the model code, helping write plotting and testing code, reviewing our hand-written code additions, and cleaning up implementations. Specifically, we designed and implemented the custom stateless prediction patch for single-target inference for linear attention first and then applied the patch to other models with tool assistance (Appendix F).

We want to be clear that we did not use AI to determine central parts such as the methodology, experimental design, interpretation of results, or conclusions. We also did not use the tools to define the research questions, make experimental decisions, interpret results, or draw scientific conclusions. We reviewed all AI-assisted text edits and suggestions, reviewed and tested all AI-assisted code suggestions, and take full responsibility for the final content of this work.

REFERENCES

Arshia Afzal, Elias Abad Rocamora, Leyla Naz Candogan, Pol Puigdemont, Francesco Tonin, Yongtao Wu, Mahsa Shoaran, and Volkan Cevher. Linear attention for efficient bidirectional sequence modeling. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=Ar62cqTduE.

Stefan K. Baur and Sohyeong Kim. Exploration of autoregressive models for in-context learning on tabular data. In *NeurIPS 2024 Third Table Representation Learning Workshop*, 2024. URL https://openreview.net/forum?id=4dOJ0PRY7R.

Ali Behrouz, Peilin Zhong, and Vahab Mirrokni. Titans: Learning to memorize at test time. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=8GjSf9Rh7Z.

Alessio Benavoli, Giorgio Corani, and Francesca Mangili. Should we really use post-hoc tests based on mean-ranks? *The Journal of Machine Learning Research*, 17(1):152–161, 2016.

Bernd Bischl, Giuseppe Casalicchio, Matthias Feurer, Pieter Gijsbers, Frank Hutter, Michel Lang, Rafael Gomes Mantovani, Jan N. van Rijn, and Joaquin Vanschoren. OpenML benchmarking suites. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*, 2021. URL https://openreview.net/forum?id=OCrD8ycKjG.

Ricardo Buitrago and Albert Gu. Understanding and improving length generalization in recurrent models. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=2OEb20dy7B.

Tri Dao and Albert Gu. Transformers are ssms: generalized models and efficient algorithms through structured state space duality. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org, 2024.

Nan Ding, Tomer Levinboim, Jialin Wu, Sebastian Goodman, and Radu Soricut. CausalLM is not optimal for in-context learning. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=guRNebwZBb.

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*, 2026. URL https://openreview.net/forum?id=jZqCqpCLdU.

Salvador Garcia and Francisco Herrera. An extension on" statistical comparisons of classifiers over multiple data sets" for all pairwise comparisons. *Journal of machine learning research*, 9(12), 2008.

Léo Grinsztajn, Edouard Oyallon, and Gaël Varoquaux. Why do tree-based models still outperform deep learning on typical tabular data? *Advances in neural information processing systems*, 35: 507–520, 2022.

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.

Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=tEYskw1VY2.

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*, 01 2025. doi: 10.1038/s41586-024-08328-6. URL https://www.nature.com/articles/s41586-024-08328-6.

Sture Holm. A simple sequentially rejective multiple test procedure. *Scandinavian journal of statistics*, pp. 65–70, 1979.

Sukjun Hwang, Aakash Lahoti, Ratish Puduppully, Tri Dao, and Albert Gu. Hydra: Bidirectional state space models through generalized matrix mixers. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=preo49P1VY.

Hassan Ismail Fawaz, Germain Forestier, Jonathan Weber, Lhassane Idoumghar, and Pierre-Alain Muller. Deep learning for time series classification: a review. *Data mining and knowledge discovery*, 33(4):917–963, 2019.

Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers are rnns: Fast autoregressive transformers with linear attention. In *International conference on machine learning*, pp. 5156–5165. PMLR, 2020.

Felix Koch, Marcel Wever, Fabian Raisch, and Benjamin Tischler. State-space models for tabular prior-data fitted networks. In *1st ICML Workshop on Foundation Models for Structured Data*, 2025.

Barak Lenz, Opher Lieber, Alan Arazi, et al. Jamba: Hybrid transformer-mamba language models. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=JFPaD7lpBD.

Junwei Ma, Valentin Thomas, Rasa Hosseinzadeh, Alex Labach, Jesse C. Cresswell, Keyvan Golestan, Guangwei Yu, Anthony L. Caterini, and Maksims Volkovs. TabDPT: Scaling tabular foundation models on real data. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=pIZxEOZCId.

Vladyslav Moroshan, Julien Siems, Arber Zela, Timur Carstensen, and Frank Hutter. Tempopfn: Synthetic pre-training of linear rnns for zero-shot time series forecasting. *arXiv preprint arXiv:2510.25502*, 2025.

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.

Liudmila Prokhorenkova, Gleb Gusev, Aleksandr Vorobev, Anna Veronika Dorogush, and Andrey Gulin. Catboost: unbiased boosting with categorical features. *Advances in neural information processing systems*, 31, 2018.

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. In *International Conference on Machine Learning*, 2026.

Herbert Robbins and Sutton Monro. A stochastic approximation method. *The annals of mathematical statistics*, pp. 400–407, 1951.

Imanol Schlag, Kazuki Irie, and Jürgen Schmidhuber. Linear transformers are secretly fast weight programmers. In *Proceedings of the 38th International Conference on Machine Learning (ICML 2021)*, Proceedings of Machine Learning Research, pp. 9355–9366. PMLR, 2021. URL https://proceedings.mlr.press/v139/schlag21a.html.

Zhenghang Song, Tang Qian, Lu Chen, Yushuai Li, Zhengke Hu, Bingbing Fang, Yumeng Song, Junbo Zhao, Sheng Zhang, and Tianyi Li. Feat: A linear-complexity foundation model for extremely large structured data. *arXiv preprint arXiv:2603.16513*, 2026.

Yu Sun, Xinhao Li, Karan Dalal, Jiarui Xu, Arjun Vikram, Genghan Zhang, Yann Dubois, Xinlei Chen, Xiaolong Wang, Sanmi Koyejo, Tatsunori Hashimoto, and Carlos Guestrin. Learning to (learn at test time): RNNs with expressive hidden states. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=wXfuOj9C7L.

Arnuv Tandon, Karan Dalal, Xinhao Li, Daniel Koceja, Marcel Rød, Sam Buchanan, Xiaolong Wang, Jure Leskovec, Sanmi Koyejo, Tatsunori Hashimoto, et al. End-to-end test-time training for long context. *arXiv preprint arXiv:2512.23675*, 2025.

Asher Trockman, Hrayr Harutyunyan, J Zico Kolter, Sanjiv Kumar, and Srinadh Bhojanapalli. Mimetic initialization helps state space models learn to recall. In *Workshop on Neural Network Weights as a New Data Modality*, 2025. URL https://openreview.net/forum?id=nerNr9fjfD.

Neehal Tumma, Noel Loo, and Daniela Rus. Preconditioned deltanet: Curvature-aware sequence modeling for linear recurrences. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=UC6YiTOeKb.

Joaquin Vanschoren, Jan N. van Rijn, Bernd Bischl, and Luis Torgo. Openml: networked science in machine learning. *ACM SIGKDD Explorations Newsletter*, 15(2):49–60, June 2014. ISSN 1931-0153. doi: 10.1145/2641190.2641198. URL http://dx.doi.org/10.1145/2641190.2641198.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Johannes von Oswald, Nino Scherrer, Seijin Kobayashi, Luca Versari, Songlin Yang, Maximilian Schlegel, Kaitlin Maile, Yanick Schimpf, Oliver Sieberling, Alexander Meulemans, Guillaume Lajoie, Rif A. Saurous, Charlotte Frenkel, Razvan Pascanu, Blaise Aguera y Arcas, and Joao Sacramento. Mesanet: Sequence modeling by locally optimal test-time training. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=xa3OnTb6c3.

Ke Alexander Wang, Jiaxin Shi, and Emily B. Fox. Test-time regression: a unifying framework for designing sequence models with associative memory, 2025. URL https://arxiv.org/abs/2501.12352.

Frank Wilcoxon. Individual comparisons by ranking methods. *Biometrics bulletin*, 1(6):80–83, 1945.

Songlin Yang and Yu Zhang. Fla: A triton-based library for hardware-efficient implementations of linear attention mechanism, January 2024. URL https://github.com/fla-org/flash-linear-attention.

Songlin Yang, Bailin Wang, Yikang Shen, Rameswar Panda, and Yoon Kim. Gated linear attention transformers with hardware-efficient training. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org, 2024a.

Songlin Yang, Bailin Wang, Yu Zhang, Yikang Shen, and Yoon Kim. Parallelizing linear transformers with the delta rule over sequence length. *Advances in neural information processing systems*, 37: 115491–115522, 2024b.

Songlin Yang, Jan Kautz, and Ali Hatamizadeh. Gated delta networks: Improving mamba2 with delta rule. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=r8H7xhYPwz.

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.

Appendices

A Computational Resources **15**

B Training Details **15**

B.1 Linear-Time Architecture Details 15

B.2 Pretraining and Hyperparameter Details 15

B.3 External Baselines 16

B.4 Hidden-State Oracle Details 17

B.5 Layer-Local Ridge State Details 17

C Evaluation Details **17**

C.1 Metrics Used in Evaluation 17

C.2 Significance Testing 18

C.2.1 Significance Testing for Training Strategies 18

C.2.2 Significance Testing for Sequence Mixers 18

D Additional Results **19**

D.1 Real-World Length Generalisation on TabArena 19

D.2 Hidden-State Oracle on Real Data 19

D.3 Layer-Local Ridge State Comparison 19

D.4 Scaling Pretraining Sequence Length 20

D.5 Generalisation and Mitigation Strategies for GLA 20

D.6 Generalisation Failure Analysis for Linear Attention 22

D.7 DeltaNet Update Dynamics Analysis 24

D.8 Extended Real-World Model Comparison 27

D.9 Two-Axis Row-Feature Processing 28

E Memory and Compute Scaling Analysis **29**

F Adding Stateless Prediction to Flash-Linear-Attention Models **30**

A Computational Resources

Experiments were performed primarily on a cluster node with 8 NVIDIA RTX 2080 Ti GPUs, each with 11 GB of VRAM, and an Intel Xeon Gold 5120 processor. For pretraining with longer context sizes, where GPU memory was a constraint, we used a second cluster node with 8 NVIDIA RTX A5000 GPUs, each with 24 GB of VRAM, and an AMD EPYC 7742 64-Core Processor.

Pretraining each model took approximately one to two days on a single GPU, depending on the underlying architecture. Length-generalisation evaluations additionally required several hours per model. In total for the experiments in the paper, we trained the five linear/recurrent model families across up to four different training setups, as well as causal and non-causal softmax-attention Transformer variants, the mitigation strategies for DeltaNet and GLA, the stabilised DeltaNet variants with final-state readout and write-rate decay, and the layer-local ridge state model. Across all final experiments, this corresponds to approximately 100-125 GPU-days for pretraining and evaluation.

Additional compute was used for preliminary experiments, implementation development, and selecting fair parameter configurations for each architecture.

B Training Details

B.1 Linear-Time Architecture Details

Table 4 summarises the update and readout rules for the evaluated model families.

Table 4: Unified recurrent-memory view of the linear-time sequence mixers considered in this work. The table is adapted from Yang et al. (2024b). $\boldsymbol{S}_t$ denotes the recurrent memory state, and $\boldsymbol{o}_t$ is the unnormalised memory readout. We omit normalisation terms, output projections, and implementation-specific details for clarity.

Model	Recurrence	Readout
Linear Attention	$\boldsymbol{S}_t = \boldsymbol{S}_{t-1} + \boldsymbol{v}_t\boldsymbol{k}_t^\top$	$\boldsymbol{o}_t = \boldsymbol{S}_t\boldsymbol{q}_t$
+ Kernel	$\boldsymbol{S}_t = \boldsymbol{S}_{t-1} + \boldsymbol{v}_t\phi(\boldsymbol{k}_t)^\top$	$\boldsymbol{o}_t = \boldsymbol{S}_t\phi(\boldsymbol{q}_t)$
Linear Attn. Gated	$\boldsymbol{S}_t = \boldsymbol{S}_{t-1}\mathrm{Diag}(\alpha_t) + \boldsymbol{v}_t\boldsymbol{k}_t^\top$	$\boldsymbol{o}_t = \boldsymbol{S}_t\boldsymbol{q}_t$
Delta Rule	$\boldsymbol{S}_t = \boldsymbol{S}_{t-1}\left(\mathbf{I} - \beta_t\boldsymbol{k}_t\boldsymbol{k}_t^\top\right) + \beta_t\boldsymbol{v}_t\boldsymbol{k}_t^\top$	$\boldsymbol{o}_t = \boldsymbol{S}_t\boldsymbol{q}_t$
Delta Rule Gated	$\boldsymbol{S}_t = \boldsymbol{S}_{t-1}\left(\alpha_t(\mathbf{I} - \beta_t\boldsymbol{k}_t\boldsymbol{k}_t^\top)\right) + \beta_t\boldsymbol{v}_t\boldsymbol{k}_t^\top$	$\boldsymbol{o}_t = \boldsymbol{S}_t\boldsymbol{q}_t$
Mamba-2	$\boldsymbol{S}_t = \gamma_t\boldsymbol{S}_{t-1} + \boldsymbol{v}_t\boldsymbol{k}_t^\top$	$\boldsymbol{o}_t = \boldsymbol{S}_t\boldsymbol{q}_t$

The size-matched models in Tables 2 and 3 use embedding dimension $d = 320$, 4 heads where applicable, and the feed-forward/intermediate dimension is set to $640 = 2d$. We set the number of layers separately for each architecture to match all models to approximately 12.5M parameters, resulting in 12 layers for most of the backbones. Table 5 shows the exact parameter counts for the models in these tables.

B.2 Pretraining and Hyperparameter Details

Overall, we use a near-identical setup to TabPFN v1, but with different hyperparameters. All models are pretrained on synthetic tasks sampled on the fly from the TabPFN v1 prior. For each minibatch, we sample fresh synthetic datasets, including the number of features, the context/query split, and the number of label classes. As the prior samples new datasets for each minibatch, we used the pretraining loss and a validation loss on additional synthetic datasets for general and backbone-specific hyperparameter selection. The additional synthetic validation datasets are needed for model development, as different training strategies, as described in Section 3.2, provide different access to information, which can change pretraining loss.

Unless stated otherwise, all models use the shared pretraining config described in Table 6. In total, each model sees 4000 minibatches/epoch $\times$ 200 epochs $\times$ 8 datasets/minibatch $=$ 6.4M synthetic datasets. The loss is always computed on query rows.

Table 5: Parameter counts for the models in the controlled comparisons of Tables 2 and 3. Counts include the encoders, backbone, and prediction head. Shared weights are counted once. Delta-FS, Delta + Decay, and Delta-FS + Decay have the same parameter count as Delta.

Model	Parameters
Softmax Non-Causal	12,507,210
Delta	12,489,162
Delta Gated	12,504,426
Linear Non-Causal	12,473,482
Linear Gated	12,567,882
Linear Causal	12,473,802
Mamba-2	12,492,582
Layer-Local Ridge	12,473,802
Delta + Decay (Int-MT)	12,535,242

Table 6: Shared pretraining hyperparameters for the size-matched real-world comparisons.

Hyperparameter	Value
Synthetic prior	TabPFN v1 prior
Rows per dataset	1000
Context/query split	uniformly sampled, min. 64 context rows
Input features	uniformly sampled from 2-20
Classes	up to 10
Training budget	200 epochs, 4000 minibatches/epoch
Batching	8 datasets/minibatch, 2-step grad. accumulation
Effective batch size	16 synthetic datasets
Embedding/hidden size	320
Attention heads	4
FFN/intermediate size	640
Optimiser	AdamW, lr $3 \cdot 10^{-5}$, weight decay 0.01
Schedule	cosine decay, 10 warmup epochs
Loss	cross-entropy on query rows

Short-convolution modules are disabled (`use_short_conv=False`), except for DeltaNet Interleaved Multi-Target (Int-MT), which uses convolution width 4. In our exploratory evaluations, short convolutions benefited DeltaNet in the interleaved setting, while reducing performance or providing no material improvement for the other evaluated configurations.

B.3 External Baselines

To provide reference performance on our real-world benchmarks, we include TabPFN v2.5 (Grinsztajn et al., 2025; Hollmann et al., 2025), TabICLv2 (Qu et al., 2025; 2026), TabFlex (Zeng et al., 2025), and CatBoost (Prokhorenkova et al., 2018).

TabPFN v2.5 and TabICLv2 represent the state of the art in tabular foundation models according to TabArena (Erickson et al., 2026). TabFlex is the most closely aligned baseline, as it uses a non-causal linear attention backbone and is also trained on the TabPFN v1 prior (Hollmann et al., 2023). Importantly, these baselines are not controlled for training budget or model size: we evaluate the models released by their respective authors. The foundation model baselines have comparable or larger parameter counts than our 12.5M-parameter models, and were pretrained on substantially more data, including larger datasets. Our models were pretrained on 6.4B rows of data, while we estimate the pretraining data exposure for these baseline models at 40B-100B+ rows. These differences, together with the more advanced priors used by TabICLv2 and TabPFN v2.5, help explain the performance gap in Table 2 between our controlled models and the external baselines.

CatBoost is included as a strong gradient-boosted decision-tree baseline. As shown in the TabArena evaluation by Erickson et al. (2026), CatBoost performs very well out of the box without hyperparameter tuning, supporting its use as a default configuration baseline.

B.4 Hidden-State Oracle Details

For each evaluation dataset, the hidden-state oracle proceeds as follows. We first run a selected linear-time model (in this case, the pretrained Comb-ST DeltaNet) on the context samples to obtain an initial hidden state $\boldsymbol{S}$ for each layer. We then freeze all model parameters and optimise only these hidden states using multi-epoch mini-batch gradient descent on the end-to-end test-time regression objective over the context set

$$\mathcal{L}(\boldsymbol{S}) = \mathbb{E}_{(x_i,y_i)\sim\mathcal{C}}\mathrm{CE}(f_\theta(\boldsymbol{S}, x_i), y_i). \tag{7}$$

Here, $f_\theta(\boldsymbol{S}, x_i)$ denotes the prediction of the frozen model for a context row x_i that is processed as a query. Each layer reads from its own state in $\boldsymbol{S}$, but since the loss is backpropagated through the whole network, the states of all layers are optimised jointly. This procedure keeps the model architecture, projections, and readout fixed, modifying only the per-layer fast-weight states $\boldsymbol{S}$.

For hyperparameter tuning and early stopping, we split the context set into training and validation subsets, holding out 10% of the context rows. Early stopping is necessary because the optimised hidden states can overfit the context set, even at a sequence length of 128k. This further supports the conclusion that fixed-size hidden-state capacity is not the primary bottleneck.

B.5 Layer-Local Ridge State Details

The layer-local ridge state model replaces the recurrent state update with the closed-form solution of a length-scaled ridge objective. This deviates from the hidden-state oracle by not optimising the hidden states jointly across all layers on the end-to-end objective but by computing the state of each layer independently from the layer-local objective of Equation 3 and using its closed-form solution. Additionally, unlike the hidden-state oracle, which uses frozen model parameters, we pretrain the layer-local ridge state model from scratch and match the pretraining budget (results in Appendix D.3). Let $\boldsymbol{K} \in \mathbb{R}^{n\times d_{qk}}$ and $\boldsymbol{V} \in \mathbb{R}^{n\times d_v}$ collect the keys and values as rows. The Moore-Penrose solution $\boldsymbol{S}_n^\star = \boldsymbol{V}^\top(\boldsymbol{K}^\dagger)^\top$ to the layer-local least-squares problem corresponds to the least-squares memory of Wang et al. (2025). In our experiments and in line with Tumma et al. (2026), learned key matrices can become ill-conditioned, making the ridgeless solution unstable and increasing both pretraining loss and long-context degradation. We therefore use a length-scaled ridge objective with closed-form solution

$$\boldsymbol{S}_n^{\mathrm{ridge}} = \operatorname*{argmin}_{\boldsymbol{S}\in\mathbb{R}^{d_v\times d_{qk}}} \frac{1}{n}\sum_{t=1}^{n} \|\boldsymbol{v}_t - \boldsymbol{S}\boldsymbol{k}_t\|_2^2 + \lambda\,\|\boldsymbol{S}\|_F^2 = \boldsymbol{V}^\top\boldsymbol{K}\left(\boldsymbol{K}^\top\boldsymbol{K} + n\lambda\boldsymbol{I}\right)^{-1}.$$

This state accounts for the covariance structure of key representations, remains a permutation-invariant function of the context, and scales the regularisation strength with the context length n.

C Evaluation Details

C.1 Metrics Used in Evaluation

In the main evaluation, we report raw ROC-AUC and accuracy, averaged over splits and datasets. Details on significance testing are provided in Appendix C.2. For the length generalisation experiments, we sample 500 synthetic datasets of 128k rows from the pretraining prior and truncate them to context prefixes of increasing length n_c, holding the query set fixed at $n_q = 100$ samples. We report normalised ROC-AUC and accuracy. Since the synthetic datasets vary substantially in difficulty, normalisation makes confidence intervals more comparable when aggregating across datasets.

The shaded regions in the plots show 95% confidence intervals. Normalisation constants are computed per dataset using a fixed reference set consisting of the controlled model families from Table 2, including the four-head softmax reference, and the hidden-state oracle, rather than only the models shown in each figure. For normalisation, all evaluated context lengths of the same synthetic dataset share the same normalisation constants. This makes scores comparable across the different length-generalisation figures.

Specifically, for each dataset d, metric m, and model/context-length configuration $r \in \mathcal{R}$, we normalise raw scores using the minimum over reference configurations at context lengths up to 1k

($\mathcal{R}_{\leq 1\text{k}}$) and the maximum over all reference configurations:

$$\hat{s}_{d,r}^{(m)} = \frac{s_{d,r}^{(m)} - \min_{r' \in \mathcal{R}_{\leq 1\text{k}}} s_{d,r'}^{(m)}}{\max_{r' \in \mathcal{R}} s_{d,r'}^{(m)} - \min_{r' \in \mathcal{R}_{\leq 1\text{k}}} s_{d,r'}^{(m)}}. \tag{8}$$

C.2 Significance Testing

Our significance testing follows the pairwise comparison protocol used by Ismail Fawaz et al. (2019), which, in turn, is based on the recommendation of Benavoli et al. (2016) to use tests that perform pairwise comparisons, such as the Wilcoxon signed-rank test. Accordingly, for each benchmark and metric, we perform paired two-sided Wilcoxon signed-rank tests (Wilcoxon, 1945) and additionally apply Holm's alpha correction at $\alpha = 5\%$ (Holm, 1979; Garcia & Herrera, 2008).

C.2.1 Significance Testing for Training Strategies

Let $x_{f,d,s,r}^{(m)}$ denote the value of metric m for model family f, dataset d, training setup s, and cross-validation split r.

For this analysis, we include only model families for which all four training setups are available. In our experiments, these are Gated Linear Attention, DeltaNet, Gated DeltaNet, and Mamba-2. We exclude standard Linear Attention because the interleaved-label setup would require custom positional encodings to obtain comparable performance.

1. Average over the cross-validation splits:

$$\bar{x}_{f,d,s}^{(m)} = \frac{1}{R} \sum_{r=1}^{R} x_{f,d,s,r}^{(m)}$$

2. For each dataset and setup, we then average over the model families:

$$\tilde{x}_{d,s}^{(m)} = \frac{1}{F} \sum_{f=1}^{F} \bar{x}_{f,d,s}^{(m)}$$

3. For each benchmark and metric, we use Wilcoxon signed-rank tests with Holm correction to compare two setups s and s' pairwise.

C.2.2 Significance Testing for Sequence Mixers

For the recurrence formulation comparison, we used the same paired Wilcoxon/Holm procedure described above. The training setup is Combined Single-Target, except for the configurations labelled Int-MT. For each model, dataset, benchmark, and metric, we first average over cross-validation splits. We then compare all pairs of included models using the resulting dataset-level paired samples. Holm correction is applied separately for each benchmark and metric: across the 21 pairs of Table 2, the 21 pairs of Table 3, and all 55 pairs in Tables 8 and 9. Because each table corrects within its own set of models, the same model pair can receive different significance decisions across tables. Shared bars indicate non-significant differences, not equivalence.

D Additional Results

D.1 Real-World Length Generalisation on TabArena

Sequence length generalisation results on the TabArena benchmark across selected models are shown in Figure 4. This visualises the effect of causal degradation in a long-context real-world task.

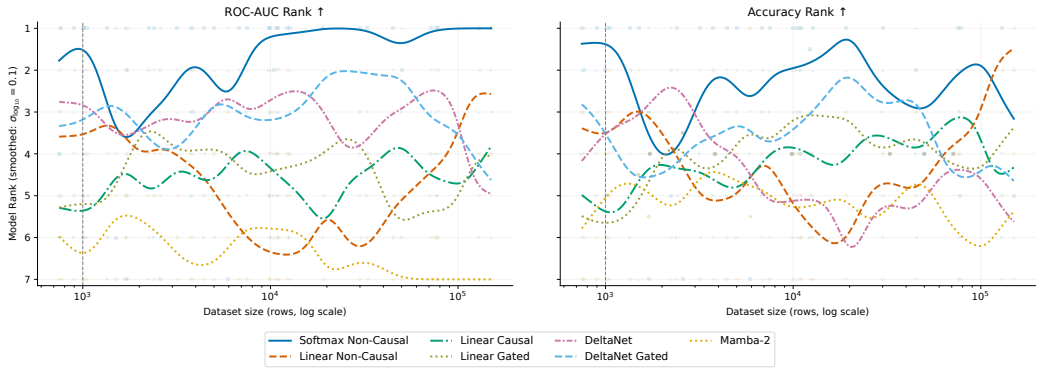


Figure 4: Model ranks as a function of dataset size in TabArena. The ranks for each model are smoothed in $\log 10$-space with a Gaussian kernel of $\sigma_{\log 10} = 0.1$. At moderate sizes, DeltaNet variants are competitive with softmax attention. With increasing dataset size n_c, non-causal softmax and linear attention outperform the causal models.

D.2 Hidden-State Oracle on Real Data

The hidden-state oracle of Section 4 is evaluated on datasets sampled from our synthetic prior. We here repeat the analysis on the five largest TabArena classification datasets, using the same procedure as in Appendix B.4 and stratified 5-fold cross-validation.

Table 7 reports mean ROC-AUC across folds. The oracle outperforms causal DeltaNet, non-causal linear attention, and non-causal softmax attention on every dataset and on mean performance. The capacity conclusion drawn from the synthetic prior therefore also holds on real-world data: a fixed-size state reaching strong performance exists at these context lengths.

Table 7: Hidden-state oracle comparison on the five largest TabArena classification datasets, reporting ROC-AUC averaged over stratified 5-fold cross-validation splits. Bold marks the best value per dataset and best mean.

Dataset	Rows	DeltaNet	Hidden-State Oracle	Softmax Non-Causal	Linear Non-Causal
GiveMeSomeCredit	150,000	0.8446	**0.8615**	0.8559	0.8546
Airline satisfaction	129,880	0.9314	**0.9931**	0.9765	0.9486
SDSS17	78,053	0.9851	**0.9938**	0.9915	0.9889
APSFailure	76,000	0.9828	**0.9862**	0.9845	0.9816
Diabetes130US	71,518	0.5587	**0.6165**	0.5887	0.5418
Mean		0.8605	**0.8902**	0.8794	0.8631

D.3 Layer-Local Ridge State Comparison

Figure 5 shows the synthetic length-generalisation comparison between the layer-local ridge state model (Appendix B.5), the DeltaNet variants, and non-causal linear attention. Pretrained under the same protocol as the recurrent models, the ridge state is length-stable and substantially improves upon non-causal additive linear attention, yet the final-state readout DeltaNet with write-rate decay closely

matches and sometimes exceeds it over the evaluated range. If DeltaNet's success were exclusively due to approximating the closed-form solution of the layer-local objective, replacing its update with the ridge solution should be expected to improve performance. This suggests that the layer-local associative-memory objective is not a complete proxy for downstream tabular ICL performance.

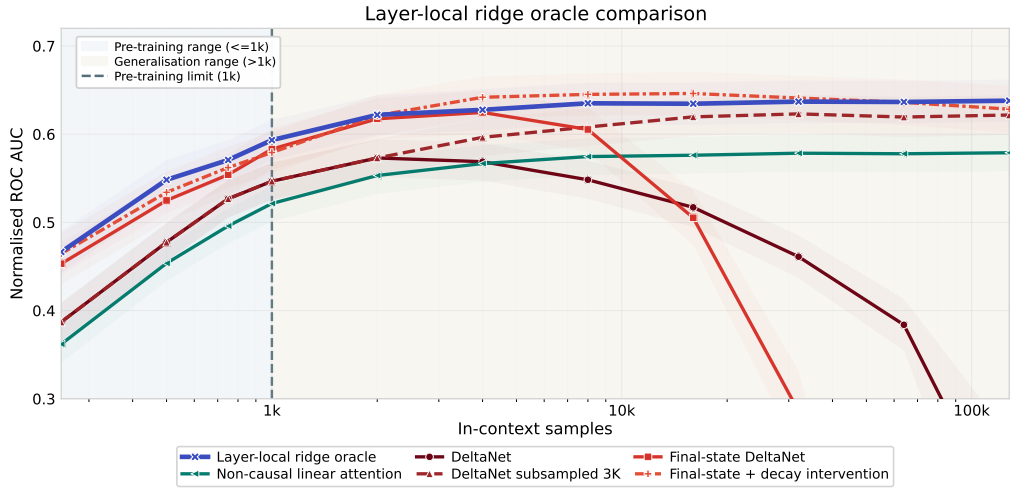


Figure 5: Comparison of layer-local ridge model with DeltaNet variants and non-causal linear attention over increasing sequence lengths. Curves show globally normalised ROC-AUC with 95% confidence intervals. The layer-local ridge model is pretrained from scratch using the same protocol, but replaces the recurrent state update with a closed-form ridge regression solve.

The full benchmark comparison, including the layer-local ridge state model, is shown in Tables 8 and 9. The ridge state performs competitively but is outperformed by the stabilised DeltaNet, with significant differences on TabArena.

D.4 Scaling Pretraining Sequence Length

Figure 6 shows length generalisation when pretraining DeltaNet with longer context lengths. For all models except the DeltaNet reference, we sample the number of rows per pretraining dataset from a log-uniform distribution between 200 and a maximum of 8k, 16k, or 64k rows.

In the figure, we see that with longer-context pretraining, we improve performance at longer evaluation contexts as expected. However, this does not address the underlying issue; it only shifts the degradation to longer sequence lengths. Importantly, these runs are not compute-matched. Although all models are pretrained on the same number of datasets, long-context models see substantially more rows during pretraining. This also explains why their short-context performance does not deteriorate, unlike in the compute-matched setting in Figure 2.

D.5 Generalisation and Mitigation Strategies for GLA

Figure 7 visualises the same length-generalisation analysis as Section 4.2 for gated linear attention (GLA). For GLA, the mitigation strategies are more effective than for DeltaNet, although they do not fully remove the degradation.

State passing (Buitrago & Gu, 2025), which exposes the model to longer effective histories by reusing the cached recurrent states from prior batches, improves long-context behaviour. However, it still introduces degradation and reduces the performance within the pretraining range. *State weaving* (Moroshan et al., 2025), which introduces non-causal information flow by initialising the hidden state of the subsequent layer with the final hidden state of the previous layer, shows minor improvements at pretraining context sizes, but does not reduce length generalisation degradation. *Mimetic initialisation* (Trockman et al., 2025), which initialises gates to be open by default and makes GLA behave closer to ungated additive linear attention, also reduces degradation. *Bidirectionality* and *final-state readout*,

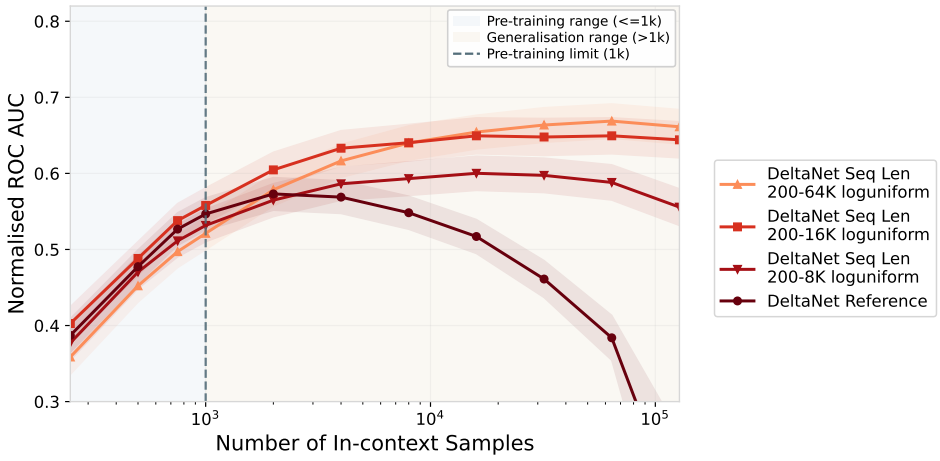


Figure 6: Length generalisation of DeltaNet models pretrained with longer context lengths. The reference model is pretrained with at most 1000 rows, while the other models sample the row number from a log-uniform distribution between 200 and the maximum shown in the legend. Longer-context pretraining improves long-context performance but mainly shifts degradation to larger sequences. The comparison is not compute-matched, since all models see the same number of datasets but a different total number of rows. A compute-matched version is in Section 4 of the main paper.

which both remove the readout-level causal bottleneck, are the most effective overall. Final-state readout is strongest up to around $16\times$ the pretraining length, after which bidirectionality overtakes it and remains the strongest at very long contexts.

We hypothesise that these improvements are mostly explained by moving GLA closer to causal linear attention through mimetic initialisation, while bidirectionality and final-state readout move its information flow towards a non-causal formulation. Both causal and non-causal linear attention exhibit less degradation than the underlying GLA, consistent with the more stable long-context behaviour of these variants.

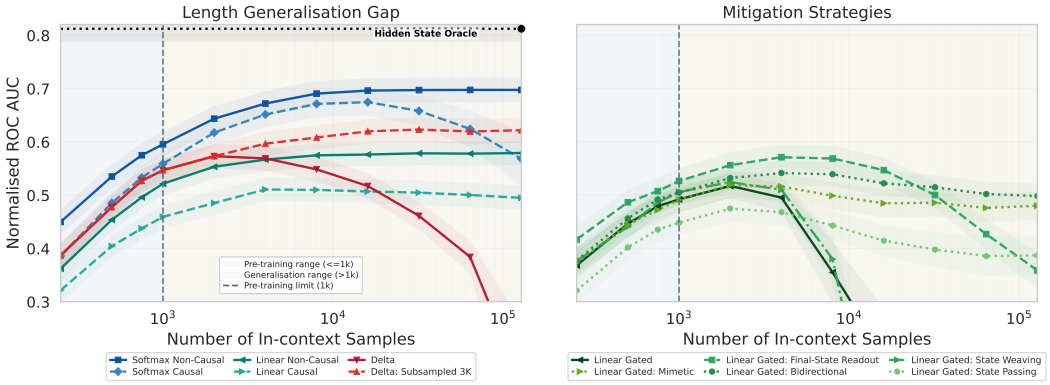


Figure 7: Length generalisation for GLA beyond the 1k pretraining length. We report normalised ROC-AUC averaged over 500 synthetic datasets with 95% confidence intervals. **(a) Length Generalisation Gap:** Compared to other models, GLA exhibits strong degradation starting at $4\times$ the pretraining context length. **(b) Mitigation Attempts:** State passing and mimetic initialisation mitigate but do not eliminate long-sequence degradation, while bidirectionality is the strongest mitigation for GLA at very long contexts. Final-state readout is strongest up to around $16\times$ the pretraining length but degrades substantially beyond it. State weaving slightly improves performance in the pretraining context range, but shows no significant improvement in addressing degradation.

D.6 Generalisation Failure Analysis for Linear Attention

When comparing causal linear attention to its non-causal counterpart, we observe that the Frobenius norm of the recurrent key-value state drifts relative to other layers with increasing context length in the causal setup, while this effect is much weaker in the non-causal setup, as shown in Figure 8. This behaviour is consistent both when using the K-sum denominator normalisation $\frac{1}{Z_i}$ from Equation 2 and when omitting it. We controlled for this magnitude drift by explicitly renormalising the recurrent key-value state. For a linear-attention layer ℓ and head h, let $S_{\ell,h,t} \in \mathbb{R}^{d_v \times d_{qk}}$ denote the recurrent key-value state after processing context position t. At readout time, we use the Frobenius-normalised state:

$$\widetilde{S}_{\ell,h,t} = S_{\ell,h,t} \cdot \frac{\alpha_{\ell,h}\sqrt{d_v}}{\max(\|S_{\ell,h,t}\|_F, \varepsilon)}. \tag{9}$$

Here $\|S_{\ell,h,t}\|_F = \sqrt{\sum_{i,j} (S_{\ell,h,t})_{ij}^2}$ denotes the Frobenius norm, ε is a small numerical constant, and $\alpha_{\ell,h}$ is a learned per-layer, per-head scale factor initialised to one and parametrised in log-space.

The normalisation preserves the direction of the recurrent state matrix while controlling its magnitude. The readout then uses the normalised state rather than the unnormalised state. We find that this constraint has only a very limited effect on both causal and non-causal linear attention performance (top panel of Figure 8).

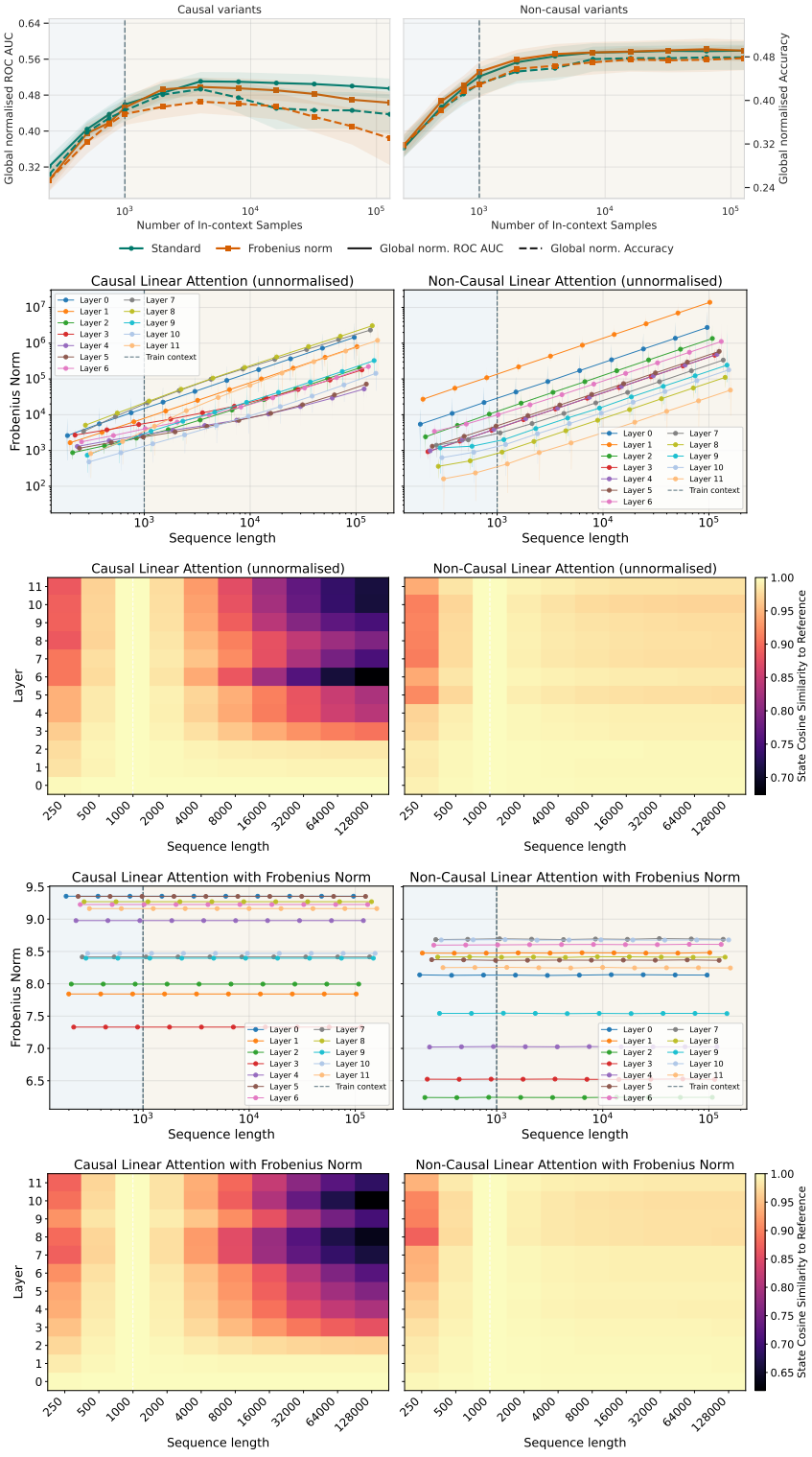


Figure 8: Length generalisation performance of standard and Frobenius-normalised causal and non-causal linear attention variants (top panel), and Frobenius norm evolution for the first head, together with directional drift averaged over heads and repetitions, across sequence lengths, for the unnormalised (second and third panels) and Frobenius-normalised (bottom two panels) models. Constraining the state magnitude does not remove the degradation of the causal model.

For the retrained models with the Frobenius-normalised recurrent states, the readout uses a state whose magnitude is constrained, so remaining changes in the effective state are primarily directional. We therefore measure hidden-state drift by comparing the orientation of the recurrent key-value state at each sequence length to the corresponding state at the 1000-row pretraining context length. For each model, repetition, layer, and head, we flatten the effective recurrent state matrix and compute its cosine similarity to the same layer and head at sequence length 1000. The heatmaps in Figure 8 report the average of cosine similarity over heads and repetitions for each layer and sequence. For causal linear attention the cosine similarity decreases steadily with context length, most strongly in deeper layers, whereas for the non-causal model it stays close to one. This indicates that the Frobenius-normalised causal state drifts away from its short-context configuration as the context grows, whereas the non-causal state remains close to its short-context configuration.

D.7 DeltaNet Update Dynamics Analysis

We here provide the full analysis of the DeltaNet update dynamics summarised in Section 4.4.

Linear attention vs DeltaNet state construction. The additive linear-attention state $\boldsymbol{S}_n = \sum_{t=1}^{n} \boldsymbol{v}_t \boldsymbol{k}_t^\top = \boldsymbol{V}^\top \boldsymbol{K}$ is a commutative set aggregate for fixed layer-local keys and values and does not depend on the arbitrary order of the context rows. While the raw sum grows with n, its length-normalised version converges under i.i.d. sampling to the expected key-value outer product $\mathbb{E}[\boldsymbol{v}\boldsymbol{k}^\top]$, so with appropriate state or readout normalisation it behaves like a stable set statistic. It can also be interpreted as a single gradient step from a zero-initialised state on the layer-local memory objective of Equation 3 (Wang et al., 2025). The delta-rule state, in contrast, is not an additive set aggregate but an ordered residual update. In the test-time regression view (Wang et al., 2025), DeltaNet performs first-order updates on the per-sample squared loss:

$$\boldsymbol{S}_t = \boldsymbol{S}_{t-1}\left(\mathbf{I} - \beta_t \boldsymbol{k}_t \boldsymbol{k}_t^\top\right) + \beta_t \boldsymbol{v}_t \boldsymbol{k}_t^\top = \boldsymbol{S}_{t-1} - \beta_t \nabla_{\boldsymbol{S}} \ell_t(\boldsymbol{S}_{t-1}), \qquad \ell_t(\boldsymbol{S}) = \frac{1}{2} \left\| \boldsymbol{S}\boldsymbol{k}_t - \boldsymbol{v}_t \right\|_2^2 .$$

Therefore, final-state readout does not make DeltaNet fully order-independent. For additive linear attention, replacing prefix states with the final state recovers the non-causal set-aggregate structure, whereas for DeltaNet the state $\boldsymbol{S}_n$ itself remains the result of an ordered online optimisation trajectory.

Learned write rates and effective memory. One central learned component of the delta-rule update is the input-dependent write rate $\beta_t = \sigma(\boldsymbol{w}_\beta^\top \boldsymbol{h}_t)$. For a fixed unit-norm key direction, the update changes the memory readout from its previous value towards the current value:

$$\boldsymbol{S}_t \boldsymbol{k}_t = (1 - \beta_t) \boldsymbol{S}_{t-1} \boldsymbol{k}_t + \beta_t \boldsymbol{v}_t .$$

Thus, a larger β_t gives more influence to the current sample and overwrites the previous memory more strongly along the key direction, while a smaller β_t preserves more of the previous state. The effective memory horizon of DeltaNet is therefore controlled by the learned sequence of write rates, which can overfit to the pretraining range and be overly aggressive when the same recurrence is unrolled over much larger sequence lengths.

Figure 9 compares the effective write rates of standard DeltaNet, a variant pretrained on sequence lengths of up to 64,000 samples instead of 1,000, and the write-rate decay intervention, extending the layer view of Figure 3 in the main paper. The long-context pretrained models are not compute-matched and are used only for analysis. The standard DeltaNet variants learn a decreasing write-rate schedule from early layers onward. Since the models do not use explicit positional encodings, later layers need to infer positional information from the evolving causal representations produced by earlier layers. Long-context pretraining increases this effect and generally yields lower effective write rates at later sequence positions and inside deeper layers. This pattern is consistent with improved long-context performance, suggesting that reduced late-sequence write rates can help preserve earlier information and reduce degradation.

The write-rate decay explicitly decays the effective write rate towards zero across the sequence length in every layer, whereas both learned schedules reduce the write rates with long sequences. Especially in deeper layers, DeltaNet reduces its write rates far less aggressively than the long-sequence version and the decay variant. The top panel shows the resulting length generalisation, where standard DeltaNet collapses beyond the pretraining range while the decay variant remains nearly stable across the full evaluation range.

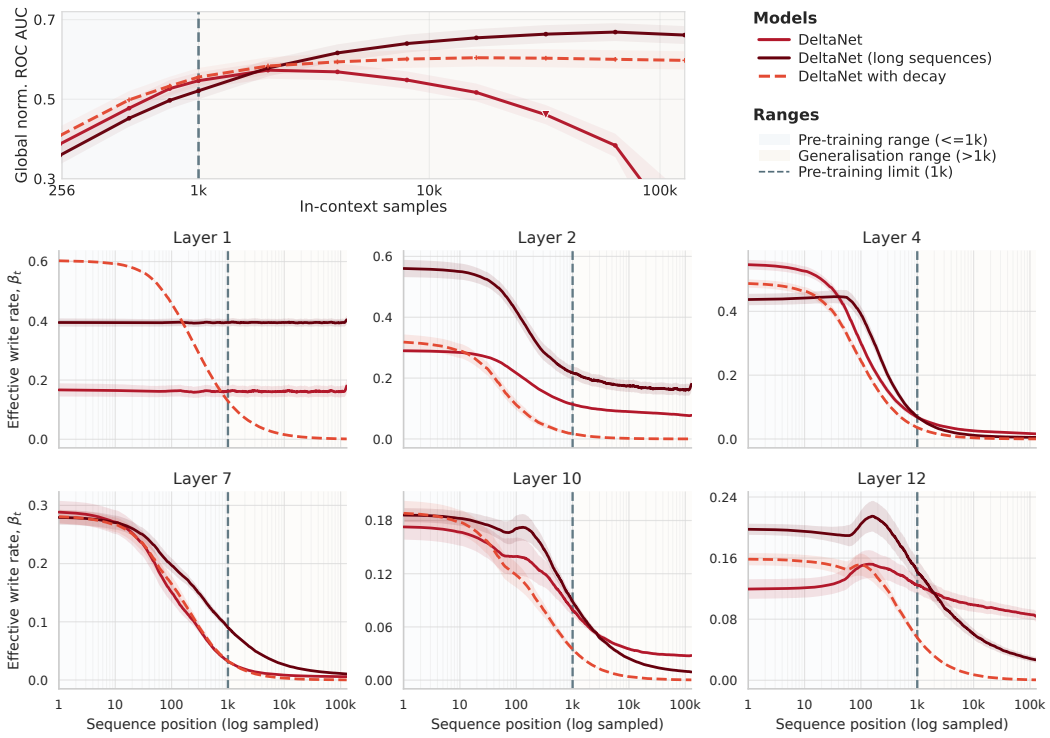


Figure 9: Effective DeltaNet write rate β_t across sequence positions and its relation to length generalisation. The top panel shows globally normalised ROC-AUC as the number of in-context samples increases. The lower panels show effective write rates β_t across sequence positions for selected layers of DeltaNet, a variant pretrained on longer sequences, and the write-rate decay intervention. For the long-sequence pretraining variants, dataset sizes are sampled log-uniformly between 200 and 64k rows. Write-rate values are averaged across 50 datasets, with means and 95% confidence intervals visualised. Standard DeltaNet learns a decreasing write-rate profile in deeper layers, where long-sequence pretraining further lowers write rates at later positions. The write-rate decay explicitly decays the effective write rate towards zero across the sequence length in every layer.

Figure 10 shows the resulting length generalisation of standard and final-state readout DeltaNet under the decay intervention of Section 4.4 and under long-sequence pretraining.

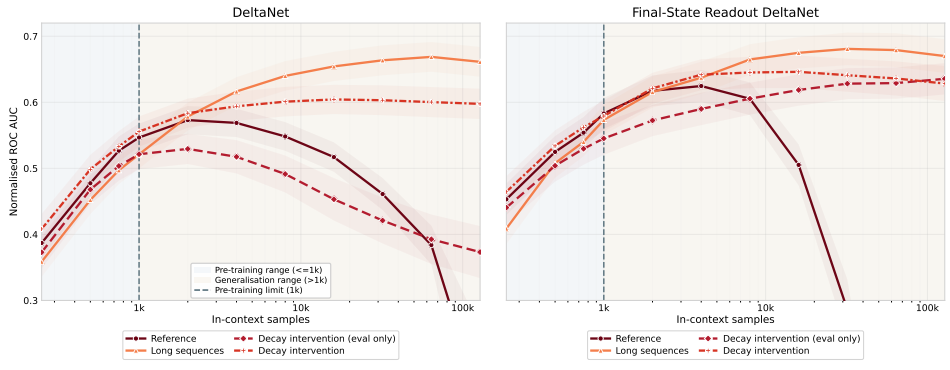


Figure 10: Length generalisation of standard DeltaNet (left) and final-state readout DeltaNet (right) under the decay intervention and long-sequence pretraining; *eval only* applies the decay to the reference checkpoint at evaluation time. Long-sequence pretrained models are not compute-matched and serve only as analysis references.

Final-state contribution analysis. We additionally analyse a lower-bound proxy for the contribution of earlier positions to the final DeltaNet state. In the simplified case of identical unit-norm key directions, the contribution of a source position s to the final state is

$$C_s = \beta_s \prod_{t=s+1}^{n} (1 - \beta_t). \tag{10}$$

Figure 11 shows these contributions across layers and lookback distances. Long-context pretraining variants retain more contribution from earlier samples, while the 1000-row pretrained variants mostly focus on recent samples. This supports the interpretation that DeltaNet overfits its write rates to the pretraining length range.

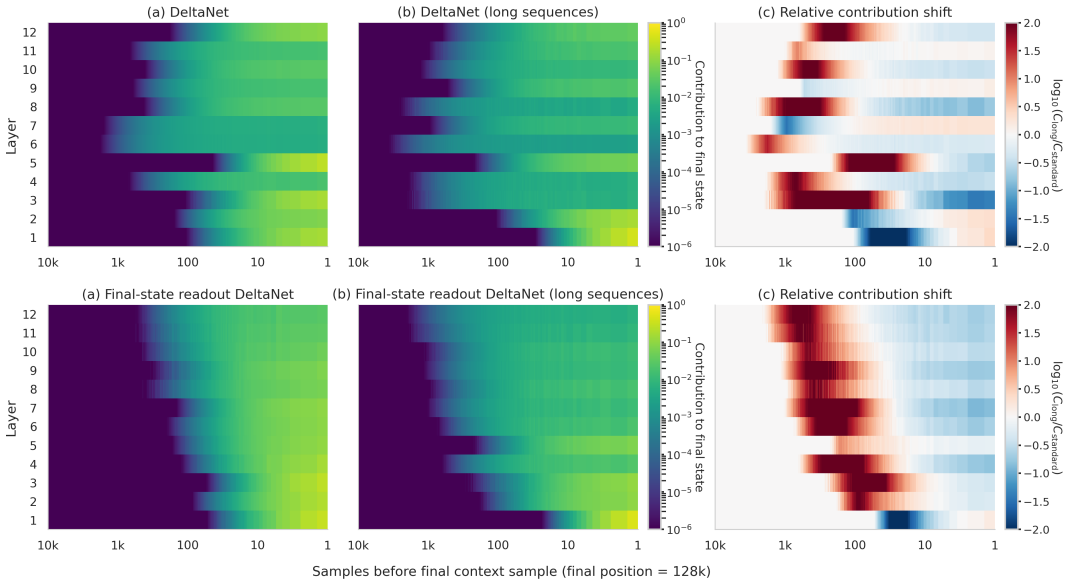


Figure 11: Contribution to the final DeltaNet hidden state across lookback distances. The top row compares DeltaNet with its long-sequence pretrained variant, and the bottom row compares the corresponding final-state readout variants. Panels (a,b) show a lower bound on the contribution $C_s = \beta_s \prod_{j>s}(1 - \beta_j)$ of each source position s to the final hidden state, averaged across 50 synthetic datasets and for each layer. Panel (c) shows the log-ratio $\log_{10}(C_{\text{long}}/C_{\text{standard}})$ where red indicates a larger contribution for the long-sequence pretrained model and blue indicates a larger contribution for the standard model.

D.8 Extended Real-World Model Comparison

Tables 8 and 9 extend the comparison of Table 3. We here add the layer-local ridge state model, which is a closed-form solution to the layer-local objective of Equation 3 (Appendix B.5), multiple subsampled variants of DeltaNet, and an Interleaved-Multi-Target (Int-MT) configuration of DeltaNet without decay. The 3K variants partition each context into disjoint blocks of at most 3000 rows and average the block logits.

Table 8: Full comparison on the 30 OpenML-CC18* datasets. Models are ordered by ROC-AUC. FS denotes final-state readout, Decay denotes the time-dependent decay intervention, Subs. denotes the subsampled and ensembled configuration, and Int-MT denotes the Interleaved Multi-Target configuration (Appendix B.2). All other models use Combined Single-Target. Bold and underline mark the best and second-best mean per metric, while bold p-values mark adjusted $p \leq 0.05$ indicating significant differences. The pairwise significance comparison uses two-sided Wilcoxon signed-rank tests with Holm adjustment like in Table 3. The 1K sample cap of OpenML-CC18* makes each Subs. 3K variant identical to its full-context counterpart.

	Mean		Adjusted p (ROC-AUC above diagonal, ACC below)										
# Model	AUC	ACC	1	2	3	4	5	6	7	8	9	10	11
1 Softmax Non-Causal	**0.8900**	**0.8166**	–	1.00	.60	1.00	1.00	1.00	.43	.17	.25	.25	.05
2 Delta-FS + Decay	0.8884	0.8145	1.00	–	1.00	1.00	1.00	1.00	1.00	.21	.27	.27	.77
3 Delta (Int-MT)	0.8880	0.8098	.51	1.00	–	1.00	1.00	1.00	1.00	1.00	.99	.99	1.00
4 Delta-FS	0.8874	0.8137	1.00	1.00	**.04**	–	1.00	1.00	1.00	1.00	.99	.99	1.00
5 Delta-FS (Subs. 3K)	0.8874	0.8137	1.00	1.00	**.04**	1.00	–	1.00	1.00	1.00	.99	.99	1.00
6 Layer-Local Ridge	0.8873	0.8131	1.00	1.00	1.00	1.00	1.00	–	1.00	.74	1.00	1.00	1.00
7 Delta + Decay (Int-MT)	0.8873	0.8120	.94	1.00	1.00	.94	.94	1.00	–	1.00	1.00	1.00	1.00
8 Delta + Decay	0.8864	0.8091	.26	.56	1.00	.24	.24	.66	1.00	–	1.00	1.00	1.00
9 Delta	0.8860	0.8078	.66	.66	1.00	**.04**	**.04**	1.00	1.00	1.00	–	1.00	1.00
10 Delta (Subs. 3K)	0.8860	0.8078	.66	.66	1.00	**.04**	**.04**	1.00	1.00	1.00	1.00	–	1.00
11 Linear Non-Causal	0.8833	0.8039	.17	1.00	1.00	.10	.10	.28	1.00	1.00	1.00	1.00	–

Table 9: Full comparison on the 38 TabArena* datasets. Here we use the identical setup from Table 8.

	Mean		Adjusted p (ROC-AUC above diagonal, ACC below)										
# Model	AUC	ACC	1	2	3	4	5	6	7	8	9	10	11
1 Softmax Non-Causal	**0.8172**	**0.8506**	–	1.00	.79	**.03**	**<.01**	**<.01**	**<.001**	**<.001**	**<.001**	.08	**<.001**
2 Delta + Decay (Int-MT)	0.8141	0.8459	.74	–	1.00	.74	**<.01**	**<.001**	**<.001**	**<.001**	**<.01**	.07	**<.001**
3 Delta-FS + Decay	0.8139	0.8468	1.00	1.00	–	**.01**	**<.01**	**<.001**	**<.001**	**<.001**	**<.001**	1.00	**<.001**
4 Delta-FS (Subs. 3K)	0.8123	0.8456	1.00	1.00	.60	–	.55	**.01**	**.01**	**<.001**	.43	1.00	**<.001**
5 Delta + Decay	0.8100	0.8438	.38	.60	.12	1.00	–	1.00	.11	**.01**	1.00	1.00	**<.001**
6 Delta (Subs. 3K)	0.8090	0.8421	**.01**	.09	**<.01**	**.02**	.85	–	1.00	.51	1.00	1.00	**<.01**
7 Delta	0.8074	0.8388	**<.001**	**<.01**	**<.001**	**.01**	.42	1.00	–	1.00	1.00	1.00	**.03**
8 Layer-Local Ridge	0.8034	0.8418	**<.01**	.38	**.01**	.07	.68	1.00	1.00	–	1.00	1.00	.08
9 Delta-FS	0.8022	0.8352	.71	1.00	1.00	1.00	1.00	1.00	.35	1.00	–	1.00	.83
10 Delta (Int-MT)	0.8004	0.8415	.09	1.00	.06	1.00	1.00	1.00	1.00	1.00	1.00	–	1.00
11 Linear Non-Causal	0.8004	0.8396	**<.01**	**.02**	**<.001**	**.01**	.35	.79	1.00	.07	.85	1.00	–

D.9 Two-Axis Row-Feature Processing

The main study uses row-wise processing, in which each row is compressed by a linear encoder into a single token. Modern tabular foundation models instead process two axes, attending across features as well as across rows (Hollmann et al., 2025; Qu et al., 2026). We here test whether the length-generalisation failure holds for both causal DeltaNet and causal linear attention in a TabPFN-v2-style two-axis configuration. We use softmax attention across features and the recurrence across rows, under the same prior and protocol as the row-wise models.

Table 10 reports mean ROC-AUC over the same 500 synthetic evaluation datasets. Two-axis processing does not resolve the failure, and its effect depends strongly on the update rule. For DeltaNet it increases the degradation, whereas causal linear attention remains comparatively stable. It is important to note that while we matched the number and size of the pretraining datasets, the two-axis models incur substantially higher computational cost in both memory and runtime per dataset and a higher parameter count of 14.9M due to the second axis, so they are not parameter-matched to the row-wise models.

Table 10: Length generalisation under two-axis versus single-axis (row-wise) processing, reporting mean ROC-AUC over the 500 synthetic evaluation datasets by context length. Two-axis processing increases long-context degradation substantially for DeltaNet, while causal linear attention remains comparatively stable. The two-axis models are not parameter-matched to the row-wise models.

Model	1k	8k	32k	128k
DeltaNet, two-axis	0.7808	0.7813	0.7592	0.7059
DeltaNet, single-axis	0.7791	0.7806	0.7719	0.7569
Causal Linear Attention, two-axis	0.7777	0.7863	0.7852	0.7812
Causal Linear Attention, single-axis	0.7775	0.7805	0.7781	0.7723

E MEMORY AND COMPUTE SCALING ANALYSIS

Figures 12 and 13 report the runtime and context-state size of the core context-processing and query-processing subroutines. Here, *fit* denotes processing the labelled context and constructing the corresponding cache or recurrent state, not task-specific parameter training. *Predict* denotes evaluating query samples given this cached context representation.

For softmax attention, context processing scales quadratically in the number of context samples, while query processing scales linearly when the context KV cache is already available. In contrast, non-causal linear attention and DeltaNet process the context in linear time and evaluate queries in constant time once the fixed-size recurrent state has been constructed. Similarly, the softmax KV cache grows linearly with the number of context samples, whereas recurrent models store a fixed-size hidden state.

The measurements reported in the figures correspond to a single preprocessing configuration. In real-world tabular foundation-model inference, predictions are typically ensembled across multiple preprocessing configurations. For TabPFN v1 and v2, this is referred to as an ensemble configuration (Hollmann et al., 2023; 2025), often using $N_{\text{ens}} \in \{10, \ldots, 32\}$ transformed versions of the same dataset. Algorithm 1 shows this procedure.

Algorithm 1 Prediction with preprocessing ensembles

Require: Context $(X_{\text{ctx}}, y_{\text{ctx}})$, queries X_{qry}, preprocessing configurations $\{\pi_e\}_{e=1}^{N_{\text{ens}}}$

for $e = 1, \ldots, N_{\text{ens}}$ **do**
  $(\tilde{X}_{\text{ctx}}^{(e)}, \tilde{y}_{\text{ctx}}^{(e)}, \tilde{X}_{\text{qry}}^{(e)}) \leftarrow \pi_e(X_{\text{ctx}}, y_{\text{ctx}}, X_{\text{qry}})$
  $C^{(e)} \leftarrow \text{Forward}_{\text{ctx}}(\tilde{X}_{\text{ctx}}^{(e)}, \tilde{y}_{\text{ctx}}^{(e)})$
  $\hat{z}^{(e)} \leftarrow \text{Forward}_{\text{qry}}(C^{(e)}, \tilde{X}_{\text{qry}}^{(e)})$
end for
$\bar{z} \leftarrow \frac{1}{N_{\text{ens}}} \sum_{e=1}^{N_{\text{ens}}} \hat{z}^{(e)}$
return $\text{softmax}(\bar{z})$

Naïvely, the context and queries are processed together, resulting in a combined fit and predict time of

$$T_{\text{qry,total}} = N_{\text{ens}} \cdot (T_{\text{ctx}} + T_{\text{qry}}), \tag{11}$$

where T_{ctx} denotes the *fit* time required to process the context, and T_{qry} denotes the *predict* time required to process the queries given the processed context. However, for every additional query, the full training context state would need to be recomputed again.

Alternatively, the fit and predict stage can be separated and amortised, but this requires caching the context representations (KV-cache or hidden-state $\boldsymbol{S}_{n_c}$) for all ensemble members. In this algorithm, the effective costs scale as

$$T_{\text{ctx,total}} = N_{\text{ens}} \cdot T_{\text{ctx}}, \tag{12}$$
$$T_{\text{qry,total}} = N_{\text{ens}} \cdot (T_{\text{qry}} + T_{\text{load}}), \tag{13}$$
$$M_{\text{cache,total}} = N_{\text{ens}} \cdot M_{\text{cache}}, \tag{14}$$

where M_{cache} denotes the size of the context representation and T_{load} denotes the time required to load the cached context representation for one ensemble member when the cache needs to be offloaded from GPU memory to main memory or disk. The implementation of such an ensemble caching algorithm is beyond the scope of this project. We therefore omit a separated end-to-end fit and predict runtime analysis and report the pure T_{ctx}, T_{qry}, and M_{cache} in Figures 12 and 13 instead.

This ensemble effect further highlights the memory advantage of parametric tabular foundation models. In our setting, the softmax-transformer KV cache at sequence length 128k requires 2.34 GB, whereas the DeltaNet recurrent state requires only 1.17 MB. With $N_{\text{ens}} = 10$, this corresponds to 23.4 GB for softmax attention compared to 11.7 MB for DeltaNet. For datasets with millions of samples, the KV cache would grow significantly, limiting real-world applicability both in runtime and memory cost. Additionally, it would likely require disk offloading of the cache, further increasing inference latency.

The memory burden further increases along the feature axis under two-axis processing (Appendix D.9), where both the cache and the recurrent state scale with the number of feature groups. A dataset with 100 features and a feature grouping of five results in 20 groups, multiplying the memory requirements of both approaches by 20. A 10-member ensemble would then cache roughly 469 GB of keys and values against 234 MB of recurrent state. Such small states, combined with constant-time query processing, could allow linear-time tabular foundation models to be deployed on edge devices.

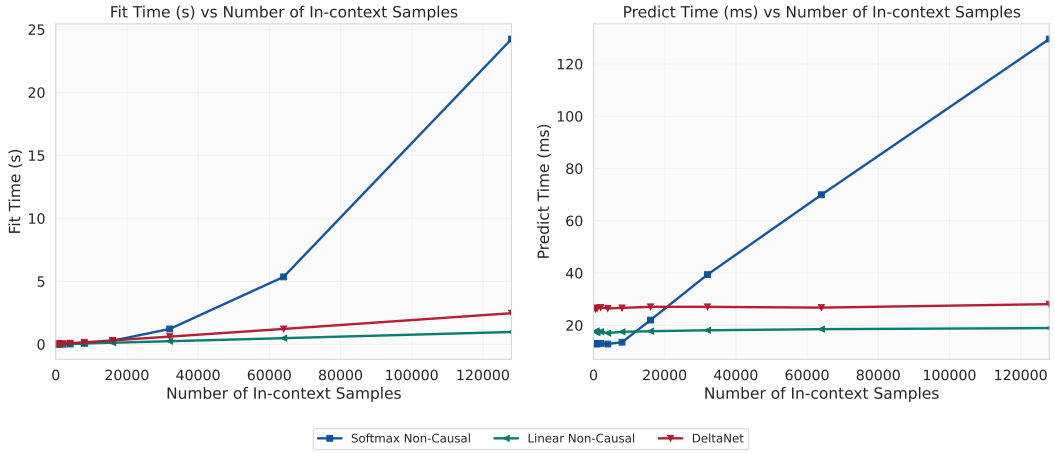


Figure 12: Runtime scaling with the number of in-context samples. Softmax attention scales quadratically during context processing and linearly during query processing with a cached context. Linear attention and DeltaNet scale linearly during context processing and remain constant during query processing from a fixed-size recurrent state.

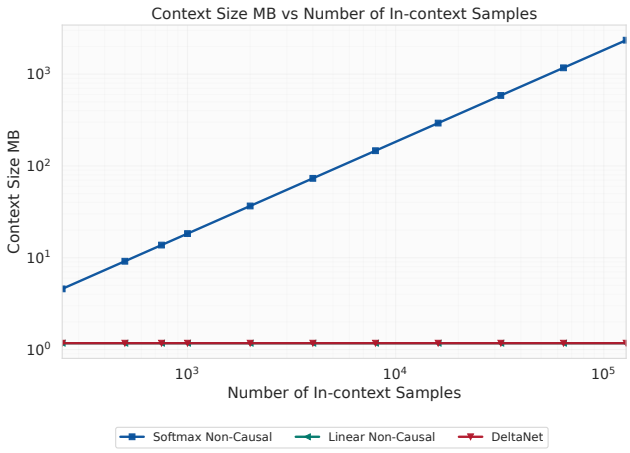


Figure 13: Context-state size across increasing context lengths. Softmax attention stores a KV cache that grows linearly with the number of context samples, whereas linear-time recurrent models store a fixed-size hidden state.

F Adding Stateless Prediction to Flash-Linear-Attention Models

Causal linear attention architectures used in our experiments are imported from the FLA library (Yang & Zhang, 2024). These implementations natively support autoregressive sequence processing. However, as discussed in Section 3.2, non-transductive evaluation requires a different prediction mode: each query sample must be predicted independently from the context set, without conditioning on any other query samples. We refer to this setting as *single-target* prediction in the main paper.

We considered three approaches for adding this prediction mode:

Algorithm 2 Stateless single-target prediction for one linear attention layer

Require: Cached context state $S_c \in \mathbb{R}^{B\times H\times d_v\times d_{qk}}$
Require: Query projections $q, k \in \mathbb{R}^{B\times n_q\times H\times d_{qk}}$ and $v \in \mathbb{R}^{B\times n_q\times H\times d_v}$
1: $O^{\text{ctx}} \leftarrow \text{einsum}(\texttt{"bihk,bhvk->bihv"}, q, S_c)$ ▷ shared context readout
2: $s \leftarrow \text{einsum}(\texttt{"bihk,bihk->bih"}, q, k)$ ▷ query-key scores
3: $O^{\text{self}} \leftarrow \text{einsum}(\texttt{"bih,bihv->bihv"}, s, v)$ ▷ self terms
4: $O \leftarrow O^{\text{ctx}} + O^{\text{self}}$ ▷ final output
5: **return** O

1. **Naive implementation:** The simplest solution is to evaluate each query sample separately. For a test set with n_q query samples, this requires n_q forward passes. One can either run the full sequence consisting of the context set and one query sample each time or first process the context set once, cache the resulting recurrent state, and then perform n_q single-sample forward passes while resetting the recurrent state to the cached context state after each query. The latter avoids repeatedly processing the context, but the query evaluations remain sequential.
2. **Broadcasting the recurrent state:** A natural way to parallelise the naive approach is to broadcast the final context recurrent state along a new query dimension or, equivalently, expand the batch dimension by a factor of n_q. Each query could then be evaluated in parallel from the same context state. However, the FLA implementations materialise the expanded recurrent state rather than representing it as virtual views. This leads to n_q explicit copies of the final recurrent states. For query sets with hundreds or thousands of samples, the resulting memory cost is prohibitive, and the additional computation is inefficient.
3. **Custom stateless patch:** We therefore implement a custom stateless prediction patch. The goal is to obtain the parallel one-step prediction behaviour of the broadcasted implementation without materialising n_q copies of the recurrent state. Since single-target evaluation only requires the prediction for each query sample, and not the recurrent state after the query has been processed, the query update can be specialised. We use the standard FLA kernels to process the context samples, while during query prediction, we reuse the FLA model logic but replace the recurrent kernel update with a custom PyTorch implementation that applies the cached context state independently to all query samples without writing back query-dependent states.

To illustrate the *custom stateless patch* that we use in this paper, Algorithm 2 shows the corresponding stateless one-step computation for a linear attention layer with an additive recurrent state update. For readability, we omit the standard query scaling factor.

For a standard linear-attention recurrent layer, the state update and readout are

$$\boldsymbol{S}_t = \boldsymbol{S}_{t-1} + \boldsymbol{v}_t\boldsymbol{k}_t^\top, \qquad \boldsymbol{o}_t = \boldsymbol{S}_t\boldsymbol{q}_t. \tag{15}$$

After processing the context samples, we obtain a cached context state $\boldsymbol{S}_c$. For an independent query sample i, the standard one-step computation would first form the query-updated state

$$\boldsymbol{S}_i = \boldsymbol{S}_c + \boldsymbol{v}_i\boldsymbol{k}_i^\top, \qquad \boldsymbol{o}_i = \boldsymbol{S}_i\boldsymbol{q}_i. \tag{16}$$

Expanding the readout gives

$$\begin{aligned} \boldsymbol{o}_i &= \boldsymbol{S}_i\boldsymbol{q}_i \\ &= (\boldsymbol{S}_c + \boldsymbol{v}_i\boldsymbol{k}_i^\top)\boldsymbol{q}_i \\ &= \boldsymbol{S}_c\boldsymbol{q}_i + (\boldsymbol{v}_i\boldsymbol{k}_i^\top)\boldsymbol{q}_i \\ &= \boldsymbol{S}_c\boldsymbol{q}_i + (\boldsymbol{k}_i^\top\boldsymbol{q}_i)\boldsymbol{v}_i. \end{aligned} \tag{17}$$

The stateless patch computes this expanded form directly for all query samples in parallel. Thus, it reads from the shared context state $\boldsymbol{S}_c$ and adds each query's own one-step contribution, without materialising the query-updated states $\boldsymbol{S}_i$.