FORGETTING WITHOUT FORGETTING: QUERY-DEPENDENT DECAYS FOR EFFECTIVE LONG-CONTEXT MODELING

Anonymous authors
Paper under double-blind review

ABSTRACT

Position encodings are an integral part of transformers, heavily influencing language modeling performance as well as capabilities like recall and extrapolation to long contexts. Yet no existing encoding excels at all three aspects. The ubiquitous encoding `RoPE` performs well within training lengths, but degrades beyond. Decay-based encodings (like `ALiBi` or `FoX`) can extrapolate better, as they favor recent tokens more strongly, but this hurts long-range recall. Removing position encodings entirely (`NoPE`) can improve recall, but hurts language modeling. To resolve this, we introduce Query-Dependent Decay (`QDecay`): a lightweight component that can be added on top of any decay-based encoding, enabling each query to independently control the intensity of the underlying decay. A query can choose to either favor recent tokens (like `ALiBi`), or "turn off" the decay completely when distant information matters (like `NoPE`). Critically, this choice is made independently for each query in the sequence — favoring recent tokens for one query does not affect the decay for the next. Attaching `QDecay` on top of the simple `ALiBi` decay, the resulting position encoding achieves strong performance on all three desiderata. On standard language modeling benchmarks, `QDecay` matches or outperforms other position encodings, including the popular `RoPE`. Next, `QDecay` achieves the best recall on RULER at 32K+ token context, improving on other encodings by 18+ accuracy points, and remains competitive at shorter lengths. Third, `QDecay` extrapolates well: on Books3, it matches or outperforms other encodings on validation perplexity at all context lengths, maintaining and even improving performance through 64K tokens ($16\times$ training length), whereas all other encodings degrade. Our interpretability ablations identify `QDecay`'s ability to completely "turn off" the underlying decay as a key driver of recall. `QDecay` adds negligible ($< 0.1\%$) overhead in parameters and is compatible with fast FlashAttention-based algorithms.

1 INTRODUCTION

The Transformer architecture (Vaswani et al., 2017) is the backbone of large language models, which are routinely used in applications such as working with long documents, large codebases, and extended chat conversations. These tasks often involve sequences far longer than those seen during pretraining, requiring models to adeptly perform language modeling at unfamiliar lengths as well as recall relevant information from past context. These abilities strongly depend on the choice of *position encoding*: the component which tells a transformer where tokens occur in a sequence.

However, existing encodings make different tradeoffs. Rotary position encoding (`RoPE`) (Su et al., 2024), the de facto standard, performs well within the training length but struggles to extrapolate beyond it (Press et al., 2022) without further interventions (Peng et al., 2024). Decay-based encodings such as `ALiBi` (Press et al., 2022) and `FoX` (Lin et al., 2025) can extrapolate further by explicitly down-weighting or *decaying* distant tokens, effectively ignoring tokens outside training lengths. However, the same decay can make a model "forget" relevant information simply because it is far back in the context. Removing position encodings altogether (`NoPE`) (Kazemnejad et al., 2023) avoids decay and can preserve long-range recall, but hurts language modeling (Haviv et al.,

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

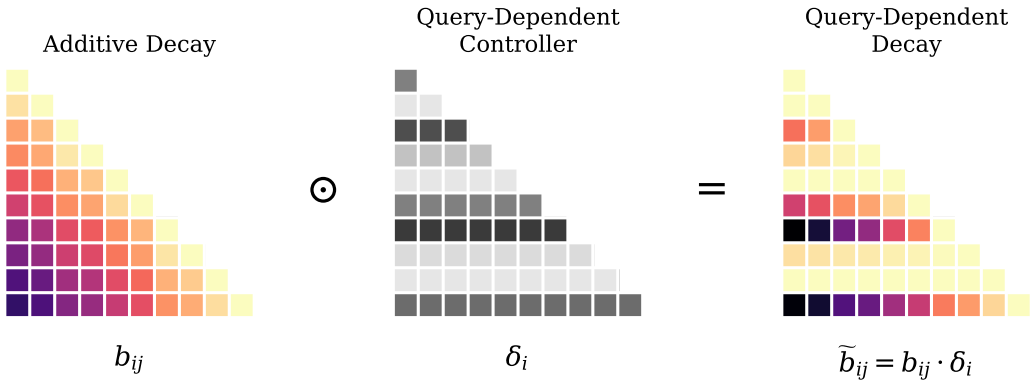


Figure 1: **`QDecay` controls the intensity of any decay-based additive encoding at read time.** Darker colors indicate a stronger decay and yellow indicates no decay (or `NoPE`). **Left:** An additive decay b_{ij} that accumulates decay across tokens and forgets as context grows. **Middle:** `QDecay` predicts one controller δ_i per query (row). **Right:** The elementwise product produces the controlled decay $\widetilde{b}_{ij}$: setting $\delta_i = 0$ removes the decay and recovers `NoPE` for query i, $\delta_i = 1$ leaves it unchanged, and $\delta_i > 1$ strengthens it. This results in a row-wise variation in the underlying decay.

2022) due to the inability to model local interactions. None of the existing encodings alone combine strong language modeling, length extrapolation, and long-range recall.

To resolve this, we propose Query-Dependent Decay, or `QDecay` for short. `QDecay` is a lightweight component added on top of any additive decay-based position encoding (e.g., `FoX` or `ALiBi`): allowing the model to *control* the intensity of the underlying decay – serving as a continuous "turn on or off" switch. Concretely, `QDecay` allows each token to control the decay for its past: the token can choose to decay past tokens, or decide to have no forgetting at all (see Figure 1). This enables the model to perform recall by "turning off" the decay completely to achieve `NoPE`, and at the same time extrapolate to long sequences by "turning on" the underlying decay, as required by the current context – getting the best of both worlds. This choice is independent for each query, and decay for one query does not affect the next.

When `QDecay` controller is instantiated with the simple `ALiBi` decay (Press et al., 2022), the resulting encoding, termed `QDecay-ALiBi`, is the only one that fulfills:

- **Long-Context Extrapolation.** `QDecay-ALiBi` extrapolates up to sequences of length 64K (16× the training length), where other position encodings like `RoPE`, `FoX`, `ALiBi` or `Wall` (Pai et al., 2026) fail (Figure 4).
- **Long-Context Recall.** `QDecay-ALiBi` achieves the best long-context recall at 32K and 64K on a suite of recall tasks in the RULER benchmark (Hsieh et al., 2024) (Figure 5), beating the closest encoding `PaTH` (Yang et al., 2025) by ∼18 absolute points, and being competitive at shorter lengths.
- **Strong Langauge Modeling.** Further, `QDecay-ALiBi` matches or improves upon existing methods at standard short-context language modeling abilities and real-world recall tasks, outperforming the de-facto `RoPE` (Table 1).

To better understand `QDecay`, we demonstrate that the flexibility to "turn off" the `ALiBi` encoding in `QDecay-ALiBi` is the primary driver of its strong long-context recall capabilities (Section 5). An interpretability analysis of the trained `QDecay-ALiBi` shows its heads have emergent specialization in each layer: some decay strongly like `ALiBi`, some behave like `NoPE`, and others are dynamic based on the input. In effect, the model learns its own "hybrid" and interestingly, contrary to existing `RoPE-NoPE` hybrids (Meta AI, 2025; Cohere Team, 2025), `QDecay-ALiBi` favors `NoPE` in the first layer, and it outperforms them in language modeling and long-context recall (Section 5).

`QDecay-ALiBi` is compatible with a FlashAttention-style (Dao, 2024; Dao et al., 2023) algorithm and involves no notable parameter ($< 0.1\%$) or runtime overhead. We release efficient inference

kernels for QDecay based on FlashAttention (Appendix D.2). We show that QDecay is a drop-in replacement for existing positional encodings (RoPE) across both full-attention-only architectures and hybrids with sliding-window attention (Section 4.5). Finally, QDecay as a mechanism opens the door to new positional encodings, as it can be applied to any additive decay-based encoding.

2 BACKGROUND

We first introduce our notation and review additive and multiplicative position encodings.

Let $\boldsymbol{q}_i, \boldsymbol{k}_j \in \mathbb{R}^d$ denote the query at position i and the key at position j, respectively. Let s_{ij} denote their pre-softmax attention logit. For causal attention, where $j \leq i$, a relative positional bias may be incorporated additively through b_{ij} or multiplicatively through $\boldsymbol{M}_{ij} \in \mathbb{R}^{d\times d}$:

$$s_{ij} = \boldsymbol{q}_i^\top \boldsymbol{k}_j + b_{ij} \qquad \text{(additive)} \qquad \text{or} \qquad s_{ij} = \boldsymbol{q}_i^\top \boldsymbol{M}_{ij} \boldsymbol{k}_j \qquad \text{(multiplicative).} \tag{1}$$

Additive encodings. One of the simplest additive encodings is the ALiBi decay (Press et al., 2022), which uses $b_{ij} = -\gamma(i-j)$, where $\gamma \geq 0$. Each attention head applies a fixed, data-independent linear decay as the distance between $\boldsymbol{q}_i$ and $\boldsymbol{k}_j$ increases. FoX (Lin et al., 2025) makes this decay data-dependent by predicting a nonnegative decay γ_t at every token. The bias between positions j and i is the accumulated (sum) decay along the intervening sequence, $b_{ij} = -\sum_{t=j+1}^{i} \gamma_t$, where $\gamma_t \geq 0$. Unlike ALiBi, the resulting decay depends on the contents of the tokens between the query and key.

Multiplicative biases. Multiplicative position encodings transform query–key interactions through $\boldsymbol{M}_{ij}$. RoPE (Su et al., 2024) uses the fixed, data-independent separable transformation $\boldsymbol{M}_{ij} = \boldsymbol{R}_i^\top \boldsymbol{R}_j$, where $\boldsymbol{R}_i$ is a position-dependent rotation matrix. PaTH (Yang et al., 2025) uses a data-dependent transformation based on cumulative products of Householder transforms $\boldsymbol{M}_{ij} = \prod_{t=j+1}^{i} \boldsymbol{H}_t$, where each $\boldsymbol{H}_t$ is predicted from token t. Wall (Pai et al., 2026) uses an accumulated channel-wise decay $\boldsymbol{M}_{ij} = \operatorname{diag}\left(\bigodot_{t=j+1}^{i} \boldsymbol{\alpha}_t\right)$, where $\boldsymbol{\alpha}_t \in (0,1]^d$. Each channel therefore accumulates its own data-dependent decay along the path from key j to query i. In spirit, Wall plays a role analogous to a channel-wise multiplicative version of FoX, although the two methods modify the attention score differently: FoX adds a scalar penalty to the logit, whereas Wall rescales query–key interactions independently across channels. We defer a detailed related work to Section 7.

3 QDECAY: QUERY-DEPENDENT DECAY

3.1 QDECAY CONTROLLER

QDecay is a query-dependent controller placed on top of any decay-based additive position encoding. It lets each query set the intensity of the underlying decay at read time. For an additive decay-based encoding with bias b_{ij}, QDecay modifies the attention logit between query i and key j to:

$$s_{ij} = \boldsymbol{q}_i^\top \boldsymbol{k}_j + \widetilde{b}_{ij}, \qquad \widetilde{b}_{ij} = \underbrace{b_{ij}}_{\text{any decay}} \cdot \underbrace{\delta_i}_{\text{“controller”}}, \qquad \delta_i \geq 0, \tag{2}$$

where the controller δ_i is predicted from the current token $\boldsymbol{x}_i$[1].

In practice, existing additive decays b_{ij} are accumulative: they grow with the distance between query and key, either by a fixed per-step amount as in ALiBi ($b_{ij} = -\gamma(i-j)$) or by a data-dependent sum over the intervening tokens as in FoX (Section 2); hence in Figure 1 the colors darken as one moves further back in the context. The QDecay controller δ_i therefore rescales how quickly this accumulated decay grows for query i.

[1]Strictly, δ_i depends on the hidden state $\boldsymbol{x}_i$ of the current token rather than on the query vector $\boldsymbol{q}_i$ itself. We call it *query-dependent* because it is set by the token that is *attending* to the past, i.e., the query position i

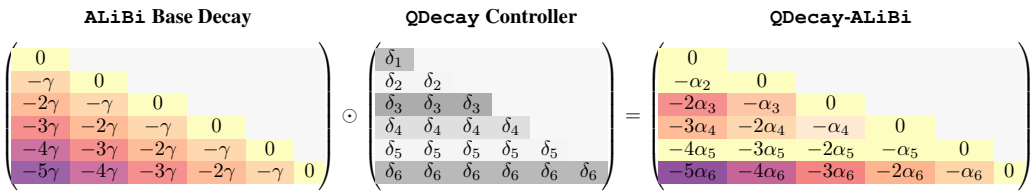


Figure 2: QDecay-ALiBi: Each query i rescales its row of the ALiBi bias by the input- or query-dependent controller δ_i, giving an ALiBi-shaped bias with slope $\alpha_i = \gamma\delta_i$. Blank entries are causally masked. Shading is illustrative: darker cells indicate stronger decay, and darker gray indicates a larger δ_i.

When $\delta_i \to 0$, the underlying encoding is removed and the query recovers NoPE; $\delta_i = 1$ leaves the encoding unchanged; and $\delta_i > 1$ increases the intensity of its decay. Crucially, this choice is made afresh by every query: one query can let the underlying decay forget the past ($\delta_i > 0$), and the very next can undo that forgetting by setting $\delta_{i+1} \to 0$ and reading the past without any decay. Nothing is ever permanently forgotten unlike in FoX or Wall (see discussion in Section 7). QDecay thus enables *forgetting without forgetting*: it forgets when that helps, while keeping every token recoverable for recall.

3.2 QDecay-ALiBi

Parameterization. For simplicity, we instantiate QDecay with the additive decay-based encoding, ALiBi, whose per-head decay is just $b_{ij} = -(i-j)\gamma$ where $\gamma > 0$. Now, attaching the QDecay controller gives:

$$\widetilde{b}_{ij} = \underbrace{-(i-j)\gamma}_{\texttt{ALiBi base decay}} \cdot \underbrace{\delta_i}_{\text{“controller”}} = -(i-j)\alpha_i, \qquad \text{where} \quad \alpha_i = \gamma\delta_i, \quad \delta_i = \mathrm{ReLU}\left(\boldsymbol{w}^\top \boldsymbol{x}_i + c\right). \tag{3}$$

For simplicity, we fold the decay γ and the controller δ_i into a single quantity $\alpha_i = \gamma\delta_i$, the *effective* query-dependent decay for QDecay-ALiBi: a large α_i favors nearby keys, while $\alpha_i = 0$ removes the underlying ALiBi decay and recovers NoPE for query i. Figure 2 illustrates how the controller rescales each row of the ALiBi bias. The controller parameters $\boldsymbol{w}$ and c are learned separately for each head.

Initialization. We initialize per-head decay in QDecay-ALiBi as $\gamma = e^A$ where e^A geometrically from $1/2$ to $1/128$ across heads, inspired from Mamba-2 initialization (Dao & Gu, 2024). Moving slightly away from traditional ALiBi, we make the scalar A learnable in QDecay-ALiBi. Further, we set the QDecay controller parameters $\boldsymbol{w} = \mathbf{0}$ and $c = 1$, giving $\delta_i = 1$ for every query at initialization. Thus, each head starts with the ALiBi slope γ and can learn when to retain decay or move towards NoPE. Other initializations did not result in better performance (Appendix A.2).

Training. We predict all query-dependent decays in parallel with a linear projection, adding $O(ND)$ work for a sequence of length N and hidden dimension D. We implement the bias as a FlexAttention score modifier, which evaluates b_{ij} tile by tile and avoids materializing an $N \times N$ bias matrix. QDecay-ALiBi adds minimal parameters per layer: a linear projection and one learned scalar per head, requiring just $O(D)$ additional parameters ($\leq 0.1\%$).

We explore applying the QDecay controller to other additive decays in Appendix A.6.

4 Experiments

We provide a high-level overview of our experiments along with key takeaways.

- Section 4.1 demonstrates that QDecay-ALiBi achieves near-perfect recall on synthetic MQAR task at $\mathbf{100}\times$ the training length, where every other encoding fails.

- Section 4.2 shows QDecay-ALiBi achieves scores competitive with or better than existing encodings on standard short-context language modeling and recall benchmarks, outperforming RoPE at 250M and 800M scales.
- Importantly, in Section 4.3, we show QDecay-ALiBi achieves the best long-context recall by **18+ absolute points** at 8–16× the pretraining length on the RULER benchmark, while staying on par or better at shorter lengths.
- In Section 4.4, we show that QDecay-ALiBi is the only encoding whose Books3 perplexity keeps improving through **64K** tokens whereas every other encoding eventually degrades.
- In Section 4.5, we show that in sliding-window + full-attention hybrids, simply using QDecay-ALiBi in every layer outperforms the standard RoPE (SWA) + NoPE (Full) recipe across all abilities: language modeling, long-context recall and extrapolation.
- Further, in Section 4.6, we show that QDecay-ALiBi is complementary to multiplicative encodings: alternating QDecay-ALiBi with Wall or PaTH layers improves overall language modeling, recall, and extrapolation compared to Wall or PaTH alone respectively.

Since QDecay-ALiBi is the main instantiation studied in this paper, unless stated otherwise, QDecay refers to QDecay-ALiBi in the remainder of this paper for brevity.

Baselines: We compare QDecay broadly with existing position encodings. This includes *additive* methods like: FoX (Lin et al., 2025), ALiBi (Press et al., 2022), and *multiplicative* methods like: RoPE (Su et al., 2024), Wall (Pai et al., 2026) and PaTH (Yang et al., 2025). Full training and evaluation details, along with per-dataset and per-task results, are provided in Appendix E.

4.1 Synthetic Experiment

Figure 3 tests recall on the multi-query associative recall (MQAR) task (Arora et al., 2024), where the model must recall the values of queried keys from context. We train on sequences of 128 tokens containing just 8 single token key-value pairs (rest are filler tokens) and test on sequences up to 100×. We grid search the best learning rate for all methods.

4.2 Language Modeling

Setup. We pretrain transformers at two scales: 250M parameters on 10B tokens (24L12H768D) and 800M parameters on 30B tokens (24L16H1536D), following Afzal et al. (2026). All models are trained on FineWeb-Edu (Penedo et al., 2024) with the Llama-2 tokenizer and a 4K context length, using a batch size of 0.5M tokens, 1B tokens of warmup, and cosine decay from a peak learning rate of 8×10^{-4} (250M) and 4×10^{-4} (800M). Full training details are in Appendix E.1. Full evaluation details are in Appendix E.2.

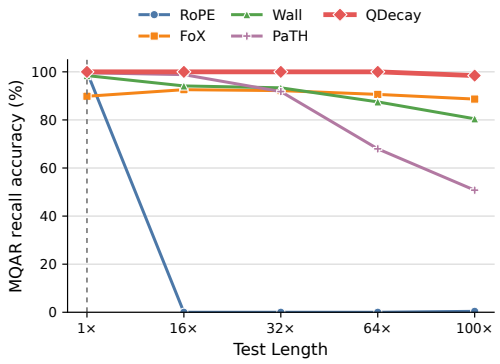


Figure 3: **MQAR recall.** Existing encodings fail to recall beyond the training length, even on a simple recall task. QDecay retains near-perfect performance up to 100× the training length.

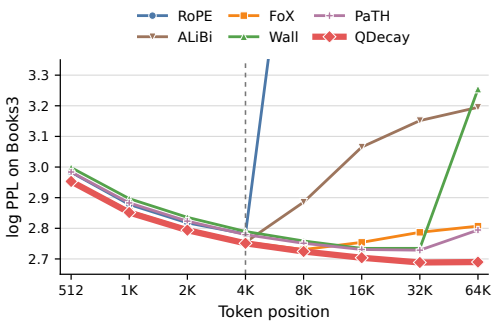


Figure 4: **Long-context extrapolation.** Cumulative log perplexity on books3 for 800M models pretrained at 4K (dashed line). QDecay is the only encoding that extrapolates to 64K.

We report averages across short-context language-modeling and recall tasks in Table 1.

QDecay is on-par or better at short context. QDecay is competitive with existing encodings on standard short-context language modeling and real-world recall tasks up to 2K tokens, even outperforming the popular RoPE in these settings. In contrast, always-on NoPE performs substantially worse on all three short-context aggregates. QDecay can recover NoPE for individual queries without discarding a useful recency bias everywhere.

Model	LM ppl. ↓	LM eval. ↑	Recall ↑
250M/10B			
RoPE	36.7	42.0	32.8
ALiBi	29.2	42.9	40.5
FoX	30.8	43.0	35.0
Wall	33.6	42.2	**42.9**
PaTH	29.6	42.3	42.2
NoPE	59.5	39.2	22.5
QDecay	**29.1**	**43.2**	42.8
800M/30B			
RoPE	18.0	49.5	54.1
ALiBi	16.4	50.8	**57.5**
FoX	**16.2**	**51.1**	51.3
Wall	17.2	49.4	55.6
PaTH	16.6	50.6	56.3
QDecay	**16.2**	50.5	56.6

Table 1: **Language modeling.** Language-modeling perplexity, evaluation accuracy, and real-world recall, each averaged across tasks. Details in Appendix E.2.

4.3 Long-context Recall

We evaluate long-context recall on RULER (Hsieh et al., 2024), using seven tasks that span single-needle (S-NIAH-1,2,3) and multi-needle (MK1, MK2, MQ, MV) settings, and report accuracy averaged across tasks. We evaluate from within the 4K pretraining length up to 16K for 250M models and 64K for 800M models. Figure 5 reports the 250M (left) and 800M (right) results.

QDecay achieves the best recall at extreme extrapolation. At 32K and 64K, 8–16× the pretraining length, QDecay has the best strong recall performance while other encodings collapse to near zero (Figure 5). These gains do not come at the cost of recall within 4K, where QDecay matches or exceeds existing encodings.

We note that while the data-dependent multiplicative encodings Wall and PaTH do perform better on average at RULER 8-16K context length at 800M scale, both impose significant additional overhead: they are substantially slower to train than the others (Appendix D.1), and in the case of Wall add a non-negligible amount of model parameters and FLOPs.

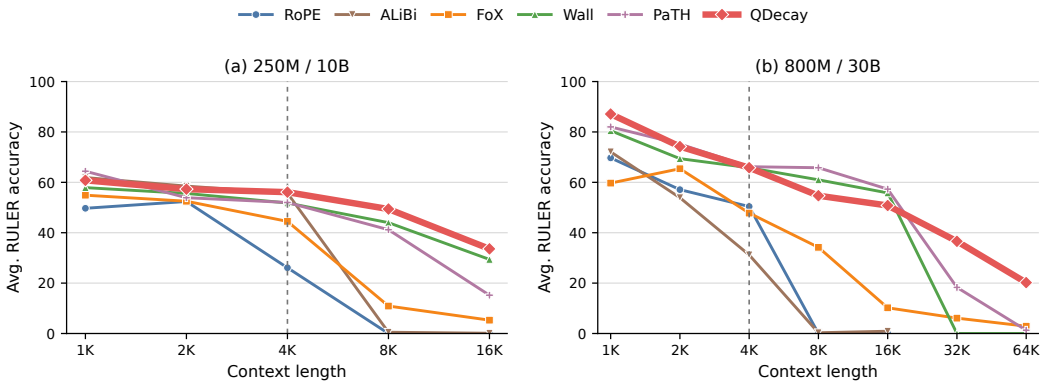


Figure 5: **Long-context recall.** Average RULER accuracy (across seven tasks) for 250M models up to 16K (left) and 800M models up to 64K (right); all models are pretrained at 4K (dashed line). QDecay is the only encoding that retains recall at 8–16× the pretraining length.

4.4 Long-context Extrapolation

Following Lin et al. (2025), we evaluate the 800M models on long books (books3 (Gao et al., 2020)) and report cumulative log perplexity, the average loss over the first L tokens, for L from 0.5K to 64K (Figure 4). Since the models are pretrained at 4K, this measures extrapolation up to 16× the pretraining length.

QDecay is the only encoding that extrapolates to 64K. The perplexity of QDecay remains stable through 64K, while every other encoding degrades, RoPE immediately beyond 4K and FoX, Wall, and PaTH at longer lengths. Importantly, **QDecay extrapolates without sacrificing recall.**

Section 4.3 shows QDecay retains recall at 64K, where other encodings collapse. We discuss why existing encodings fail in Section 7.

4.5 QDECAY IS A DROP-IN REPLACEMENT

Many recent models interleave sliding-window and full-attention layers for memory-efficient long-context inference (Agarwal et al., 2025; Meta AI, 2025; Cohere Team, 2025). To test whether QDecay is a drop-in encoding for such practically deployed configurations, we train hybrids with three 2K sliding-window layers per full-attention layer (3:1 ratio) and compare the standard RoPE (SWA) + NoPE (Full) recipe (Meta AI, 2025; Yang et al., 2026) against simply using QDecay for both the sliding-window and full-attention layers.

Table 2: **QDecay is a drop-in replacement.** Results on 800M/30B scale. All hybrids use 3:1 mixture of 2K sliding-window and full-attention layers.

	Language modeling			RULER (averaged across tasks)				
Model	LM ppl. ↓	LM eval. ↑	Recall ↑	1K	2K	4K	8K	16K
Full attention								
RoPE	18.0	49.5	54.1	69.7	57.1	50.4	0.0	0.0
QDecay	**16.2**	**50.5**	**56.6**	**87.1**	**74.2**	**65.8**	**54.7**	**50.8**
Full + sliding-window attention								
RoPE (SWA) + NoPE (Full)	18.5	50.0	41.0	75.7	**73.3**	33.9	19.0	11.7
QDecay (SWA + Full)	**16.2**	**51.5**	**50.7**	**77.7**	72.6	**62.7**	**47.7**	**32.6**

The all-QDecay hybrid outperforms the RoPE–NoPE hybrid on standard language modeling, real-world recall, and long-context RULER (Table 2). As a result, rather than deciding where to place QDecay in a SWA + Full attention hybrid, one can simply use QDecay in every layer and get improvements across language modeling and recall.

4.6 QDECAY HELPS MULTIPLICATIVE ENCODINGS EXTRAPOLATE

To test whether the additive QDecay can complement multiplicative encodings, we construct two hybrids: beginning with a multiplicative encoding (Wall or PaTH) layer followed by QDecay, interleaved throughout. Adding QDecay improves both multiplicative encodings on long-context recall and extrapolation, with the hybrids degrading more gracefully beyond the pretraining length than Wall or PaTH alone (Table 7, Appendix B).

5 UNDERSTANDING QDECAY

"Turning off" the underlying decay is the main driver for recall. The flexibility to switch to NoPE helps QDecay recall information from far back in the context. Handicapping this ability hurts long-context recall. Appendix A.4 demonstrates this by redefining the minimum possible value for the QDecay controller to some positive number $\epsilon > 0$ instead of zero, making it hard "turn off" the decay. This results in decrease in long-context recall performance as one increases ϵ.

Further, QDecay uses a ReLU to parameterize the controller δ_i, and hence the effective decay α_i (Section 3). Crucially, ReLU allows the model to choose "pure" NoPEs, that is α_i can be exactly 0. A softplus parameterization for α_i restricts the model's ability to predict "pure" NoPEs (since softplus asymptotes to zero), and consequently performs worse than ReLU, especially on recall beyond pretraining lengths of 4K (Appendix A.3). The same problem plagues FoX, whose strictly positive softplus decay can never fully recover NoPE (Section 7), and replacing it with a ReLU improves FoX's recall (Appendix A.5).

QDecay chooses its own hybrid NoPE is important for recall (Yang et al., 2026) and indeed many large open-source models interleave NoPE layers with a relative encoding such as RoPE with sliding-window (Yang et al., 2026; Cohere Team, 2025; Meta AI, 2025; Agarwal et al., 2025). However, the optimal placement of NoPE, RoPE, and sliding-windows is unclear and relies on heuristics, despite recall being highly sensitive to these choices (Cabannes et al., 2026). A consequence of controlling the decay is: QDecay lets optimization and the current context decide where to place NoPE ("turn off" decay), per token, rather

than hand-engineering it. The same holds for sliding windows: a query with large decay α_i attends only to recent tokens and acts as a sliding-window whose width it picks itself. Interestingly, the learned configuration in `QDecay` differs from existing hybrids. Interestingly, Figure 6 shows that `NoPE` emerges in the first layer in `QDecay`, whereas hybrids typically place `NoPE` layers at fixed intervals after several `RoPE` layers. The same pattern appears at 800M. See Appendix C for discussion regarding the general left-to-right colour gradient pattern we observe in Figure 6.

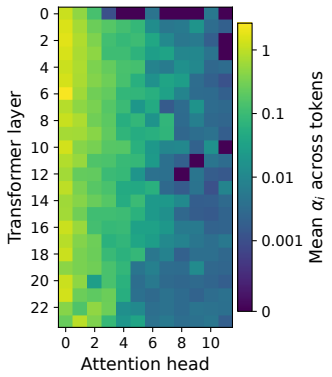


Figure 6: Mean effective decay α_i per layer and head across tokens in the 250M `QDecay` model. Darker cells indicate heads near `NoPE`. Interestingly, `NoPE` emerges in the **first layer**, contrary to current `NoPE`-`RoPE` hybrid designs (Meta AI, 2025). More results and details in Appendix C.

`QDecay` has dynamic decay depending on the context The trained `QDecay` model uses the flexibility of the decay α_i to change position encodings depending on the content. Nearly every head sets the decay to exactly zero for some tokens and to a positive value for others. For a few heads, this variation is substantial across tokens. Appendix C reports variance of the decay across all heads in the 250M model.

Figure 9 (Appendix C) illustrates this behavior on a passkey prompt for four selected early-layer heads: the decay is exactly zero on much of the repeated filler and switches to strong decay while processing the passkey.

Further, this query dependence is essential: replacing the query-dependent α_i with a query-independent scalar decay per head ("learnt `ALiBi`") hurts recall on long sequences (Appendix A.1).

6 Related Work

Rotary position encodings. The most widely adopted position encoding is `RoPE` (Su et al., 2024). `RoPE` rotates each query and key by an angle proportional to its absolute position, so that their dot product depends only on the relative distance between them. However, `RoPE` commonly overfits to relative distances seen during pretraining, resulting in poor length extrapolation (Press et al., 2022; Barbero et al., 2025; Lin et al., 2025; Yang et al., 2026).

Decay-based position encodings. Decay-based encodings (Press et al., 2022; Lin et al., 2025; Pai et al., 2026; Yang et al., 2025) alleviate extrapolation issues by down-weighting distant tokens, so that tokens beyond the training length are effectively ignored. However, since the decay accumulates over the tokens between the key and the query, distant tokens are eventually "forgotten" regardless of their relevance. Unlike these methods, which commit to a decay as tokens enter the context, `QDecay` allows each query to control the strength of the decay at read time.

This mechanism is closely related to forget gates in modern linear recurrences (Afzal, 2026). Modern recurrent and state-space language models compress history into a fixed-size state, motivating increasingly expressive mechanisms for deciding what information to retain, transform, route, or decay as each token is processed (Yang et al., 2024; Afzal et al., 2026; Lahoti et al., 2026). Decay-based encodings similarly determine a decay for each token at *write time*, when the token enters the key-value cache. Attention, however, retains the cache, so it need not forget tokens a priori. `QDecay` exploits this distinction: each query can change the decay applied to past tokens at *read time*. Thus, `QDecay` *has no equivalent linear recurrence* (Afzal, 2026), since a recurrence cannot undo forgetting that has already been applied to its state.

No positional encoding. No position encoding (`NoPE`) has been found to be an important tool for long-context recall (Yang et al., 2026). However, lacking recency bias, pure `NoPE` fails to model language well (Haviv et al., 2022) (see Table 1). `QDecay` interpolates between these behaviors: queries can retain recency bias if useful, while recovering `NoPE` when long-range recall is required.

Retrieval in language models. Lots of prior work has identified that a small subset of attention heads in a transformer that play a disproportionate role in retrieving information from long contexts

(Bick et al., 2025b; Wu et al., 2025; Zhang et al., 2025). Hybrid models, which combine recurrent or state-space layers with attention layers, provide a natural way to exploit this concentration (Glorioso et al., 2024; Lieber et al., 2024). In particular, Bick et al. (2025b; 2026) show that retrieval ability is concentrated in a small fraction of attention heads, and that retaining these heads while replacing the remainder with recurrent layers can preserve much of the Transformer's retrieval capability. This difficulty is also evident in work transferring Transformer capabilities into recurrent state-space models through distillation, where retrieval-related capabilities are among the hardest to recover (Bick et al., 2024; Wang et al., 2024; Bick et al., 2025a). QDecay addresses the same retrieval challenge within attention itself, allowing each query to remove positional decay when distant information is relevant.

7 Discussion & Future Work

Failure modes of existing encodings. ALiBi and FoX accumulate decay with distance, so the logit of an old key is driven toward $-\infty$ and the key is eventually forgotten. This forgetting is what allows them to extrapolate: their books3 perplexity degrades gracefully rather than diverging (Figure 4). However, the same forgetting is fatal for recall: RULER accuracy falls off beyond the pretraining length (Figure 5). Their parameterization compounds this problem: each γ_t is strictly positive and can only approach zero, so some decay always accumulates over a sufficiently long sequence and the model can never fully recover NoPE, which recall requires. Indeed, allowing the decay to reach exactly zero substantially improves FoX's recall (Appendix A.5).

Multiplicative decays fail differently. Wall applies an accumulated channel-wise decay $\boldsymbol{M}_{ij} = \mathrm{diag}\left(\bigodot_{t=j+1}^{i} \boldsymbol{\alpha}_t\right)$, with $\boldsymbol{\alpha}_t \in (0,1]^d$, so the logit $\boldsymbol{q}_i^\top \boldsymbol{M}_{ij} \boldsymbol{k}_j$ shrinks toward 0 rather than $-\infty$. Old keys are therefore not discarded, but instead converge toward similar nonzero attention weights. As a result, the correct key is drowned in a growing mass of stale keys. This breaks both recall and extrapolation, which is why Wall collapses on both far beyond the pretraining length (Figures 4 and 5). A similar argument may apply to PaTH, whose accumulated Householder factors are contractive rather than channel-wise decays. This contraction acts more slowly than additive decay, consistent with PaTH retaining performance longer before eventually collapsing (Figures 4 and 5).

Write-time versus read-time encodings. More generally, FoX, Wall, and PaTH are all *write-time* encodings: the position encoding or the transformation between key j and query i is determined by the intervening tokens $t \in (j, i]$. Past tokens thus determine how every future query sees the past, and no future query can undo the forgetting (additive) or contraction (multiplicative) that has already accumulated. QDecay instead provides *read-time* control over an additive encoding (ALiBi in this work): each query i can independently change how strongly the past is decayed (Section 3).

Future Work. Additive and multiplicative encodings have complementary strengths. Additive decay provides true forgetting, with logits driven to $-\infty$, which is what makes extrapolation safe (Section 7). Multiplicative encodings such as Wall and PaTH provide data-dependent, direction-wise transformations of the keys that a scalar additive bias cannot express. Combining a read-time additive decay with a multiplicative transformation, for example query-controlled decay on top of a PaTH-style key transformation, is a natural direction for obtaining the strengths of both families. Further, extending the read-time control of QDecay to multiplicative encodings, say PaTH (Yang et al., 2025) or Wall (Pai et al., 2026) is interesting future work.

8 Conclusion

We introduced QDecay, a lightweight query-dependent controller that can attach to any additive decay-based position encoding and allow each query set the intensity of the underlying encoding. Instantiated with ALiBi, across different tasks the resulting encoding QDecay-ALiBi achieves the best long-context recall and extrapolation at $8-16\times$ the pretraining length. QDecay-ALiBi matches or improves existing encodings at short context and outperforms the de-facto RoPE on all benchmarks. QDecay-ALiBi adds negligible parameters, is compatible with a fast FlashAttention-style algorithm and can readily serve as a drop-in replacement in attention-based models.

AI USE STATEMENT

We used LLMs for reviewing the text of the paper for grammar and clarity. All final text was written by the authors. The plotting and visualization code used to produce the figures was also generated by LLMs. We also used LLMs to help write some boilerplate code like data-loading, training and evaluation loops. Finally, we used LLMs to optimize our `QDecay` inference kernels. All LLM-generated code was reviewed by the authors.

REPRODUCIBILITY STATEMENT

`QDecay-ALiBi` can be implemented via PyTorch `FlexAttention` kernels (Dong et al., 2024) as a score modifier (`score_mod` function) (Section 3) and does not require a custom kernel; we release a basic FlexAttention-based implementation that can be used for both training and inference. We also release optimized inference kernels.

For `PaTH` we use the implementation in the `flash-linear-attention` (`fla`) library (Yang & Zhang, 2024), and for `Wall` we use the kernels released by Tilde Research[2] (Pai et al., 2026).

Training details are in Appendix E.1 and evaluation details in Appendix E.2.

We plan to release our training and evaluation code together with the pretrained model checkpoints.

REFERENCES

Arshia Afzal. On the legacy of linear transformers in positional embeddings. `https://arshiaafzal.github.io/blog/2026/pe/`, 2026. Blog post.

Arshia Afzal, Aviv Bick, Eric P. Xing, Volkan Cevher, and Albert Gu. Raven: High-recall sequence modeling with sparse memory routing. *arXiv preprint arXiv:2607.25357*, 2026.

Sandhini Agarwal, Lama Ahmad, Jason Ai, Sam Altman, Andy Applebaum, Edwin Arbus, Rahul K Arora, Yu Bai, Bowen Baker, Haiming Bao, et al. gpt-oss-120b & gpt-oss-20b model card. *arXiv preprint arXiv:2508.10925*, 2025.

Simran Arora, Sabri Eyuboglu, Aman Timalsina, Isys Johnson, Michael Poli, James Y Zou, Atri Rudra, and Christopher Ré. Zoology: Measuring and improving recall in efficient language models. In *International conference on learning representations*, volume 2024, pp. 15664–15730, 2024.

Federico Barbero, Alex Vitvitskyi, Christos Perivolaropoulos, Razvan Pascanu, and Petar Veličković. Round and round we go! what makes rotary positional encodings useful? In *International Conference on Learning Representations*, 2025.

Aviv Bick, Kevin Y. Li, Eric P. Xing, J. Zico Kolter, and Albert Gu. Transformers to ssms: Distilling quadratic knowledge to subquadratic models. In *Advances in Neural Information Processing Systems*, 2024.

Aviv Bick, Tobias Katsch, Nimit Sohoni, Arjun Desai, and Albert Gu. Llamba: Scaling distilled recurrent models for efficient language processing. *arXiv preprint arXiv:2502.14458*, 2025a.

Aviv Bick, Eric P. Xing, and Albert Gu. Understanding the skill gap in recurrent language models: The role of the gather-and-aggregate mechanism. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 4324–4344. PMLR, 2025b.

Aviv Bick, Eric P. Xing, and Albert Gu. Retrieval-aware distillation for transformer-ssm hybrids. In *International Conference on Machine Learning*, 2026.

Loïc Cabannes, Maximilian Beck, Maria Lomeli, Gergely Szilvasy, Matthijs Douze, Jade Copet, Pierre-Emmanuel Mazaré, Gabriel Synnaeve, and Hervé Jégou. Short window attention enables long-term memorization. In *International Conference on Learning Representations*, volume 2026, pp. 79759–79774, 2026.

[2] `https://github.com/tilde-research/wall-attention-release`

Cohere Team. Command A: An enterprise-ready large language model. *arXiv preprint arXiv:2504.00698*, 2025.

Tri Dao. Flashattention-2: Faster attention with better parallelism and work partitioning. In *International Conference on Learning Representations*, volume 2024, pp. 35549–35562, 2024.

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*, 2024.

Tri Dao, Daniel Haziza, Francisco Massa, and Grigory Sizov. Flash-Decoding for long-context inference. `https://princeton-nlp.github.io/flash-decoding/`, 2023. Blog post.

Juechu Dong, Boyuan Feng, Driss Guessous, Yanbo Liang, and Horace He. Flex attention: A programming model for generating optimized attention kernels. *arXiv preprint arXiv:2412.05496*, 2(3):4, 2024.

Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, et al. The pile: An 800gb dataset of diverse text for language modeling. *arXiv preprint arXiv:2101.00027*, 2020.

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. The language model evaluation harness, 07 2024. URL `https://zenodo.org/records/12608602`.

Paolo Glorioso, Quentin Anthony, Yury Tokpanov, James Whittington, Jonathan Pilault, Adam Ibrahim, and Beren Millidge. Zamba: A compact 7b ssm hybrid model. *arXiv preprint arXiv:2405.16712*, 2024.

Adi Haviv, Ori Ram, Ofir Press, Peter Izsak, and Omer Levy. Transformer language models without positional encodings still learn positional information. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pp. 1382–1390, 2022.

Cheng-Ping Hsieh, Simeng Sun, Samuel Kriman, Shantanu Acharya, Dima Rekesh, Fei Jia, Yang Zhang, and Boris Ginsburg. Ruler: What's the real context size of your long-context language models? *arXiv preprint arXiv:2404.06654*, 2024.

Pin-Lun Hsu, Yun Dai, Vignesh Kothapalli, Qingquan Song, Shao Tang, Siyu Zhu, Steven Shimizu, Shivam Sahni, Haowen Ning, and Yanning Chen. Liger kernel: Efficient triton kernels for LLM training. *arXiv preprint arXiv:2410.10989*, 2024.

Amirhossein Kazemnejad, Inkit Padhi, Karthikeyan Natesan Ramamurthy, Payel Das, and Siva Reddy. The impact of positional encoding on length generalization in transformers. *Advances in Neural Information Processing Systems*, 36:24892–24928, 2023.

Aakash Lahoti, Kevin Y. Li, Berlin Chen, Caitlin Wang, Aviv Bick, J. Zico Kolter, Tri Dao, and Albert Gu. Mamba-3: Improved sequence modeling using state space principles. In *International Conference on Learning Representations*, 2026.

Opher Lieber, Barak Lenz, Hofit Bata, Gal Cohen, Jhonathan Osin, Itay Dalmedigos, Erez Safahi, Shaked Meirom, Yonatan Belinkov, Shai Shalev-Shwartz, et al. Jamba: A hybrid transformer-mamba language model. *arXiv preprint arXiv:2403.19887*, 2024.

Zhixuan Lin, Evgenii Nikishin, Xu Owen He, and Aaron Courville. Forgetting transformer: Softmax attention with a forget gate. In *The Thirteenth International Conference on Learning Representations*, 2025.

Meta AI. The Llama 4 herd: The beginning of a new era of natively multimodal AI innovation. `https://ai.meta.com/blog/llama-4-multimodal-intelligence/`, April 2025.

Dhruv Pai, Timor Averbuch, Ashley Zhang, Ben Keigwin, and Alec Dewulf. Wall attention: Diagonal gates for softmax attention. https://blog.tilderesearch.com/blog/wall-attn, 2026. Blog post.

Guilherme Penedo, Hynek Kydlíček, Loubna Ben allal, Anton Lozhkov, Margaret Mitchell, Colin Raffel, Leandro Von Werra, and Thomas Wolf. The fineweb datasets: Decanting the web for the finest text data at scale. In *Advances in Neural Information Processing Systems*, volume 37, pp. 30811–30849, 2024.

Bowen Peng, Jeffrey Quesnelle, Honglu Fan, and Enrico Shippole. YaRN: Efficient context window extension of large language models. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=wHBfxhZu1u.

Ofir Press, Noah A. Smith, and Mike Lewis. Train short, test long: Attention with linear biases enables input length extrapolation. In *International Conference on Learning Representations*, 2022.

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

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.

Junxiong Wang, Daniele Paliotta, Avner May, Alexander M. Rush, and Tri Dao. The mamba in the llama: Distilling and accelerating hybrid models. In *Advances in Neural Information Processing Systems*, volume 37, 2024.

Wenhao Wu, Yizhong Wang, Guangxuan Xiao, Hao Peng, and Yao Fu. Retrieval head mechanistically explains long-context factuality. In *International Conference on Learning Representations*, 2025.

Bowen Yang, Bharat Venkitesh, Dwaraknath Gnaneshwar Talupuru, Hangyu Lin, David Cairuz, Phil Blunsom, and Acyr Locatelli. Rope to nope and back again: A new hybrid attention strategy. *Advances in Neural Information Processing Systems*, 38:64133–64157, 2026.

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

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*, 2024.

Songlin Yang, Yikang Shen, Kaiyue Wen, Shawn Tan, Mayank Mishra, Liliang Ren, Rameswar Panda, and Yoon Kim. PaTH attention: Position encoding via accumulating householder transformations. In *Advances in Neural Information Processing Systems*, 2025.

Wuwei Zhang, Fangcong Yin, Howard Yen, Danqi Chen, and Xi Ye. Query-focused retrieval heads improve long-context reasoning and re-ranking. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 23791–23805. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.emnlp-main.1214.

A Ablations

A.1 Ablating query dependence

To isolate the effect of query dependence, we compare `QDecay` with a query-independent variant that learns one fixed decay rate per attention head. This variant is equivalent to `ALiBi` with learned slopes, whereas `QDecay` predicts a separate decay rate for each query.

Table 3: Comparison of query-independent learned `ALiBi` slopes and query-dependent `QDecay`. Recall averages SWDE, FDA, and SQuAD, while RULER scores are averaged across tasks.

Model	LM ppl. ↓	Recall ↑	RULER 4K	RULER 8K	RULER 16K
Learned `ALiBi`	30.7	39.6	57.0	2.7	1.1
`QDecay`	29.1	42.8	56.1	49.4	33.6

`QDecay` improves aggregate language-modeling perplexity and short-context recall. The two models are comparable on RULER at 4K, but the query-independent variant collapses at longer contexts: its score falls to 2.7 at 8K and 1.1 at 16K, compared with 49.4 and 33.6 for `QDecay`.

A.2 QDecay initialization

We compare two initializations of the per-head decay scale e^A while holding the `ReLU` parameterization and all other training choices fixed. Our default initializes the scales geometrically from $1/2$ to $1/128$ across heads; the alternative initializes every head at $1/16$.

Table 4: Validation perplexity for different `QDecay` initializations.

Initialization of e^A	Val. PPL ↓
Geometric, $1/2$ to $1/128$	11.29
Uniform, $1/16$	11.39

The uniform initialization does not improve validation perplexity over the log-spaced initialization.

A.3 QDecay parameterization

We consider two ways to parameterize the query-dependent multiplier: $\delta_i = \mathrm{softplus}(\boldsymbol{w}^\top \boldsymbol{x}_i + c)$ and $\delta_i = \mathrm{ReLU}(\boldsymbol{w}^\top \boldsymbol{x}_i + c)$. `Softplus` is strictly positive for finite inputs, so a query can only approach `NoPE` asymptotically, whereas `ReLU` can output exactly zero and hence realize pure `NoPE` for an individual query.

Table 5: Comparison of the `Softplus` and `ReLU` parameterizations. Recall averages SWDE, FDA, and SQuAD, while RULER scores are averaged across tasks.

Model	LM ppl. ↓	LM eval. ↑	Recall ↑	RULER 4K	RULER 8K	RULER 16K
`QDecay` (`Softplus`)	28.2	43.0	43.7	48.0	35.2	15.3
`QDecay` (`ReLU`)	29.1	43.2	42.8	56.1	49.4	33.6

The two parameterizations perform essentially the same on the short-context recall tasks ($\sim$ 2K tokens) SWDE, FDA, and SQuAD. In contrast, `ReLU` improves average RULER accuracy by 8.1, 14.2, and 18.3 points at 4K, 8K, and 16K, respectively.

A.4 Preventing heads from reaching NoPE

`QDecay` recovers `NoPE` when its predicted decay α_i approaches zero. To test whether this limit matters, we impose a positive floor ϵ on the `softplus` output: $\delta_i = \epsilon + \mathrm{softplus}(\boldsymbol{w}^\top \boldsymbol{x}_i + c)$

and $\alpha_i = e^A \delta_i$. A positive floor prevents δ_i from approaching zero, making it harder for individual queries to recover NoPE. We test floor values $\epsilon \in \{0.01, 0.1, 1.0\}$.

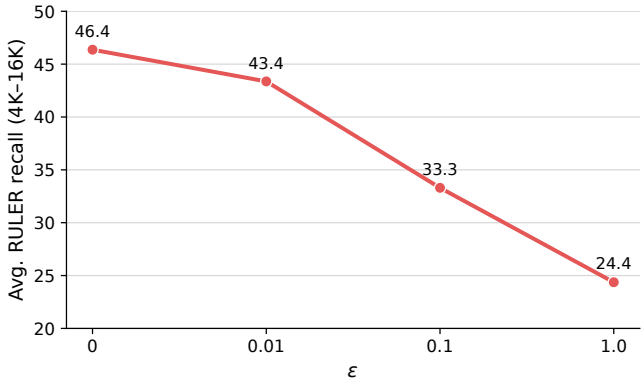


Figure 7: Average RULER recall as the minimum query-dependent multiplier increases. Each point averages accuracy across RULER tasks and the 4K, 8K, and 16K context lengths. The $\epsilon = 0$ point is the default ReLU QDecay model; positive-ϵ points use the floored softplus parameterization.

A.5 ReLU PARAMETERIZATION FOR FORGET GATES IMPROVES FoX

We compare the original softplus parameterization in FoX with a ReLU parameterization that can set the predicted decay exactly to zero. For a comparison shared by both runs, we average accuracy across the RULER S-NIAH-1, S-NIAH-2, and S-NIAH-3 tasks at each sequence length.

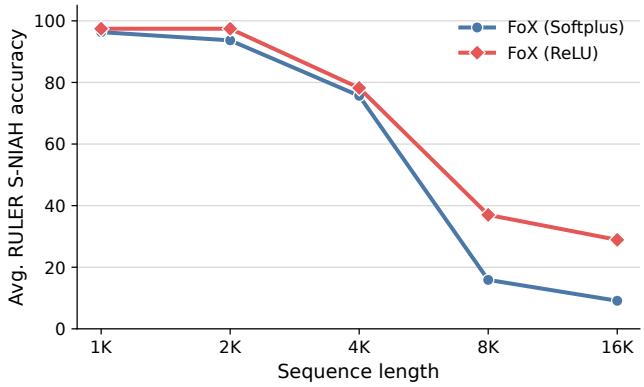


Figure 8: RULER S-NIAH accuracy for FoX with softplus and ReLU decay parameterizations. Each point averages S-NIAH-1, S-NIAH-2, and S-NIAH-3. Both models are pretrained at 4K context length; 8K and 16K test length extrapolation.

The two parameterizations are similar through the 4K pretraining length. Beyond it, ReLU improves average accuracy. Allowing exact zero decay therefore improves FoX's length extrapolation.

A.6 QDecay ON TOP OF FoX

The controller δ_i can be applied to any additive decay. Here we instantiate it on the accumulated FoX bias, $\widetilde{b}_{ij} = \delta_i \cdot b_{ij}^{\text{FoX}}$ with $b_{ij}^{\text{FoX}} = -\sum_{t=j+1}^{i} \gamma_t$ (Section 2), and call the result QDecay-FoX. The controller uses the same ReLU parameterization and the same geometric per-head initialization as QDecay-ALiBi; the FoX forget gate is initialized so that every increment γ_t equals one, so the two models coincide at initialization and differ only in whether the accumulated decay is data-dependent.

We train an 800M/30B model with the same setup as Table 1 and compare it with `ALiBi`, `FoX`, and `QDecay-ALiBi`.

Model	LM ppl. ↓	LM eval. ↑	Recall ↑
800M/30B			
`ALiBi`	16.4	50.8	**57.5**
`FoX`	16.2	51.1	51.3
`QDecay-ALiBi`	16.2	50.5	56.6
`QDecay-FoX`	**15.9**	**51.8**	57.4

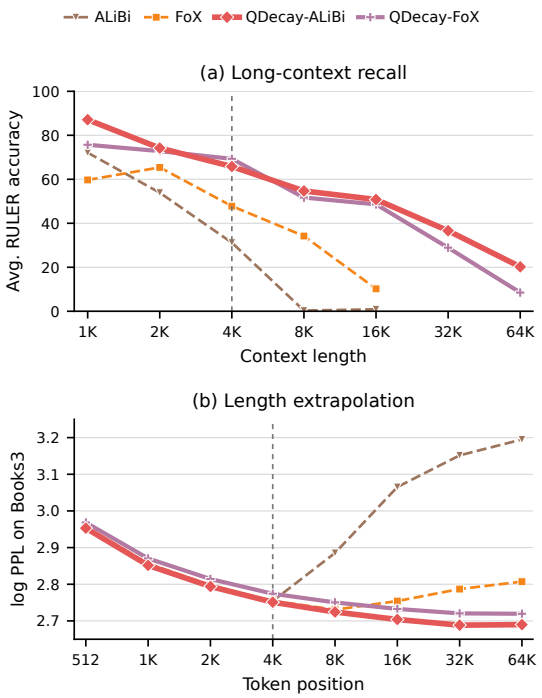


Table 6: **The `QDecay` controller on `ALiBi` versus on `FoX`.** Results at 800M/30B scale. **Left:** Language-modeling perplexity, evaluation accuracy, and recall, each averaged across the same tasks as Table 1. **Upper right:** Average RULER accuracy through 64K; `ALiBi` and `FoX` curves stop at 16K where longer evaluations are unavailable. **Lower right:** Books3 cumulative log perplexity through 64K. Dashed vertical lines mark the 4K pretraining context length.

B Complementarity with multiplicative encodings

Table 7 reports the full results for the 800M/30B hybrids of Section 4.6, which interleave a multiplicative encoding (`Wall` or `PaTH`) layer with a `QDecay` layer throughout the network.

Model	LM ppl. ↓	LM eval. ↑	Recall ↑
800M/30B			
`Wall`	17.2	49.4	55.6
`Wall` + `QDecay`	**15.4**	51.4	58.8
`PaTH`	16.6	50.6	56.3
`PaTH` + `QDecay`	15.9	**51.6**	**59.7**
`QDecay`	16.2	50.5	56.6

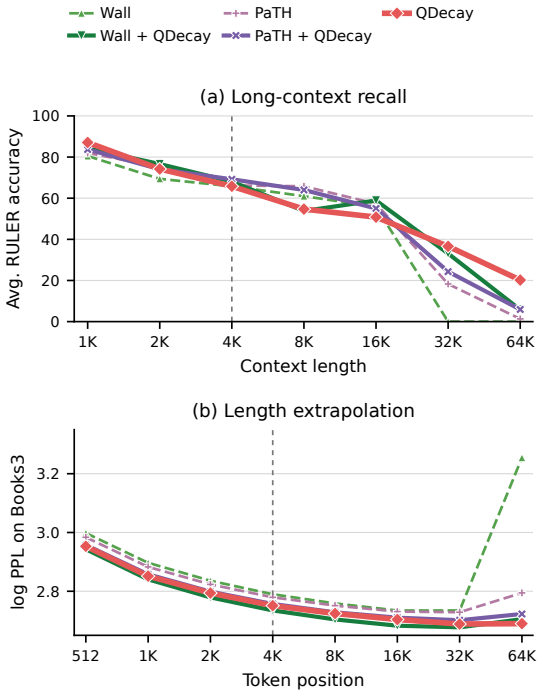


Table 7: **Additive `QDecay` improves multiplicative encodings on long-context recall and extrapolation.** Results at 800M/30B scale. **Left:** Language-modeling perplexity, evaluation accuracy, and recall, each averaged across the same tasks as Table 1. **Upper right:** Average RULER accuracy through 64K. **Lower right:** Books3 cumulative log perplexity through 64K. Dashed vertical lines mark the 4K pretraining context length. Beyond it, the `QDecay` hybrids degrade more gracefully than `Wall` or `PaTH` alone.

Adding `QDecay` to `Wall` as a hybrid improves perplexity and recall. Adding `QDecay` to `PaTH` as a hybrid improves the same metrics. The hybrids also remain stable on Books3 through 64K unlike `Wall` and `PaTH` alone.

C Learned QDecay decay rates

We measure the effective query-dependent decay $\alpha_i = e^{A_h}\,\mathrm{ReLU}(\boldsymbol{w}^\top \boldsymbol{x}_i + c)$ for every layer and attention head across tokens. For each model, we evaluate across 4K length five RULER S-NIAH-2 examples and five SWDE examples. This yields ∼24K tokens. We then average α_i over these tokens separately for every layer–head pair.

Figure 11 shows the resulting means for the 800M model. An average decay below 0.001 across tokens is effectively `NoPE`. For visual clarity, we therefore snap such layer–head means to zero in both plots. As in the 250M model (Figure 6), the 800M learned decays span several orders of magnitude and include near-`NoPE` heads throughout the network, including the **first layer.**

Effect of initialization on the learned pattern. Within each layer, the ordering of heads from strong decay to near-`NoPE` in Figures 6 and 11 reflects the Mamba-2-style geometric initialization of the per-head slopes γ (Section 3): heads initialized with smaller slopes are the ones that settle closest to `NoPE`. The within-layer structure of the heatmaps is therefore inherited from the initialization rather than discovered from scratch, whereas the *across-layer* pattern, in particular the emergence of `NoPE` in the first layer, is not prescribed by the initialization, which is identical for every layer. Section A.2 compares this geometric initialization against a uniform one.

Figure 10 reports the token-level variance for the 250M model over the same pooled queries. As discussed in Section 5, 287 of the 288 heads have nonzero variance, showing that the learned decay

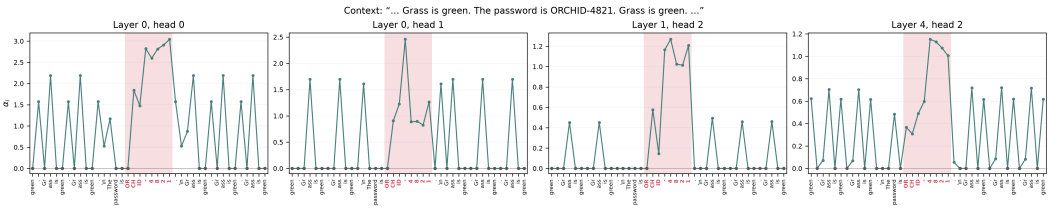


Figure 9: **QDecay dynamically adapts its decay to the content.** Per-token α_i around the passkey (shaded tokens `OR CH ID - 4 8 2 1`) for four selected heads in the early layers, with 16 tokens of context on each side. The decay is exactly zero on much of the repeated filler and switches to strong decay while processing the passkey.

changes across tokens even within a head. Figure 9 shows this token-level behavior directly on a passkey prompt for four selected early-layer heads.

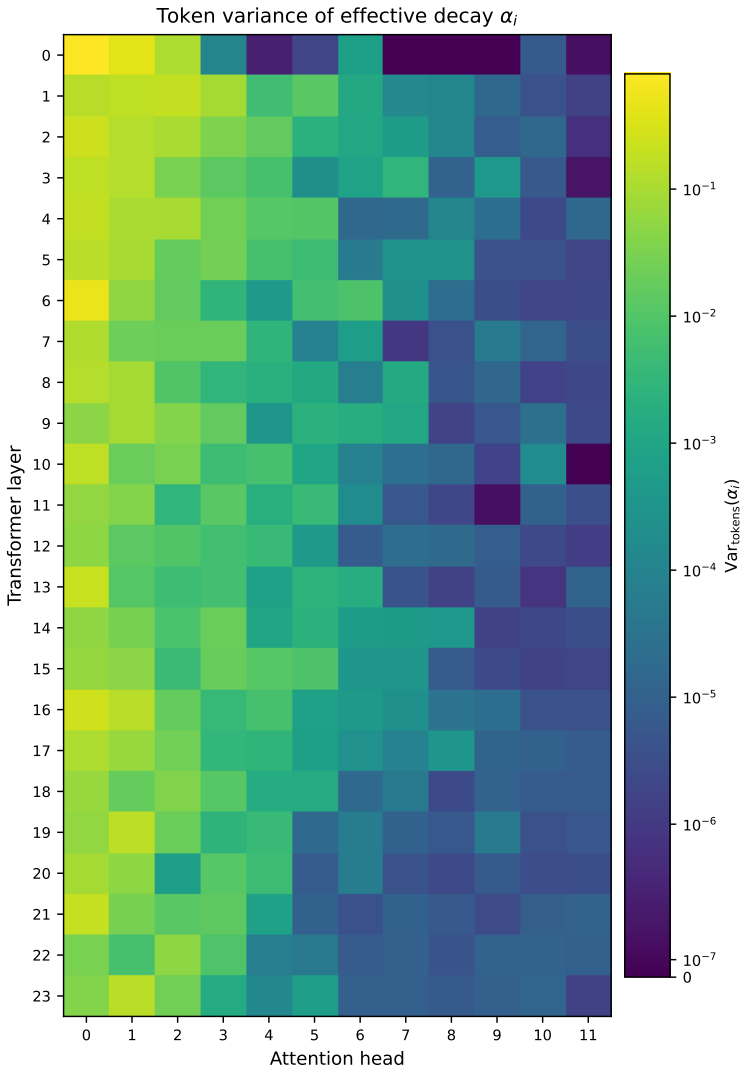


Figure 10: Token-level variance of the effective decay α_i for every layer and attention head in the 250M `QDecay` model. Each cell reports the variance over ∼24K tokens.

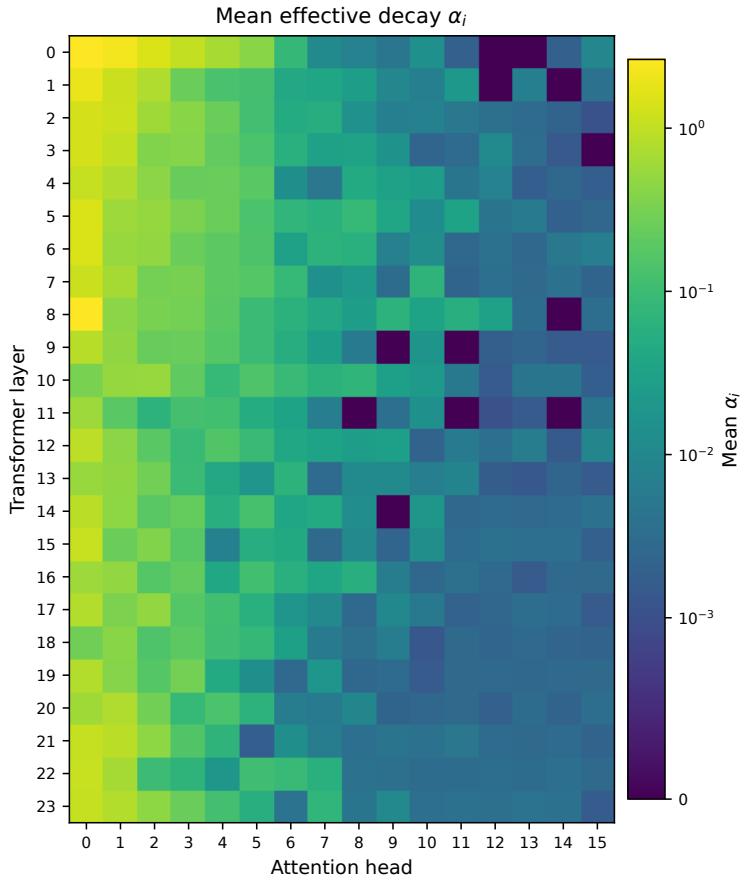


Figure 11: Mean effective decay α_i for every layer and attention head in the 800M `QDecay` model with the Mamba-2-style geometric initialization. Each cell averages over the pooled query tokens described above; means below 0.001 are snapped to zero for display.

D Performance

D.1 Training

Training speed In Table 8, we show wall-clock training time for all our methods at the 800M scale. All methods are trained for the same number of gradient steps (57,600). We use official implementations of each method wherever possible; we implement `ALiBi` via FlexAttention. For training, we used our implementation of `QDecay` via FlexAttention as well, for simplicity and ease of experimentation. **This implementation is attached in the supplementary material.**

We found that `PaTH` and `Wall` impose significant additional training memory overhead, exceeding the GPU capacity at this configuration, so for these methods we reduce the microbatch size and use gradient accumulation.

Our implementation of `QDecay` imposes a modest training time ($\sim 34\%$) and GPU memory ($\sim 9\%$) overhead compared to the highly optimized implementation of attention with `RoPE` from FlashAttention. However, this difference is mostly due to the overhead of FlexAttention itself compared to FlashAttention. Similar to Section D.2, we explored implementing faster `QDecay` training kernels based on FlashAttention; these kernels are within 10% total (forward & backward) speed of FlashAttention with `RoPE` (and therefore the end-to-end training slowdown is even less than 10%). We plan to release these kernels after the deadline.

Model	Layer implementation	Micro-batch	Peak GPU mem. (GB)	Wall-clock training time	
				Runtime (hr)	Rel. to RoPE
RoPE	FlashAttention	8	63.3	16.8	1.00
FoX	fla forgetting-attention	8	64.4	20.5	1.22
ALiBi	FlexAttention	8	69.1	20.6	1.23
QDecay	FlexAttention	8	69.2	22.5	1.34
PaTH	fla PaTH	4	44.8	28.1	1.68
Wall	Tilde Triton	4	44.7	44.2	2.64

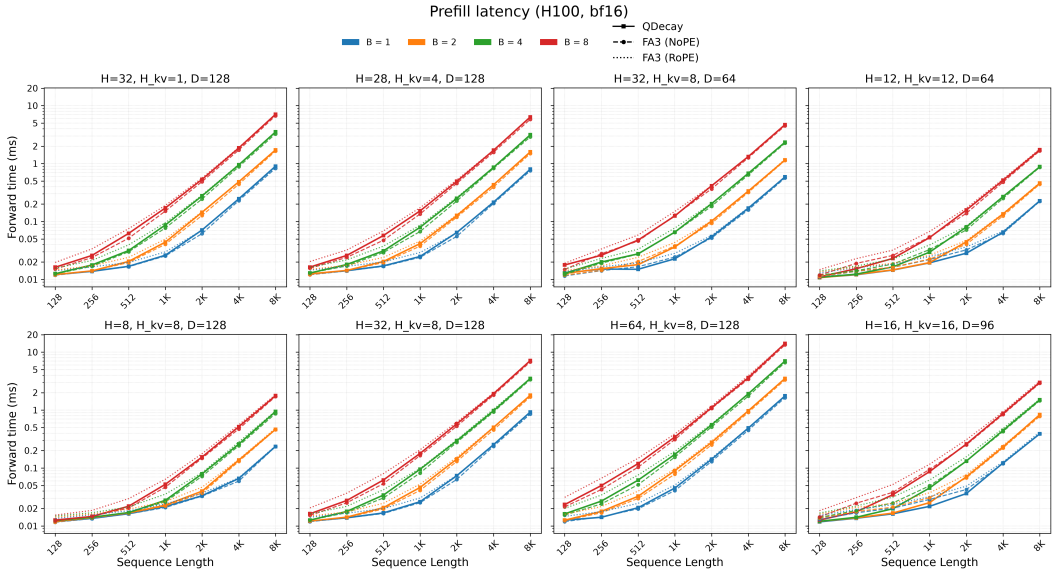
Table 8: End-to-end training time at 800M scale. Measures the total wall-clock time taken for training to completion (not evaluation). All models are trained on identical infrastructure, on 1 node with 8 NVIDIA H100 GPUs.

D.2 INFERENCE

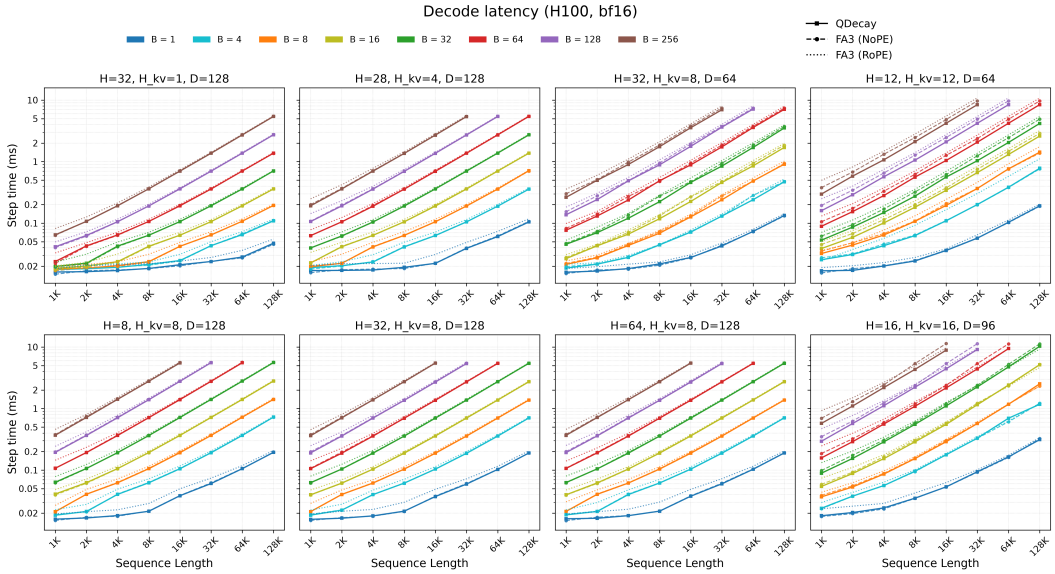
The implementation is attached in the supplementary material.

Inference speed We develop fast inference-time kernels for both (batched) prefill and decode for QDecay (QDecay-ALiBi), based on a fork of FlashAttention3 (FA3) (Dao, 2024). We implement our modifications as a simple patch on top of the FlashAttention3 code, as most of the logic is shared with standard FA3. We compare to native FA3 with no positional encoding ("FA3-NoPE"), and to FA3 with RoPE ("FA3-RoPE"). We benchmark our inference kernels on NVIDIA H100 GPUs using bf16 precision (though they can be easily extended to other datatypes and architectures, by similar modifications to the corresponding FA3 kernels). Results are in Figure 12; QDecay does not impose meaningful additional overhead in any setting. (In fact, we are able to obtain slightly better performance than even FA3-NoPE in certain settings. This is not an inherent property of QDecay; rather, this is due to tiling optimizations we implemented that are also compatible with FA3 but not in the stock FA3 implementation.) The code for our efficient kernels is in the supplementary material. For prefill, FA3-RoPE applies RoPE as a separate kernel using the Liger RoPE kernel (Hsu et al., 2024). For decode, FA3-RoPE uses FA3's fused decode path (which fuses the RoPE computation with the main attention computations). We benchmarked different standard open-source approaches for FA3 with RoPE and found each of these to be the fastest for their setting.

Inference Memory QDecay does not modify the KV-cache, as it is an additive encoding. Thus, the retained memory of QDecay during inference is the same as that of NoPE, aside from the memory for the controller parameters themselves (less than 0.1% of total parameter memory at our model sizes). (Compared to NoPE, RoPE also has to retain the cos/sin tables in memory, which is a fixed small additional memory overhead.) The peak memory increase compared to NoPE is less than 1% (QDecay requires a $B \times H \times T$ additional temporary fp32 tensor for prefill and $B \times H$ for decode, which is negligible compared to the other $O(BHTD)$-size tensors involved in the main attention computation).



(a) Prefill



(b) Decode

Figure 12: (a) Prefill and (b) decode timings for Flash-Attention 3 with `NoPE` (i.e., without any positional encoding), and with `RoPE`; and for `QDecay`. Each panel is one head configuration: H is the number of query heads, H_{kv} is the number of key/value heads ($H > H_{kv}$ in the GQA/MQA setting), and D is the head dimension. Axes are log-log. Prefill is causal. Decode involves one new query per sequence against a contiguous KV cache of the given length, including the timing for the write of the new token's key/value into the KV cache.

E Detailed Experimental Results

E.1 Training details

Models and data. We pretrain decoder-only Transformers on FineWeb-Edu (Penedo et al., 2024), using the Llama-2 tokenizer and a context length of 4K tokens. At small scale, we train 250M-scale models for 10B tokens with 24 layers, 12 attention heads, and hidden dimension 768

(24L12H768D). At medium scale, we train 800M-scale models for 30B tokens with 24 layers, 16 attention heads, and hidden dimension 1536 (24L16H1536D), following Afzal et al. (2026). The medium-scale PaTH model instead uses 12 attention heads because the fla kernel requires head size 128.

Optimization. All models use AdamW with $(\beta_1, \beta_2) = (0.9, 0.95)$, weight decay 0.1, and gradient clipping at 1.0. Each global batch contains 128 sequences, or approximately 0.5M tokens. We warm up for 2,000 steps (approximately 1B tokens) and then use cosine learning-rate decay. At small scale, the learning rate decays from 8×10^{-4} to 8×10^{-5}; at medium scale, it decays from 4×10^{-4} to 4×10^{-5}. We use training seed 42. Within each scale, we hold the data, backbone, token budget, and optimization fixed across methods. Exact parameter counts and training runs are in Tables 9 and 10.

E.2 Evaluation details

After pretraining, we evaluate zero-shot on standard short-context language-modeling benchmarks and on real-world recall tasks ($\sim$2K tokens) using lm-eval-harness (Gao et al., 2024).

LM ppl. averages perplexity on the FineWeb-Edu validation set, LAMBADA, and WikiText.

LM eval. averages zero-shot accuracy on LAMBADA, PIQA, HellaSwag, WinoGrande, ARC-Easy, and ARC-Challenge.

Recall averages SWDE, FDA, and SQuAD.

For long-context language-modeling evaluation, we use Books3 (Gao et al., 2020), obtained from https://huggingface.co/datasets/Geralt-Targaryen/books3.

Finally, we average RULER accuracy (Hsieh et al., 2024) across seven retrieval and state-tracking tasks: S-NIAH-1–3, MK1, MK2, MQ, and MV. Each task is evaluated on 300 samples at context lengths 1K–16K and on 100 samples at 32K and 64K, where the quadratic cost of attention makes evaluation substantially slower. Evaluations at 8K–64K test extrapolation beyond the 4K pretraining length.

E.3 Model configurations and runs

Table 9: Small-scale model sizes and training runs. Parameter counts are in millions.

Model	Parameters
RoPE	219.1
ALiBi	219.1
FoX	219.3
Wall	233.2
PaTH	220.5
NoPE	219.1
QDecay	219.3

Table 10: Medium-scale model sizes and training runs. Parameter counts are in millions.

Model	Parameters
RoPE	777.9
ALiBi	777.9
FoX	778.4
Wall	834.5
PaTH	780.8
QDecay	778.4

E.4 PER-DATASET LANGUAGE MODELING AND RECALL

Table 11: Small-scale language modeling results. PIQA, HellaSwag, ARC-e, and ARC-c use normalized accuracy; LAMBADA and WinoGrande use raw accuracy.

	Perplexity ↓				LM evaluation ↑						
Model	Val.	LMB.	Wiki.	Avg.	LMB.	PIQA	Hella.	Wino.	ARC-e	ARC-c	Avg.
RoPE	12.0	65.3	32.7	36.7	28.5	63.8	34.4	52.3	48.5	24.7	42.0
ALiBi	11.4	45.8	30.3	29.2	31.8	64.4	36.4	50.6	48.4	25.7	42.9
FoX	11.3	51.7	29.4	30.8	30.4	64.9	36.4	50.4	48.6	27.5	43.0
Wall	12.0	57.6	31.2	33.6	30.3	63.0	34.6	51.0	48.7	25.3	42.2
PaTH	11.6	47.0	30.3	29.6	32.1	63.8	35.5	50.7	46.2	25.7	42.3
NoPE	13.8	124.6	40.0	59.5	22.7	61.9	30.7	50.4	43.9	25.3	39.2
QDecay	11.3	46.6	29.3	29.1	31.2	64.1	36.9	51.8	49.4	26.0	43.2

Table 12: Small-scale recall results.

Model	SWDE	FDA	SQuAD	Avg.
RoPE	44.3	22.4	31.7	32.8
ALiBi	43.7	44.9	32.9	40.5
FoX	37.5	34.4	33.0	35.0
Wall	48.1	50.1	30.4	42.9
PaTH	53.5	41.7	31.5	42.2
NoPE	18.9	23.8	24.7	22.5
QDecay	50.7	45.0	32.8	42.8

Table 13: Medium-scale language modeling results. PIQA, HellaSwag, ARC-e, and ARC-c use normalized accuracy; LAMBADA and WinoGrande use raw accuracy.

	Perplexity ↓				LM evaluation ↑						
Model	Val.	LMB.	Wiki.	Avg.	LMB.	PIQA	Hella.	Wino.	ARC-e	ARC-c	Avg.
RoPE	9.0	23.7	21.4	18.0	38.7	69.7	47.1	53.4	57.5	30.7	49.5
ALiBi	8.7	20.3	20.1	16.4	40.0	69.2	49.1	53.3	60.4	32.9	50.8
FoX	8.7	19.9	19.9	16.2	39.9	70.6	49.1	53.6	62.1	31.2	51.1
Wall	8.9	21.5	21.2	17.2	40.0	68.8	47.2	53.5	56.7	30.5	49.4
PaTH	8.7	20.4	20.6	16.6	40.8	68.3	48.3	53.4	60.5	32.0	50.6
QDecay	8.6	20.2	19.7	16.2	40.7	69.5	49.5	53.4	59.3	30.7	50.5

Table 14: Medium-scale recall results.

Model	SWDE	FDA	SQuAD	Avg.
RoPE	60.7	60.8	40.9	54.1
ALiBi	64.3	65.7	42.4	57.5
FoX	56.8	56.2	41.0	51.3
Wall	62.5	63.5	40.7	55.6
PaTH	61.4	66.1	41.4	56.3
QDecay	63.7	64.4	41.8	56.6

E.5 PER-TASK RULER RESULTS

Table 15: Small-scale RULER S-NIAH accuracy across context lengths. The 8K and 16K columns evaluate extrapolation beyond the 4K pretraining context length. A dash denotes an unavailable evaluation.

	S-NIAH-1					S-NIAH-2					S-NIAH-3				
Model	1K	2K	4K	8K	16K	1K	2K	4K	8K	16K	1K	2K	4K	8K	16K
RoPE	100.0	100.0	50.0	0.0	0.0	88.0	100.0	60.0	0.0	0.0	87.0	71.0	3.0	0.0	0.0
ALiBi	100.0	100.0	95.7	1.7	0.7	100.0	100.0	100.0	0.3	0.0	96.3	96.3	78.3	0.0	0.0
FoX	100.0	100.0	91.0	33.0	17.7	100.0	99.0	98.0	12.3	2.7	89.0	82.0	38.0	2.3	7.0
Wall	99.3	99.0	99.7	94.3	80.0	100.0	100.0	99.7	91.3	49.0	96.7	98.0	83.3	36.0	6.0
PaTH	100.0	100.0	97.3	90.0	63.3	100.0	100.0	99.7	93.7	12.0	89.0	85.7	75.7	19.7	2.7
NoPE	0.3	30.0	3.7	0.0	0.0	2.3	95.3	93.7	0.0	0.0	0.0	60.3	8.7	0.0	0.0
QDecay	100.0	100.0	100.0	100.0	99.3	100.0	100.0	100.0	99.0	61.3	97.7	97.3	95.3	53.3	9.3

Table 16: Small-scale RULER accuracy on the non-S-NIAH tasks across context lengths.

	MK1					MK2					MQ					MV				
Model	1K	2K	4K	8K	16K	1K	2K	4K	8K	16K	1K	2K	4K	8K	16K	1K	2K	4K	8K	16K
RoPE	32.0	47.0	32.0	0.0	0.0	5.0	0.0	0.0	0.0	0.0	23.2	23.8	18.2	0.0	0.0	12.5	25.2	19.2	0.0	0.0
ALiBi	52.7	57.0	60.0	1.0	0.0	11.3	3.3	2.7	0.0	0.0	28.0	27.0	26.9	0.1	0.0	45.0	26.1	26.4	0.4	0.0
FoX	33.0	33.0	36.0	11.0	4.3	6.0	1.0	0.0	0.3	0.0	26.0	25.0	25.0	9.7	3.5	30.5	27.2	23.5	7.5	1.8
Wall	35.0	38.3	31.7	33.3	24.7	3.7	0.3	0.0	0.0	0.0	31.9	26.3	25.3	26.3	22.6	38.9	26.9	25.1	26.7	23.8
PaTH	46.7	34.3	37.7	35.7	14.7	7.7	1.3	0.7	0.0	0.3	50.9	29.4	27.4	24.4	5.8	56.8	26.6	25.5	25.3	7.8
NoPE	24.0	31.0	30.3	0.0	0.0	0.0	1.0	0.3	0.0	0.0	20.8	19.8	16.2	0.0	0.0	21.8	26.7	24.8	0.0	0.0
QDecay	55.7	47.3	44.3	44.0	32.3	11.3	5.0	3.0	0.7	0.3	27.1	26.2	25.3	24.8	16.6	33.7	24.8	24.7	24.2	15.9

Table 17: Medium-scale RULER S-NIAH accuracy across context lengths. The 8K and 16K columns evaluate extrapolation beyond the 4K pretraining context length.

	S-NIAH-1					S-NIAH-2					S-NIAH-3				
Model	1K	2K	4K	8K	16K	1K	2K	4K	8K	16K	1K	2K	4K	8K	16K
RoPE	100.0	100.0	84.0	0.0	0.0	89.0	64.3	89.3	0.0	0.0	13.0	24.3	48.3	0.0	0.0
ALiBi	100.0	99.7	94.0	0.0	0.0	66.7	50.3	9.3	0.7	1.0	51.0	9.0	0.0	0.0	0.0
FoX	100.0	100.0	93.3	24.7	21.7	46.7	81.3	10.0	90.0	19.3	12.7	56.3	69.3	39.3	6.0
Wall	99.7	100.0	99.7	99.0	99.3	100.0	100.0	100.0	99.3	95.3	97.3	98.3	95.7	91.0	68.0
PaTH	100.0	100.0	100.0	100.0	99.7	100.0	100.0	100.0	100.0	98.0	77.0	94.3	75.7	63.3	46.3
QDecay	100.0	100.0	100.0	78.7	94.3	100.0	100.0	100.0	100.0	100.0	96.0	98.0	93.0	58.0	38.7

Table 18: Medium-scale RULER accuracy on the non-S-NIAH tasks across context lengths.

	MK1					MK2					MQ					MV				
Model	1K	2K	4K	8K	16K	1K	2K	4K	8K	16K	1K	2K	4K	8K	16K	1K	2K	4K	8K	16K
RoPE	74.0	57.7	50.7	0.0	0.0	41.7	28.0	8.3	0.0	0.0	75.3	64.9	37.6	0.0	0.0	94.8	60.3	34.7	0.0	0.0
ALiBi	76.0	54.7	5.7	1.3	2.7	19.7	10.3	3.7	0.0	0.3	95.0	76.3	50.3	0.0	0.4	96.4	76.8	55.3	0.0	1.8
FoX	57.7	58.0	57.3	40.0	13.7	15.7	3.7	2.3	0.7	0.7	89.7	74.4	53.8	20.8	4.5	95.3	84.3	48.7	24.1	5.5
Wall	61.3	53.7	52.0	43.3	43.3	16.7	8.3	6.0	2.7	3.0	94.3	64.7	53.5	47.9	41.0	94.4	60.9	53.6	43.7	40.3
PaTH	71.3	63.3	61.0	68.7	64.0	30.3	37.7	42.0	36.7	24.7	96.8	70.2	50.0	49.8	36.5	98.2	60.8	35.1	42.0	32.2
QDecay	67.0	71.3	63.0	60.7	56.3	52.3	40.7	30.7	9.7	4.3	96.3	54.2	37.8	38.0	24.3	98.0	55.1	36.4	37.8	37.8