MULTI-STATE LINEAR ATTENTION WITH GENERALIZED SECOND-ORDER ROUTING

Anonymous authors
Paper under double-blind review

ABSTRACT

Self-attention plays a key role in long-sequence modeling, but its quadratic time complexity fundamentally limits the scaling of Large Language Models. Recently, Multi-State Linear Attention (MSLA) mechanisms have been proposed to make sequence modeling more efficient by splitting the context into segments and compressing each in its own memory state. MSLA aggregates the information extracted from each memory state using only the mean alignment between a query and each segment's keys, ignoring how the alignment is distributed within the segment. In order to model the relevance of a segment to a query more comprehensively and incorporate the variance of the alignment in the aggregation, we propose MERLIN (**M**om**E**nt **R**outing for multi-state **LIN**ear attention), a method that tracks the first and second moments of each segment's keys. Using the moments of keys, a query can recover both the mean and the variance of its alignment with each segment to perform a more content-aware aggregation of per-state readouts. Experiments on multi-state variants of Gated DeltaNet-1.3B and Mamba-2-780M show that MERLIN consistently outperforms existing aggregation methods on various long-context tasks. In particular, MERLIN outperforms baselines on RULER needle-in-a-haystack and long-context QA (+8.3% for Gated DeltaNet-1.3B and +7.0% for Mamba-2-780M) and diverse subtasks in LongBench (+13.5% and +12.0%). It even outperforms baselines that consume 4x more memory. MERLIN provides a simple and general routing mechanism for improving long-context capabilities of MSLA models. Code will be available upon acceptance.

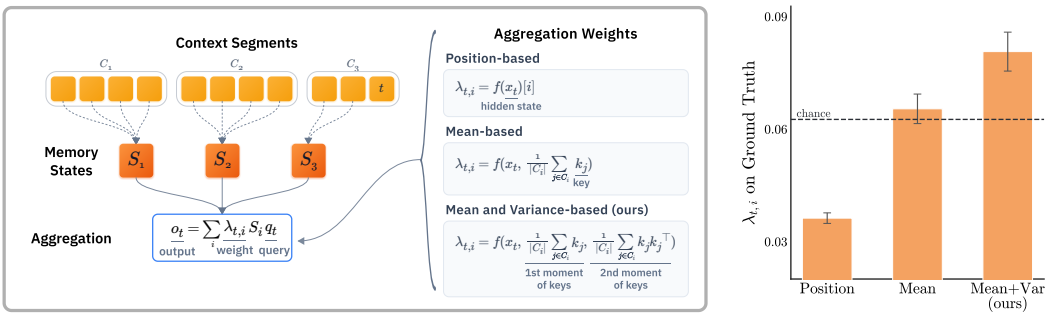
1 INTRODUCTION

The self-attention mechanism is a core operation in long-sequence modeling (Vaswani et al., 2017). However, its quadratic computational complexity and linearly growing key-value (KV) cache have become a key bottleneck in scaling compute and data (Keles et al., 2023). Linear attention (Katharopoulos et al., 2020; Sun et al., 2023; Peng et al., 2023) addresses these limitations by reformulating attention to operate in sub-quadratic complexity. For example, DeltaNet (Yang et al., 2024) and Gated DeltaNet (Yang et al., 2025) remove the softmax in standard attention and replace the KV cache with a single fixed-size memory state. However, a single fixed-size memory state is inherently limited in memory capacity and may lead to information loss when compressing longer contexts (Arora et al., 2024b; Jelassi et al., 2024; Kuratov et al., 2024). As a result, models based on single-state linear attention still underperform standard attention in practice.

Recently, Multi-State Linear Attention (MSLA) mechanisms (Behrouz et al., 2026; Guo et al., 2026; Wang et al., 2026) have been introduced to improve the long-context capabilities of single-state linear attention. MSLA partitions the context into segments and compresses each segment into its own memory state. Each token extracts segment-level outputs from preceding memory states and then aggregates them to produce its final output representation (Fig. 1a). Operating in the multi-state regime therefore introduces two novel challenges: how to divide the context into segments (***separation***), and how to combine the segment-level outputs (***aggregation***).

We study *aggregation*, which is relatively underexplored compared to *separation*. Existing methods aggregate segment-level outputs via a weighted sum (Behrouz et al., 2026; Guo et al., 2026; Wang et al., 2026), where the weights represent each segment's relevance to the query: closely aligned segments contribute more to the final output than others. However, prior works derive the weights either

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.



(a) Multi-State Linear Attention (MSLA) (b) Aggregation Weight on GT

Figure 1: **(a)** A multi-state linear attention (MSLA) model aggregates segment-level outputs via a weighted sum. Prior methods compute the weights $\lambda_{t,i}$ from position or from the mean of each segment's keys. MERLIN instead reads both the first and second moments of the keys. **(b)** On retrieval-augmented generation samples in HELMET (Yen et al., 2025), we measure the aggregation weight each method places on the ground truth (GT) segment at the query token. Position-based aggregation largely misses the GT and mean-based aggregation recovers more, while MERLIN places the most weight on it.

from position (Guo et al., 2026; Wang et al., 2026), assuming recency as the source of relevance, or from the mean (first-order) alignment between each segment's keys and the query (Behrouz et al., 2026). They overlook the variance (second-order) of the alignment, which captures sharply aligned keys that the mean alone cannot distinguish from a segment of uniformly weak ones.

To analyze why the variance matters, we examine how an MSLA model assigns aggregation weights on a long-sequence reasoning task. Fig. 1b shows that the segment containing the ground truth, which is the segment most relevant to the response, receives the highest aggregation weight when the weights are derived from both the mean and variance of the alignment, compared to the mean or position alone. This motivates an aggregation method that incorporates the variance of the alignment in deriving the weights.

We propose MERLIN (**M**om**E**nt **R**outing for multi-state **LIN**ear attention), a method that leverages the mean and variance of key-query alignment to improve aggregation in multi-state linear attention. MERLIN stores the first and second moments of each segment's keys. Using the moments, a query recovers both the mean (first-order) and variance (second-order) of its alignment with each segment's keys to derive the weights for the weighted sum of segment-level outputs. Our experiments show that MERLIN improves the long-context capabilities of multi-state linear attention models.

Our contributions are:

1. We identify segment-level output aggregation as a key bottleneck in multi-state linear attention: existing methods weight each segment's output using only the mean alignment between the query and the segment's keys (§2).
2. We propose MERLIN, a method that keeps the first and second moments of each segment's keys. A query uses these moments to recover the mean and variance of its alignment with the segment (§3.1) and to derive the aggregation weights (§3.2).
3. We evaluate multi-state variants of Gated DeltaNet-1.3B and Mamba-2-780M (§4.1). On LongBench (Bai et al., 2024), MERLIN improves the average score over the strongest baseline by 13.5% on Gated DeltaNet and 12.0% on Mamba-2 (§4.2).

2 BACKGROUND

Given an input sequence of T tokens, the query, key, and value matrices $\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V} \in \mathbb{R}^{T \times d}$ are projected from the tokens' hidden states $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_T$. Standard full attention computes:

$$\textbf{Attention (Vaswani et al., 2017):} \quad \boldsymbol{O} = \mathrm{softmax}(\boldsymbol{Q}\boldsymbol{K}^\top + \boldsymbol{M})\boldsymbol{V},$$

where $\boldsymbol{M} \in \{-\infty, 0\}^{T \times T}$ is a causal masking matrix. Softmax attention costs quadratic time, and requires a KV cache that grows linearly with the context. This motivates more efficient sub-quadratic attention formulations.

Table 1: **Computational and Memory Complexity.** *Memory* is the size of the state(s) retained to decode each token. d_r is the dimension of the moments of keys (§3.1). Memory Caching (Behrouz et al., 2026) and MERLIN assume uniform segmentation ($L = T/m$) where m is the number of tokens per segment.

Mechanism	Compute (Prefill)	Memory
Full Attention (Vaswani et al., 2017)	$O(T^2 d)$	$O(Td)$
Linear Attention (Katharopoulos et al., 2020)	$O(Td^2)$	$O(d^2)$
Multi-State Linear Attention		
Log-Linear Attention (Guo et al., 2026)	$O(Td^2 \log T)$	$O(d^2 \log T)$
Memory Caching (Behrouz et al., 2026)	$O(TLd^2)$	$O(Ld^2)$
MERLIN (ours)	$O(TL(d^2 + d_r^2))$	$O(L(d^2 + d_r^2))$

Linear Attention. Linear attention removes the softmax normalization and instead computes:

$$\textbf{Linear Attention:} \quad \boldsymbol{O} = (\boldsymbol{Q}\boldsymbol{K}^\top \odot \boldsymbol{M})\boldsymbol{V}, \quad \boldsymbol{M}_{i,j} = 1 \text{ if } i \geq j, \text{ and } 0 \text{ otherwise.}$$

This formulation can be reorganized into a recurrence over a memory state $\boldsymbol{S}_t \in \mathbb{R}^{d \times d}$:

$$\text{Katharopoulos et al. (2020):} \quad \boldsymbol{S}_t = \boldsymbol{S}_{t-1} + \boldsymbol{v}_t \boldsymbol{k}_t^\top = \sum_{t'} \boldsymbol{v}_{t'} \boldsymbol{k}_{t'}^\top, \quad \boldsymbol{o}_t = \boldsymbol{S}_t \boldsymbol{q}_t, \tag{1}$$

where $\boldsymbol{k}_t, \boldsymbol{v}_t, \boldsymbol{q}_t$ are the key, value, and query vectors of a token at position t. The memory state $\boldsymbol{S}_t$ accumulates key-value outer products, acting as an associative memory that binds each key to its value. A query reads from the memory state via a matrix-vector product to get the output representation $\boldsymbol{o}_t$. The recurrence now runs in linear time and constant memory. Recent works add gating and delta-rule correction to improve the memory capacity of the fixed-size state. For example, Gated DeltaNet (Yang et al., 2025) (GDN) uses the recurrence as follows:

$$\textbf{Gated DeltaNet (Yang et al., 2025):} \quad \boldsymbol{S}_t = \boldsymbol{S}_{t-1}\big(\alpha_t(\boldsymbol{I} - \beta_t \boldsymbol{k}_t \boldsymbol{k}_t^\top)\big) + \beta_t \boldsymbol{v}_t \boldsymbol{k}_t^\top, \tag{2}$$

where $\alpha_t \in (0, 1)$ decays the state and β_t sets the strength of the update.

Multi-State Linear Attention. Linear attention can be limiting when T grows because it needs to compress the entire context into a single fixed-size state. Multi-state linear attention relaxes this bottleneck by partitioning the sequence into L contiguous segments $C_1, \ldots, C_L$ and compressing each segment in its own state:

$$\textbf{Multi-State Linear Attention (Separation):} \quad \boldsymbol{S}_{C_i} = \sum_{j \in C_i} \boldsymbol{v}_j \boldsymbol{k}_j^\top. \tag{3}$$

Intuitively, Eq. (3) can be viewed as splitting the first equation of Eq. (1) into L parts, which we call *separation*. At step t, a query reads segment-level outputs from the last $\ell \leq L$ memory states visible at step t. The intermediate outputs are then aggregated via a weighted sum to derive the final output representation:

$$\textbf{Multi-State Linear Attention (Aggregation):} \quad \boldsymbol{o}_{t,i} = \boldsymbol{S}_{C_i} \boldsymbol{q}_t, \quad \boldsymbol{o}_t = \sum_{i=1}^{\ell} \lambda_{t,i}\, \boldsymbol{o}_{t,i}, \tag{4}$$

where $\lambda_{t,i}$ is the aggregation weight that determines how much the output from segment i contributes to the final output at position t. Eq. (4) can be viewed as the second equation of Eq. (1) applied to ℓ partitioned states, which we refer to as aggregation.

Unlike single-state linear attention, in multi-state linear attention, the number of memory states increases as the content length increases. Multi-state linear attention therefore interpolates between single-state linear attention (constant memory, linear time) and full attention (linear memory, quadratic time) (Behrouz et al., 2026). Tab. 1 summarizes the time and memory complexities of attention variants.

Aggregation in Multi-State Linear Attention. The aggregation of segment-level outputs in Eq. (4) requires weights $\lambda_{t,i}$. These weights function as routers that increase the contribution of outputs from query-relevant segments and decrease the contribution of outputs from others. The simplest methods learn a position-based weight vector $\boldsymbol{\lambda}_t$ (with entries $\lambda_{t,i}$) directly from the target token, projected from its query or hidden state (Guo et al., 2026; Wang et al., 2026):

$$\text{Guo et al. (2026):} \quad \boldsymbol{\lambda}_t = \boldsymbol{W}_\lambda \boldsymbol{x}_t \quad \text{or} \quad \text{Wang et al. (2026):} \quad \boldsymbol{\lambda}_t = \boldsymbol{W}_\lambda \boldsymbol{q}_t. \tag{5}$$

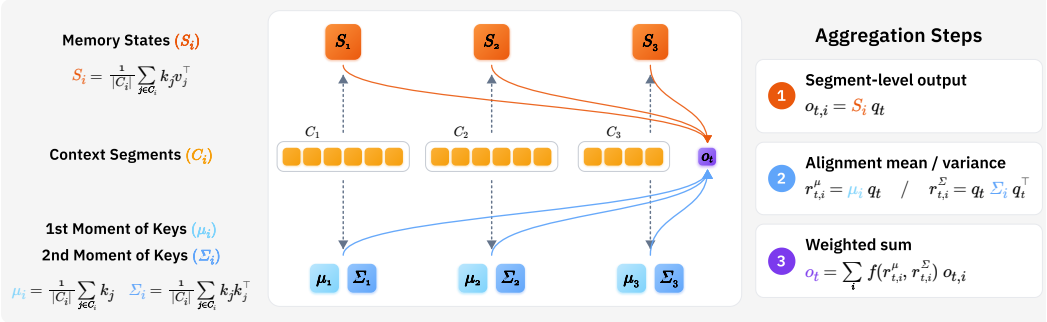


Figure 2: **MERLIN Overview.** The context is split into segments C_i (middle row). Each segment stores a memory state (top row) and the first and second moments of its keys (bottom row). At position t, the query $\boldsymbol{q}_t$ reads segment-level outputs from memory states (step 1). The query computes the mean and variance of the key-query alignment from the moments (step 2). The final output is a weighted sum of segment-level outputs, where the weights are derived from a gated mix of the alignment mean and variance (step 3).

These weights do not consider what each segment stores and therefore cannot function as a content-aware router. Behrouz et al. (2026) introduces content-awareness in aggregation weights as follows:

$$\text{Behrouz et al. (2026)}: \quad \lambda_{t,i} = \langle \bar{\boldsymbol{k}}_i, \boldsymbol{u}_t \rangle, \qquad \boldsymbol{u}_t = \boldsymbol{W}_u \boldsymbol{x}_t, \qquad \bar{\boldsymbol{k}}_i = \frac{1}{|C_i|}\sum_{j\in C_i} \boldsymbol{k}_j, \tag{6}$$

where $\boldsymbol{u}_t$ is a learned vector projected from hidden state $\boldsymbol{x}_t$ and $\bar{\boldsymbol{k}}_i$ is the first moment of keys in segment i (*i.e.*, the mean). However, this only captures the *mean* alignment between the query and a segment's keys and does not capture the *variance*.

We revisit the analysis in Fig. 1b and demonstrate that the variance of the alignment is necessary for more content-aware aggregation of segment-level readouts. We train multi-state models with different aggregation methods and evaluate on 30 randomly selected TriviaQA samples in HELMET (Yen et al., 2025) at 4K context length. This is a retrieval-augmented generation task where each sample consists of multiple passages and a prompt that requires retrieving information from one of them. Thus, the model should locate the relevant passage and extract the correct answer to the prompt.

We find that different aggregation methods assign varying amounts of aggregation weight to the segment containing the ground truth. Ideally, the weights assigned to the ground truth segment should be higher than others in order to increase the contribution of the information extracted from the segment in the final response. Fig. 1b shows that mean- and variance-based aggregation assigns the highest weight to the ground truth segment compared to position-only and mean-only aggregation. This motivates our proposed method MERLIN, which leverages both orders of alignment to compute the weights for segment-level output aggregation.

3 MERLIN

Our goal is to capture the variance of the alignment between a query and each segment's keys to derive more informative aggregation weights. We first introduce how the first and second moments of the keys are constructed (§3.1). Then, we discuss how the mean and variance of the alignment can be derived from the moments (§3.2). Finally, we elaborate on the relationship between multi-state linear attention equipped with MERLIN and softmax attention (§3.3).

3.1 FIRST AND SECOND MOMENTS OF KEYS

For each token, we project new keys and queries $\boldsymbol{k}'_t, \boldsymbol{q}'_t \in \mathbb{R}^{d_r}$ from the hidden states. We use $\boldsymbol{k}'_t, \boldsymbol{q}'_t$ instead of $\boldsymbol{k}_t, \boldsymbol{q}_t$ to construct and read from the first and second moments of keys, primarily to keep moments compact by choosing d_r that is smaller than d.

The first moment of keys is the mean-pooled vector of all keys ($\boldsymbol{k}'_j$) in the segment. The second moment is the average of the outer products $\boldsymbol{k}'_j \boldsymbol{k}'^{\top}_j$ over the segment. The first and second moments of keys can thus be formulated as:

$$\boldsymbol{\mu_i} = \frac{1}{|C_i|} \sum_{j \in C_i} \boldsymbol{k}'_j, \quad \boldsymbol{\Sigma_i} = \frac{1}{|C_i|} \sum_{j \in C_i} \boldsymbol{k}'_j \boldsymbol{k}'^{\top}_j. \tag{7}$$

The first and second moments are fixed-size, and their dimensions depend only on d_r. Together, they summarize segment i, where $\boldsymbol{\mu_i}$ captures the average direction of the keys and $\boldsymbol{\Sigma_i}$ captures how the keys are spread within the segment.

3.2 Aggregation Weight Derivation

Given a token at position t, we compute the following scalar scores for each visible segment C_i (the segments leading up to the current one):

$$r^{\mu}_{t,i} = \boldsymbol{\mu_i}^{\top} \boldsymbol{q}'_t, \quad r^{\Sigma}_{t,i} = \boldsymbol{q}'^{\top}_t \boldsymbol{\Sigma_i} \boldsymbol{q}'_t - (r^{\mu}_{t,i})^2. \tag{8}$$

Here, $r^{\mu}_{t,i}$ captures the mean key-query alignment, measuring how well the segment's keys broadly align with the query. Meanwhile, $r^{\Sigma}_{t,i}$ captures the variance, measuring the spread of the per-token alignment between the query and the keys in the segment. We mix the softmax-normalized $r^{\mu}_{t,i}$ and $r^{\Sigma}_{t,i}$ with a learned gate to get the final weight $\lambda_{t,i}$:

$$\textbf{MERLIN (ours):} \quad \lambda_{t,i} = \gamma_t \mathrm{softmax}_i(\tau_\mu \, r^{\mu}_{t,i}) + (1 - \gamma_t)\mathrm{softmax}_i(\tau_\Sigma \, r^{\Sigma}_{t,i}), \tag{9}$$

where $\mathrm{softmax}_i$ denotes the softmax operation over the segment index i, γ_t is a scalar gate, and τ_μ and τ_Σ are learnable temperatures that scale the two scores before normalization. The gate γ_t is computed from the token's hidden state $\boldsymbol{x}_t$ through a learned projection. The temperatures τ_μ, τ_Σ calibrate the sharpness of each term. This shows how the first and second moments are used to derive the weights for Eq. (4).

3.3 Connection to Softmax Attention

Suppose we ran token-level softmax attention over the keys and then grouped its mass by segment. For a target token at position t and a segment C_i, the key-query similarity can be expressed as:

$$s_{t,i,j} = \boldsymbol{q}'^{\top}_t \boldsymbol{k}'_j, \qquad j \in C_i.$$

We can define the length-normalized evidence of the segment as:

$$Z_{t,i}(\delta) = \frac{1}{|C_i|} \sum_{j \in C_i} e^{\delta \, s_{t,i,j}},$$

where δ is a temperature hyperparameter. Grouping the token-level softmax by segment, the total attention mass a full softmax layer would place on segment i can be expressed as:

$$A_{t,i} = \frac{\sum_{j \in C_i} \exp(\delta s_{t,i,j})}{\sum_{\ell} \sum_{j \in C_\ell} \exp(\delta s_{t,\ell,j})} = \mathrm{softmax}_i \left(\log |C_i| + \log Z_{t,i}(\delta) \right). \tag{10}$$

Because our segments are equal-sized, $\log |C_i|$ is the same for every segment and cancels inside the softmax. Expanding the remaining log-evidence in cumulants around $\delta = 0$, we obtain:

$$\log Z_{t,i}(\delta) = \delta \, \eta_{t,i} + \tfrac{\delta^2}{2} \, \sigma^2_{t,i} + O(\delta^3), \quad \eta_{t,i} = \frac{1}{m} \sum_{j \in C_i} s_{t,i,j}, \quad \sigma^2_{t,i} = \frac{1}{m} \sum_{j \in C_i} s^2_{t,i,j} - \eta^2_{t,i} \tag{11}$$

Therefore, the relevance of a segment is governed by the first two terms: the mean $\eta_{t,i}$ and the variance $\sigma^2_{t,i}$. These are precisely the scores $r^{\mu}_{t,i}$ and $r^{\Sigma}_{t,i}$, which are computed from the first and second moments via matrix-vector products:

$$\eta_{t,i} = \boldsymbol{\mu_i}^{\top} \boldsymbol{q}'_t = r^{\mu}_{t,i}, \qquad \sigma^2_{t,i} = \boldsymbol{q}'^{\top}_t \boldsymbol{\Sigma_i} \, \boldsymbol{q}'_t - \left(\boldsymbol{\mu_i}^{\top} \boldsymbol{q}'_t \right)^2 = r^{\Sigma}_{t,i}. \tag{12}$$

Rather than the fixed additive coefficients of Eq. (11) (δ on the mean, $\frac{\delta^2}{2}$ on the variance) inside a single softmax, MERLIN combines the two scores with a learned gated mixture of separately-normalized softmax terms (Eq. (9)), with learnable temperatures τ_μ, τ_Σ and gate γ_t. The gated mixture is also flexible to implement, since the two terms are normalized and scaled independently.

Table 2: **Performance on Long-Context Retrieval Tasks.** We evaluate mid-trained (**MidTr**) models with different aggregation methods on the NIAH and QA tasks in RULER at 16K context length, **–** is the base model without mid-training. Highest scores in each column are in **bold** and second-highest are underlined.

	MidTr	S-NIAH-1	S-NIAH-2	S-NIAH-3	MK-NIAH-1	MQ-NIAH	MV-NIAH	HotpotQA	SQuADQA	Avg
Gated DeltaNet 1.3B	–	18.6	25.2	4.2	16.8	5.1	15.8	10.0	8.3	13.0
	Vanilla	89.0	99.2	10.8	23.4	80.8	70.2	15.0	8.3	49.6
	Position	97.2	96.8	16.0	42.2	82.5	78.4	16.0	7.3	54.6
	Meanpool	**99.6**	99.4	21.2	**48.4**	86.6	85.0	12.0	10.7	57.9
	MERLIN	99.2	**100.0**	**47.0**	45.0	**91.0**	**88.8**	**17.0**	**13.7**	**62.7**
Mamba-2 780M	–	0.0	0.0	0.0	0.0	0.0	0.0	2.0	0.0	0.3
	Vanilla	**100**	**100**	8.4	24.6	92.2	86.4	16	4.7	54.0
	Position	**100.0**	**100.0**	54.2	91.0	94.5	89.2	20.0	8.0	69.6
	Meanpool	**100.0**	**100.0**	66.2	89.4	93.7	91.8	20.0	8.1	71.2
	MERLIN	99.8	99.8	**96.0**	**94.8**	**95.9**	**92.2**	**22.0**	**9.4**	**76.2**

4 EXPERIMENTS

We conduct experiments to quantify the effect of MERLIN on the long-context capabilities of multi-state linear attention models. We first explain the models, datasets, and baselines used for training and evaluation (§4.1). Then, we show the main evaluation results on the core benchmarks (§4.2). Third, we conduct analysis on positional biases and impact of segment length (§4.3). Finally, we conduct ablations on the core design choices in MERLIN (§4.4).

4.1 SETUP

Models. We use the Gated DeltaNet (Yang et al., 2025) and Mamba-2 (Dao & Gu, 2024) architectures for the following experiments. Gated DeltaNet is a linear attention model that updates the memory state following the recurrence in Eq. (2). Mamba-2 is a state-space model that uses recurrence of structured semiseparable matrices. We show that MERLIN improves upon the multi-state variants of both architectures with distinct features to demonstrate its robustness and effectiveness. A more detailed pseudocode for each model can be found in Section C.5.

Training. We first pre-train the Gated DeltaNet-1.3B (Yang et al., 2025) model on 50B tokens in FineWeb-edu (Penedo et al., 2024) following the official training configurations. For Mamba-2-780M, we use the officially released checkpoint[1] pre-trained on 300B tokens from Pile (Gao et al., 2020). These pre-trained models are still single-state: they require mid-training (or continued pre-training) to function as multi-state variants, *i.e.*, training the pre-trained model with the same objective function as pre-training on the mid-training dataset.

In order to convert single-state models to multi-state variants, we introduce separation (Eq. (3)) and aggregation (Eq. (4)) mechanisms during mid-training. We fix the separation mechanism to uniform segmentation, where each segment has the same length of 256 tokens. At training context length of 16K tokens, the model learns to aggregate as many as 64 (16K / 256) segment-level outputs. We specify the full list of training hyperparameters and model configurations in Section C.1.

We mid-train the pre-trained models in two stages. Each stage is defined by the type of training data that was used. Stage 1 uses 2B tokens from Long-Data-Collections (LDC) (TogetherAI, 2024), with 10% of the training samples (0.2B tokens) augmented with RULER (Hsieh et al., 2024) needle-in-a-haystack (NIAH)-like tasks. The augmented data contains artificial keys and values scattered throughout regular LDC sequences. The task query and ground truth answer are attached at the end of the context. This dataset is designed to train basic language modeling capability as a multi-state variant by activating its aggregation mechanism.

In stage 2, we train on a smaller dataset of 60M tokens. It consists of 50% regular LDC sequences, and 50% synthetically generated samples from RULER HotpotQA and SQuADQA. This dataset is designed to further strengthen the model's aggregation mechanism under reasoning contexts. After stage 1 and stage 2, the pre-trained single-state models now function as multi-state models with long-context capabilities. The detailed mid-training hyperparameters can be found in Section C.2.

[1] huggingface state-spaces/mamba2-780m

Table 3: **Performance on Long-Context Tasks in LongBench.** We evaluate on eleven LongBench tasks, filtered for samples with at most 16K tokens.

		Single-doc QA			Multi-doc QA		Summarization			Few-shot Learning			
	MidTr	NQ	QR	MF	HP	2W	GR	QM	MN	SS	TC	TQ	Avg
Gated DeltaNet 1.3B	-	2.4	5.0	13.9	4.5	7.9	9.1	16.1	9.2	33.0	22.3	19.5	13.0
	Vanilla	6.0	6.1	12.2	6.8	8.5	**9.9**	**16.5**	9.4	**40.5**	42.5	21.4	16.3
	Position	11.5	9.6	14.9	14.3	11.1	6.1	14.0	12.3	39.0	48.5	**23.1**	18.6
	Meanpool	8.8	15.2	18.0	14.6	9.8	8.7	15.3	10.1	34.5	54.8	22.4	19.3
	MERLIN	**13.9**	**19.4**	**20.3**	**16.3**	**14.4**	7.8	15.9	**12.8**	40.0	**60.4**	20.1	**21.9**
Mamba-2 780M	-	0.9	1.6	3.6	0.9	2.2	3.2	8.7	12.6	8.0	8.1	1.4	4.7
	Vanilla	1.0	3.2	7.6	4.2	5.2	10.2	12.3	14.7	36.5	54.5	5.7	14.1
	Position	1.2	5.1	10.5	5.7	6.8	15.5	15.0	17.4	42.0	65.1	12.5	17.9
	Meanpool	1.7	4.2	11.5	6.0	5.6	16.0	**17.7**	18.5	44.5	61.1	**15.1**	18.4
	MERLIN	**1.9**	**6.7**	**12.6**	**7.8**	**9.1**	**20.4**	16.1	**19.2**	**46.0**	**72.7**	14.2	**20.6**

Benchmarks. We evaluate the models on various long-context benchmarks. First, we evaluate on the NIAH and QA tasks in RULER (Hsieh et al., 2024) at 16K context length. The single needle variants (S-NIAH-1/2/3) evaluate the model's ability to retrieve a single key-value pair in different types of natural and synthetic text haystacks. The multi-needle variants (MK-NIAH-1/MQ-NIAH/MV-NIAH) evaluate the model's ability to retrieve from multiple key-value pairs in different formats. The HotpotQA and SQuADQA tasks in RULER evaluate multi-hop and single-hop reasoning skills.

We also evaluate on the following subtasks in LongBench (Bai et al., 2024): NarrativeQA (NQ), Qasper (QR), MultiFieldQA-en (MF), HotpotQA (HP), 2WikiMultihopQA (2W), GovReport (GR), QMSum (QM), MultiNews (MN), SAMSum (SS), TREC (TC), TriviaQA (TQ). We filter for samples under 16K context length. These tasks largely evaluate single-document QA, multi-document QA, summarization, and few-shot learning capabilities.

Finally, we evaluate on the TriviaQA retrieval-augmented-generation task in HELMET (Yen et al., 2025) at 16K context length. We use this benchmark to analyze how the position of the ground truth in the sequence impacts the model's performance. Section C.4 provides a detailed explanation of the evaluation protocol used in our experiments.

Baselines. We compare MERLIN with three other baselines: **Vanilla**, **Position**, and **Meanpool**. **Vanilla** is the unmodified single-state backbone, with neither segmentation nor aggregation mechanisms. The other two are alternative aggregation methods for multi-state variants. **Position** (Eq. (5)) derives aggregation weights from the hidden states alone, scaling segment-level outputs by a learned recency scalar (Guo et al., 2026). **Meanpool** (Eq. (6)) derives aggregation weights from first-order alignment between a query and each segment's keys (Behrouz et al., 2026).

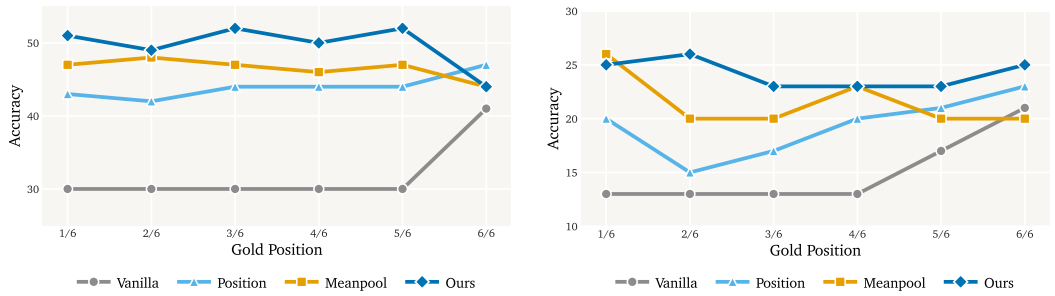
4.2 MAIN RESULTS

We report results on RULER (Hsieh et al., 2024) NIAH and QA tasks. Tab. 2 shows that MERLIN outperforms baselines in most tasks for both GDN and Mamba-2 architectures. MERLIN applied to GDN leads to an average score of 62.7, while Meanpool has an average score of 57.9, followed by Position's 54.6. In the case of Mamba-2, MERLIN shows a 7.0% relative improvement over Meanpool (76.2 vs. 71.2), and a 9.5% relative improvement over Position (vs. 69.6). This suggests that mid-training with MERLIN improves long-context retrieval.

We also report results on LongBench (Bai et al., 2024) tasks filtered to 16K. Overall, MERLIN demonstrates strongest average performance, followed by Meanpool, Position, and finally Vanilla. Tab. 3 shows that MERLIN leads to robust gains particularly on single-doc and multi-doc QA. For GDN, MERLIN leads to a 13.5% improvement in average score compared to Meanpool (21.9 vs 19.3). For Mamba-2, MERLIN leads to a 12.0% (20.6 vs 18.4) improvement compared to Meanpool. This suggests that MERLIN strengthens general long-context capabilities.

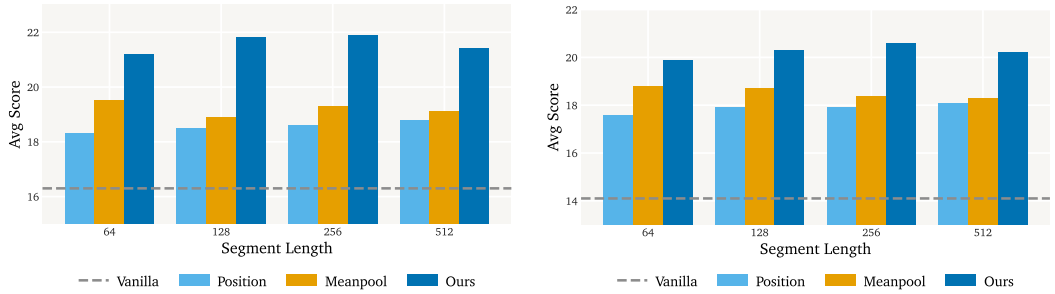
4.3 ANALYSIS

Positional Bias Analysis. We evaluate models on the TriviaQA task in HELMET (Yen et al., 2025), at 16K context length. This is a retrieval-augmented generation task, where the context



(a) Gated DeltaNet-1.3B (b) Mamba-2-780M

Figure 3: **Position-Controlled Retrieval.** Retrieval accuracy as the gold passage sweeps from the start $(1/6)$ to the end $(6/6)$ of the context. Vanilla is recency-biased: it recovers the gold only when placed near the end and is otherwise flat, while Position and mean-based aggregation are flatter but weaker. MERLIN stays high and roughly uniform across positions.



(a) Gated DeltaNet-1.3B (b) Mamba-2-780M

Figure 4: **MERLIN Outperforms Baselines Across Segment Length.** Average score as the segment size that defines each state varies over $\{64, 128, 256, 512\}$. MERLIN leads at every segment length, and its margin over mean-based aggregation persists as segments grow. Vanilla (dashed) is segment-independent.

consists of multiple passages and only one of them, which we refer to as the gold passage, is related to the final query of the prompt. The location of the gold passage is shuffled randomly within the context for each sample. Therefore, we can track the accuracy of the model's responses depending on the location of the gold passage, and detect any positional biases that might exist. Fig. 3 shows that the Vanilla baseline demonstrates a strong recency bias: it struggles unless the gold passage is located towards the end of the context. The Position baseline also demonstrates a weak recency bias, where the accuracy gradually increases as the gold passage moves closer to the end. Both the Meanpool baseline and MERLIN perform more evenly across gold passage positions, while MERLIN is better overall. This suggests that MERLIN enhances the model's ability to extract information from important segments in the context by improving aggregation of segment-level readouts.

Segment Length Analysis. We examine whether MERLIN is robust to different segment lengths. Segment length, or the number of tokens per segment, determines the number of segments per context given that the entire length of the sequence is fixed ($L = T/m$). The shorter the segment length, the more states per sequence, and therefore the higher the memory consumption of the model. We experiment with $m \in \{64, 128, 256, 512\}$ for each aggregation method. At $m = 64$, the model consumes approximately 8 times more memory compared to $m = 512$. Fig. 4 shows that for both GDN and Mamba-2, MERLIN outperforms baselines on LongBench (Bai et al., 2024) tasks across m. In particular, MERLIN at $m = 256$ (default) outperforms Meanpool and Position baselines at $m = 64$, which consume 4 times more memory. This suggests that MERLIN is more memory efficient compared to its baselines.

Memory Cost and Parameter Overhead. In Section D, we compare the memory cost of different multi-state linear attention strategies with MERLIN. We observe that the extra cost of our additional second moment features is limited to at most 12.6% for Gated DeltaNet and 4.2% for Mamba-2 at $d_r = 128$, while the memory cost decreases quadratically with d_r. We also compare the number of parameters that MERLIN adds to each backbone compared to baselines. We find that the MERLIN's parameter overhead is on par with the Meanpool baseline at 4.6% for GDN and 2.2% for Mamba-2.

Qualitative Examples. In Appendix §D, we provide examples of aggregation weight ($\lambda_{t,i}$) assignment on four HELMET (Yen et al., 2025) retrieval-augmented generation samples using MERLIN and baseline methods. As observed in the quantitative evaluations, MERLIN assigns a higher weight to the gold segment, and this supports that the variance of the alignment is critical for aggregation.

4.4 ABLATIONS

Ablations on Dimension of Moments of Keys. We report on the impact of d_r, the dimension of the first and second moments of keys (§3.2), on the performance of MERLIN. We vary $d_r \in \{16, 32, 64, 128\}$ during training and evaluate on LongBench (Bai et al., 2024). Fig. 5 shows that the average score on LongBench tasks gradually increases as d_r increases. This suggests that the width of the moments of keys is critical in deriving informative aggregation weights. However, there is a trade-off in deciding d_r. The moments of keys can be more expressive when d_r is larger. However, MERLIN consumes less memory compared to the Meanpool and Position baselines as d_r gets smaller. More details on the memory consumption can be found in Tab. 9 of Section D.

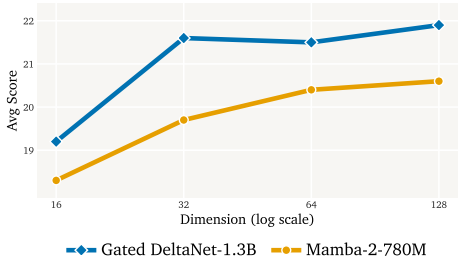


Figure 5: **Dimension of Moments.** Average LongBench scores as d_r varies over $\{16, 32, 64, 128\}$ (log axis). Performance saturates around $d_r = 32$ for GDN and $d_r = 64$ for Mamba-2.

Ablations on Parameter Sharing. For each token, we project new keys and queries (k'_t, q'_t) to construct the first and second moments, separate from the original keys and queries for the memory state (k_t, q_t). We isolate the impact of decoupling the keys and queries by experimenting with a variant of MERLIN where $k'_t = k_t$ and $q'_t = q_t$ (shared-KQ rows in Tab. 4). We further experiment with a variant of MERLIN where we control the decay and update strength of the first and second moments of keys per token, similar to the decay and write-strength features of the memory state explained in Eq. (2) (write-forget rows in Tab. 4). We find that both ablations substantially degrade the performance of MERLIN on LongBench (Bai et al., 2024). This suggests that using a dedicated set of keys and queries from moments of keys is particularly important in deriving informative aggregation weights.

	Single	Multi	Summ	Few-shot	Overall
GDN + MERLIN	**17.9**	**15.4**	**12.2**	**40.2**	**21.9**
+ write-forget	14.9	9.2	11.6	35.3	18.5
+ shared-KQ	10.7	9.2	10.9	29.8	15.7
Mamba + MERLIN	**7.1**	**8.5**	**18.6**	**44.3**	**20.6**
+ write-forget	6.7	6.9	16.6	43.8	19.6
+ shared-KQ	5.1	6.1	15.1	38.3	17.1

Table 4: **Parameter Sharing.** We train two MERLIN variants: sharing the keys and queries between the memory state and moments of keys (+shared KQ) and using write-forget gates on moments of keys (+write-forget). We evaluate on LongBench and report per-category average scores. The categories are identical to Tab. 3.

5 CONCLUSION

We investigate the aggregation problem in multi-state linear attention, where segment-level outputs are combined via a weighted sum to produce the final output representation of each token. We propose MERLIN, an aggregation method that assigns weights to each segment based on the mean and variance of the alignment between a query and the keys of each segment. In order to compute the mean and variance of the alignment, MERLIN keeps compact first and second moments of the segment's keys. We find that MERLIN is superior to existing aggregation methods in terms of performance and memory efficiency. MERLIN presents a generalizable and effective routing mechanism for the aggregation problem in multi-state linear attention models.

AI USE STATEMENT

In this work, we used generative AI tools to polish the writing, generate synthetic training datasets (§4.1), and write triton kernels for method implementation. We also used AI tools to plot figures and create a name for our method. We have not used generative AI tools to formulate and develop the conceptual framework of the method, or to identify relevant literature. We have reviewed all AI-assisted work. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

Our work presents a method that improves general long-context capabilities of multi-state linear attention models. Although we do not see a clear ethical concern, we advise readers to be cautious with the datasets and tools they may use to extend our work beyond the scope of this paper.

REPRODUCIBILITY STATEMENT

We use publicly available model architectures and datasets for training and evaluation. We discuss the method details in Section 3, and specifics of the training procedure in Section 4.1. We also provide a full summary of model configurations and training hyperparameters in Appendix Section C. The evaluation details can be found in Section C.4.

REFERENCES

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, 2024a.

Simran Arora, Sabri Eyuboglu, Michael Zhang, Aman Timalsina, Silas Alberti, Dylan Zinsley, James Zou, Atri Rudra, and Christopher Ré. Simple linear attention language models balance the recall-throughput tradeoff. *arXiv preprint arXiv:2402.18668*, 2024b.

Yushi Bai, Xin Lv, Jiajie Zhang, Hongchang Lyu, Jiankai Tang, Zhidian Huang, Zhengxiao Du, Xiao Liu, Aohan Zeng, Lei Hou, et al. Longbench: A bilingual, multitask benchmark for long context understanding. In *Proceedings of the 62nd annual meeting of the association for computational linguistics (volume 1: Long papers)*, pp. 3119–3137, 2024.

Ali Behrouz, Zeman Li, Yuan Deng, Peilin Zhong, Meisam Razaviyayn, and Vahab Mirrokni. Memory caching: Rnns with growing memory. *arXiv preprint arXiv:2602.24281*, 2026.

Aydar Bulatov, Yury Kuratov, and Mikhail Burtsev. Recurrent memory transformer. *Advances in neural information processing systems*, 35:11079–11091, 2022.

Zihang Dai, Zhilin Yang, Yiming Yang, Jaime G Carbonell, Quoc Le, and Ruslan Salakhutdinov. Transformer-xl: Attentive language models beyond a fixed-length context. In *Proceedings of the 57th annual meeting of the association for computational linguistics*, pp. 2978–2988, 2019.

Tri Dao and Albert Gu. Transformers are ssms: Generalized models and efficient algorithms through structured state space duality. *arXiv preprint arXiv:2405.21060*, 2024.

Jusen Du, Weigao Sun, Disen Lan, Jiaxi Hu, Zhang Tao, and Yu Cheng. Mom: Linear sequence modeling with mixture-of-memories. In *International Conference on Learning Representations*, volume 2026, pp. 106613–106631, 2026.

William Fedus, Barret Zoph, and Noam Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. arxiv 2021. *arXiv preprint arXiv:2101.03961*, 2021.

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.

Alex Graves, Greg Wayne, and Ivo Danihelka. Neural turing machines. *arXiv preprint arXiv:1410.5401*, 2014.

Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. *arXiv preprint arXiv:2312.00752*, 2023.

Guo Guo, Songlin Yang, Tarushii Goel, Eric P Xing, Tri Dao, and Yoon Kim. Log-linear attention. In *International Conference on Learning Representations*, volume 2026, pp. 91296–91317, 2026.

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.

Hee Seung Hwang, Xindi Wu, Sanghyuk Chun, and Olga Russakovsky. Reinforced fast weights with next-sequence prediction. *arXiv preprint arXiv:2602.16704*, 2026.

Samy Jelassi, David Brandfonbrener, Sham M Kakade, and Eran Malach. Repeat after me: Transformers are better than state space models at copying. *arXiv preprint arXiv:2402.01032*, 2024.

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.

Feyza Duman Keles, Pruthuvi Mahesakya Wijewardena, and Chinmay Hegde. On the computational complexity of self-attention. In *International conference on algorithmic learning theory*, pp. 597–619. PMLR, 2023.

Yuri Kuratov, Aydar Bulatov, Petr Anokhin, Ivan Rodkin, Dmitry Sorokin, Artyom Sorokin, and Mikhail Burtsev. Babilong: Testing the limits of llms with long context reasoning-in-a-haystack. *Advances in Neural Information Processing Systems*, 37:106519–106554, 2024.

Eshaan Nichani, Jason Lee, and Alberto Bietti. Understanding factual recall in transformers via associative memories. In *International Conference on Learning Representations*, volume 2025, pp. 4166–4207, 2025.

Yuqi Pan, Yongqi An, Zheng Li, Yuhong Chou, Ruijie Zhu, Xiaohui Wang, Mingxuan Wang, Jinqiao Wang, and Guoqi Li. Scaling linear attention with sparse state expansion. *arXiv preprint arXiv:2507.16577*, 2025.

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

Bo Peng, Eric Alcaide, Quentin Anthony, Alon Albalak, Samuel Arcadinho, Stella Biderman, Huanqi Cao, Xin Cheng, Michael Chung, Leon Derczynski, et al. Rwkv: Reinventing rnns for the transformer era. In *Findings of the association for computational linguistics: EMNLP 2023*, pp. 14048–14077, 2023.

Jack W Rae, Anna Potapenko, Siddhant M Jayakumar, and Timothy P Lillicrap. Compressive transformers for long-range sequence modelling. *arXiv preprint arXiv:1911.05507*, 2019.

Imanol Schlag, Kazuki Irie, and Jürgen Schmidhuber. Linear transformers are secretly fast weight programmers. In *International conference on machine learning*, pp. 9355–9366. PMLR, 2021.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv preprint arXiv:1701.06538*, 2017.

Sainbayar Sukhbaatar, Jason Weston, Rob Fergus, et al. End-to-end memory networks. *Advances in neural information processing systems*, 28, 2015.

Yutao Sun, Li Dong, Shaohan Huang, Shuming Ma, Yuqing Xia, Jilong Xue, Jianyong Wang, and Furu Wei. Retentive network: A successor to transformer for large language models. *arXiv preprint arXiv:2307.08621*, 2023.

TogetherAI. Long data collections database. https://huggingface.co/datasets/togethercomputer/Long-Data-Collections, 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.

Xin Wang, Hui Shen, Boyuan Zheng, Xueshen Liu, Minkyoung Cho, Zhongwei Wan, Zesen Zhao, Zhuoqing Mao, Shen Yan, and Mi Zhang. Dynamic linear attention. *arXiv preprint arXiv:2606.10650*, 2026.

Xindi Wu, Sven Elflein, James Lucas, Olga Russakovsky, Laura Leal-Taixé, Despoina Paschalidou, Jonathan Lorraine, and Aljoša Ošep. Addressable memory for video world models. *arXiv preprint arXiv:2608.07408*, 2026.

Yuhuai Wu, Markus N Rabe, DeLesley Hutchins, and Christian Szegedy. Memorizing transformers. *arXiv preprint arXiv:2203.08913*, 2022.

Songlin Yang, Bailin Wang, Yikang Shen, Rameswar Panda, and Yoon Kim. Gated linear attention transformers with hardware-efficient training. *arXiv preprint arXiv:2312.06635*, 2023.

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, 2024.

Songlin Yang, Jan Kautz, and Ali Hatamizadeh. Gated delta networks: Improving mamba2 with delta rule. In *International Conference on Learning Representations*, volume 2025, pp. 29687–29707, 2025.

Howard Yen, Tianyu Gao, Minmin Hou, Ke Ding, Daniel Fleischer, Peter Izsak, Moshe Wasserblat, and Danqi Chen. Helmet: How to evaluate long-context models effectively and thoroughly. In *International Conference on Learning Representations*, volume 2025, pp. 98914–98965, 2025.

Yu Zhang, Songlin Yang, Ruijie Zhu, Yue Zhang, Leyang Cui, Yiqiao Wang, Bolun Wang, Freda Shi, Bailin Wang, Wei Bi, et al. Gated slot attention for efficient linear-time sequence modeling. *Advances in Neural Information Processing Systems*, 37:116870–116898, 2024.

Appendices

A Notation **13**

B Related Work **15**

C Experiment Details **16**

C.1 Model Configurations 16

C.2 Mid-Training Hyperparameters 16

C.3 Datasets and Benchmarks 17

C.4 Evaluation Protocol. 17

C.5 Pseudocode 17

D Additional Analysis **18**

E Limitations **19**

A Notation

Table 5: Glossary and Notation.

Symbol	Description
Sequences and softmax attention	
T	Sequence / context length (number of tokens)
t	Position of the target token
$\boldsymbol{x}_t$	Hidden state of token t
d	Key/value dimension of one head's memory state (written d for both, whereas the backbones use $d_k \times d_v$, see Tab. 9)
$\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V} \in \mathbb{R}^{T \times d}$	Query, key, and value matrices
$\boldsymbol{q}_t, \boldsymbol{k}_t, \boldsymbol{v}_t$	Query, key, and value of token t
$\boldsymbol{W}_q,\ \boldsymbol{W}_k,\ \boldsymbol{W}_v$	Backbone projections producing $\boldsymbol{q}_t, \boldsymbol{k}_t, \boldsymbol{v}_t$ from $\boldsymbol{x}_t$
$\boldsymbol{M}$	Causal attention mask. In softmax attention it is additive with entries in $\{-\infty, 0\}$. In linear attention it is binary, with $\boldsymbol{M}_{ij} = 1$ if $i \geq j$ and 0 otherwise
$\boldsymbol{O},\ \boldsymbol{o}_t$	Attention output, as a matrix and for token t
$\mathrm{softmax}$	Row-wise softmax normalization
$\odot$	Elementwise (Hadamard) product
$\boldsymbol{I}$	Identity matrix
Linear attention	
$\boldsymbol{S}_t \in \mathbb{R}^{d \times d}$	Recurrent memory state at step t
$\alpha_t \in (0, 1)$	State decay gate in Gated DeltaNet (Eq. (2))

continued on next page

Table 5 (continued)

Symbol	Description
β_t	Write strength of the delta term (Eq. (2))
Multi-state recurrent memory	
L	Number of segments (memory states)
C_i	The i-th context segment
$i,\ \ell$	Segment indices ($1 \le i, \ell \le L$)
j	Index of a token within a segment ($j \in C_i$)
$m = \lvert C_i \rvert$	Number of tokens per segment
$\boldsymbol{S}_{C_i} \in \mathbb{R}^{d\times d}$	Memory state of segment i. For the plain linear recurrence, $\boldsymbol{S}_{C_i} = \sum_{j\in C_i} \boldsymbol{v}_j \boldsymbol{k}_j^\top$
$\boldsymbol{o}_{t,i}$	Segment-level output at position t: readout of segment i's state, $\boldsymbol{o}_{t,i} = \boldsymbol{S}_{C_i}\boldsymbol{q}_t$
$\lambda_{t,i}$	Aggregation weight of segment i at position t
$\boldsymbol{\lambda}_t,\ \boldsymbol{W}_\lambda$	Position-based weight vector (entries $\lambda_{t,i}$) and its projection
$\boldsymbol{W}_u,\ \boldsymbol{u}_t$	Target-token projection and vector used by mean-key routing
$\bar{\boldsymbol{k}}_i$	Meanpooled key of segment i
MERLIN*: key moments and aggregation weights*	
d_r	Moment dimension (hyperparameter)
$\boldsymbol{q}'_t,\ \boldsymbol{k}'_t \in \mathbb{R}^{d_r}$	New query and key of token t, projected from its hidden state
$\boldsymbol{W}_{\boldsymbol{q}'},\ \boldsymbol{W}_{\boldsymbol{k}'}$	Projections producing $\boldsymbol{q}'_t$ and $\boldsymbol{k}'_t$ from $\boldsymbol{x}_t$
$\boldsymbol{\mu_i} \in \mathbb{R}^{d_r}$	First moment of segment i's keys (mean of $\boldsymbol{k}'_t$ over C_i)
$\boldsymbol{\Sigma_i} \in \mathbb{R}^{d_r\times d_r}$	Uncentered second moment of segment i's keys (mean of $\boldsymbol{k}'_t{\boldsymbol{k}'_t}^\top$ over C_i)
$r^{\mu}_{t,i}$	Mean key–query alignment for segment i, $r^{\mu}_{t,i} = \boldsymbol{\mu_i}^\top \boldsymbol{q}'_t$
$r^{\Sigma}_{t,i}$	Variance of the key–query alignment over segment i, $r^{\Sigma}_{t,i} = {\boldsymbol{q}'_t}^\top \boldsymbol{\Sigma_i} \boldsymbol{q}'_t - (r^{\mu}_{t,i})^2$
γ_t	Learned gate mixing the softmax-normalized mean and variance terms (Eq. (9)), distinct from the decay gate above
$\tau_\mu,\ \tau_\Sigma$	Learnable temperatures for the mean term (τ_μ) and the variance term (τ_Σ)
Connection to softmax attention	
$s_{t,i,j}$	Query-key similarity, $s_{t,i,j} = {\boldsymbol{q}'_t}^\top \boldsymbol{k}'_j$ for $j \in C_i$
δ	Temperature in the grouped-softmax analysis (§3.3)
$Z_{t,i}(\delta)$	Length-normalized softmax evidence of segment i
$A_{t,i}$	Grouped softmax attention: total softmax mass on segment i
$\eta_{t,i}$	Mean of $s_{t,i,j}$ over C_i (first cumulant)
$\sigma^2_{t,i}$	Variance of $s_{t,i,j}$ over C_i (second cumulant)
Mamba-2 backbone (pseudocode)	
$P,\ N$	Head dimension and SSM state size, with $\boldsymbol{S}_{C_i} \in \mathbb{R}^{P\times N}$
$\Delta_t,\ A$	Step size at position t and per-head scalar decay rate, so the state decays by $\exp(\Delta_t A)$
$\boldsymbol{B}_t,\ \boldsymbol{D}_t$	Input matrix (the write key) and output matrix (the read query, written $\boldsymbol{C}$ in Mamba-2)
$\boldsymbol{W}_B,\ \boldsymbol{W}_D,\ \boldsymbol{W}_\Delta$	Projections producing $\boldsymbol{B}_t$, $\boldsymbol{D}_t$, and Δ_t from $\boldsymbol{x}_t$

B Related Work

Linear Attention and Recurrent Memory Models. Linear attention replaces the softmax normalization in standard attention with a feature map, turning attention into a recurrence with a fixed-size memory state (Katharopoulos et al., 2020). This state can be viewed as a fast-weight associative memory written by outer products (Schlag et al., 2021; Hwang et al., 2026). Recent works improve this recurrence with data-dependent gating (Yang et al., 2023), delta-rule updates that overwrite rather than accumulate (Yang et al., 2024), and their combination (Yang et al., 2025). Selective state-space models (Gu & Dao, 2023; Dao & Gu, 2024), and recurrent memory models such as RWKV (Peng et al., 2023) and RetNet (Sun et al., 2023) that are closely related to linear attention share the same fixed-size state form. These models compress the context into a single state, which may be limited in memory capacity due to information loss at longer contexts. Our work builds on these backbones to train more capable multi-state linear attention and recurrent memory models.

Multi-State and Hierarchical Memory. To improve the long-context capabilities of single-state linear attention and recurrent memory models, several methods partition the context into segments and compress each segment in its own memory state. As a result, these models maintain a bank of states over the context. Memory Caching (Behrouz et al., 2026) segments the context uniformly at fixed token length per segment. Log-linear attention organizes the recurrent states hierarchically by growing a logarithmic number of states along a Fenwick-tree partition (Guo et al., 2026) of the sequence. Dynamic linear attention determines state boundaries based on token-level information variation while bounding the cache by merging low-information states (Wang et al., 2026). Segment-recurrent and compressive Transformers cache or compress past activations across chunks (Dai et al., 2019; Rae et al., 2019; Bulatov et al., 2022; Wu et al., 2026). These works primarily focus on how states are formed via content segmentation (*i.e. separation*). However, we identify how the information extracted from each state is merged (*i.e. aggregation*) as a key bottleneck in multi-state linear attention and propose a method that improves that derivation of aggregation weights.

Routing over Memories via Aggregation. Multi-state linear attention and recurrent memory models extract information from each state and aggregate them to generate the final output representation of a token. Routing occurs during aggregation by adjusting the contribution of each segment to the final output. Mixture-of-Memories routes each token to one of several linear-attention states (Du et al., 2026). Sparse State Expansion routes tokens into state partitions with top-k hard selection (Pan et al., 2025). Gated Slot Attention reads a bounded set of slots through a softmax (Zhang et al., 2024). Gated Residual Memory scores cached memory modules with a context-aware gate, with a sparse variant that selects the most relevant ones (Behrouz et al., 2026). Such routers descend from mixture-of-experts gating (Shazeer et al., 2017; Fedus et al., 2021) and from content-based addressing of an external memory (Graves et al., 2014; Sukhbaatar et al., 2015; Wu et al., 2022). These aggregation methods assign weights based on position, the magnitude of its readout, or a single pooled key. However, MERLIN keeps first and second moments of each segment's keys to derive more informative aggregation weights.

Associative Recall and Long-Context Retrieval. Retrieving specific information from a long sequence is a core capability of models that handle long-context. In particular, such tasks evaluate the memory capacity of linear attention and recurrent memory models that compress the entire context into a single memory state. Multi-query associative recall and the Zoology analysis attribute much of the quality gap between recurrent memory models and standard transformers to recall (Arora et al., 2024a), and the recall-throughput tradeoff frames recall as bounded by recurrent state size (Arora et al., 2024b). Copying and factual-recall analyses make the capacity limit of the memory state more precise (Jelassi et al., 2024; Nichani et al., 2025). Furthermore, studies show that long-context models demonstrate positional biases where they prefer information located in certain depth within the context, usually close to the end (Hsieh et al., 2024; Yen et al., 2025). We use these benchmarks to demonstrate that MERLIN mitigates such biases by improving aggregation of segment-level outputs.

C EXPERIMENT DETAILS

We describe our experimental setup in detail. Section C.1 describes the model and method configurations, Section C.2 the mid-training hyperparameters, Section C.3 the datasets and benchmarks, and Section C.4 the evaluation protocol. In Section C.5 we provide the pseudocode for MERLIN.

C.1 MODEL CONFIGURATIONS

Tab. 6 lists the two backbones and the MERLIN hyperparameters. Gated DeltaNet-1.3B (21 layers, hidden size 2048) was pre-trained on 50B tokens of FineWeb-edu, and Mamba-2-780M (48 layers, hidden size 1536) on 300B tokens of the Pile. For both, MERLIN uses a segment size of m=256 and a moment dimension of d_r=128.

Table 6: **Model Configurations.**

	Gated DeltaNet-1.3B	**Mamba-2-780M**
Parameters	≈1.3B	≈780M
Pretraining corpus	FineWeb-edu	Pile
Pretraining tokens	50B	300B
Hidden size	2048	1536
Layers	21	48
Heads	6	48
Head dimension	256	64
SSM state size	–	128
Expansion factor	2	2
Short-conv kernel	4	4
MERLIN		
Segment size m	256	256
Moment dimension d_r	128	128
Key/query normalization	ℓ_2-normalized	
Temperatures τ_μ, τ_Σ, gate γ_t	learned	

C.2 MID-TRAINING HYPERPARAMETERS

Tab. 7 shows the mid-training recipe, which is shared by both backbones and by all m. We use AdamW with a weight decay of 0.1 and a peak learning rate of 1×10^{-4}, which follows a cosine schedule after 305 warmup steps. Each run uses 16,384-token sequences and a global batch of 131,072 tokens for 15,259 steps, which amounts to 2.0B tokens in bfloat16 mixed precision.

Table 7: **Mid-Training Hyperparameters.**

Hyperparameter	**Value**
Optimizer	AdamW ($\epsilon = 10^{-15}$)
Weight decay	0.1
Peak learning rate	1×10^{-4}
Learning-rate schedule	cosine (min ratio 0.1)
Warmup steps	305
Gradient clipping (max norm)	1.0
Sequence length	16,384
Global batch size	131,072 tokens
Training steps	15,259
Total tokens	2.0B
Precision	bfloat16 (mixed)
Random seed	42
Data	Long-Data-Collections, augmented with RULER NIAH/QA

C.3 Datasets and Benchmarks

Tab. 8 summarizes the mid-training data and the evaluation benchmarks, together with the metric, context length, and number of samples for each. We evaluate long-context retrieval on RULER (Hsieh et al., 2024) and long-context understanding on LongBench (Bai et al., 2024), both at up to a 16K context.

Table 8: **Summary of Datasets and Evaluation Benchmarks.** We mid-train on Long-Data-Collections augmented with RULER-style NIAH/QA, and evaluate retrieval on RULER and long-context understanding on LongBench (filtered to $\leq$16K tokens). Counts vary slightly by backbone tokenizer.

Phase	Dataset / Task	Metric	Context	Size
Mid-training	Long-Data-Collections (+ RULER NIAH/QA)	–	16K	2.0B tokens
RULER (retrieval)	S-NIAH-1/2/3	recall	16K	500 each
	MK-NIAH-1	recall	16K	500
	MQ-NIAH	recall	16K	500
	MV-NIAH	recall	16K	500
	HotpotQA	recall	16K	500
	SQuADQA	recall	16K	500
LongBench	NarrativeQA (NQ)	F1	$\leq$16K	61
	Qasper (QR)	F1	$\leq$16K	197
	MultiFieldQA (MF)	F1	$\leq$16K	150
	HotpotQA (HP)	F1	$\leq$16K	161
	2WikiMQA (2W)	F1	$\leq$16K	198
	GovReport (GR)	ROUGE	$\leq$16K	174
	QMSum (QM)	ROUGE	$\leq$16K	116
	MultiNews (MN)	ROUGE	$\leq$16K	200
	SAMSum (SS)	ROUGE	$\leq$16K	172
	TREC (TC)	accuracy	$\leq$16K	200
	TriviaQA (TQ)	F1	$\leq$16K	142

C.4 Evaluation Protocol

All evaluations use a 16K context length. We evaluate on RULER (Hsieh et al., 2024) and LongBench (Bai et al., 2024) following the official metrics and configurations from Language Model Evaluation Harness (Gao et al., 2024). For RULER, we run the eight NIAH and QA tasks listed in Tab. 8 and report the score on each task together with the average across tasks. For LongBench, we run the eleven subtasks in Tab. 8, keeping only the samples of at most 16K tokens, and score each task with its standard metric (F1, ROUGE, or accuracy).

For HELMET (Yen et al., 2025), we use the TriviaQA retrieval-augmented generation task and follow the official evaluation setup. Each prompt contains several passages, and only one of them, the gold passage, contains the answer to the question. We randomly shuffle the position of the gold passage within the context and report accuracy as a function of that position, from the first sixth $(1/6)$ to the last sixth $(6/6)$ of the context.

C.5 Pseudocode

Algorithm 1 shows MERLIN on a Gated DeltaNet backbone, assuming equal-size segments of m tokens each. The outer loop iterates over the L segments, and the inner loop iterates over the m tokens within each segment. For each token, the algorithm first updates the memory state of its segment together with the first and second moments of the segment's keys, so both accumulate token by token. It then reads out from every visible segment, meaning all closed segments plus the current, partially built one. Finally, it aggregates these readouts using the moment-derived weights from Eq. (9).

Algorithm 2 shows MERLIN on a Mamba-2 backbone. The structure is identical to Algorithm 1, but differs only on the memory state recurrence. Each token's hidden state is projected to the SSM

Algorithm 1 MERLIN (Gated DeltaNet backbone) with equal-size segments $C_1, \ldots, C_L$ of m tokens, learnable temperatures τ_μ, τ_Σ, and a learnable gate γ_t.

for $i = 1, \ldots, L$ **do**
  $\boldsymbol{S}_{C_i},\ \boldsymbol{\mu_i},\ \boldsymbol{\Sigma_i} \leftarrow \mathbf{0},\ \mathbf{0},\ \mathbf{0}$
  for $t \in C_i$ **do**
    $\boldsymbol{q}_t, \boldsymbol{k}_t, \boldsymbol{v}_t \leftarrow \boldsymbol{W}_q \boldsymbol{x}_t,\ \boldsymbol{W}_k \boldsymbol{x}_t,\ \boldsymbol{W}_v \boldsymbol{x}_t$
    $\boldsymbol{q}'_t, \boldsymbol{k}'_t \leftarrow \boldsymbol{W}_{q'} \boldsymbol{x}_t,\ \boldsymbol{W}_{k'} \boldsymbol{x}_t$
    $\boldsymbol{S}_{C_i} \leftarrow \boldsymbol{S}_{C_i} \left(\alpha_t \left(\boldsymbol{I} - \beta_t \boldsymbol{k}_t \boldsymbol{k}_t^\top\right)\right) + \beta_t \boldsymbol{v}_t \boldsymbol{k}_t^\top$ ▷ GDN memory state, Eq. (2)
    $\boldsymbol{\mu_i} \leftarrow \boldsymbol{\mu_i} + \frac{1}{m} \boldsymbol{k}'_t, \quad \boldsymbol{\Sigma_i} \leftarrow \boldsymbol{\Sigma_i} + \frac{1}{m} \boldsymbol{k}'_t \boldsymbol{k}'^\top_t$ ▷ moments of keys
    for $\ell = 1, \ldots, i$ **do**
      $\boldsymbol{o}_{t,\ell} \leftarrow \boldsymbol{S}_{C_\ell} \boldsymbol{q}_t, \quad r^{\mu}_{t,\ell} \leftarrow \boldsymbol{\mu_\ell}^\top \boldsymbol{q}'_t, \quad r^{\Sigma}_{t,\ell} \leftarrow \boldsymbol{q}'^\top_t \boldsymbol{\Sigma_\ell}\, \boldsymbol{q}'_t - (r^{\mu}_{t,\ell})^2$
    end for
    $\lambda_{t,\ell} \leftarrow \gamma_t \operatorname{softmax}_\ell(\tau_\mu\, r^{\mu}_{t,\ell}) + (1 - \gamma_t) \operatorname{softmax}_\ell(\tau_\Sigma\, r^{\Sigma}_{t,\ell})$ ▷ Eq. (9)
    $\boldsymbol{o}_t \leftarrow \sum_{\ell \le i} \lambda_{t,\ell}\, \boldsymbol{o}_{t,\ell}$ ▷ aggregate readouts, Eq. (4)
  end for
end for
return $\boldsymbol{o}_1, \ldots, \boldsymbol{o}_T$

Algorithm 2 MERLIN (Mamba-2 backbone) with equal-size segments $C_1, \ldots, C_L$ of m tokens, learnable temperatures τ_μ, τ_Σ, and a learnable gate γ_t. The state $\boldsymbol{S}_{C_i} \in \mathbb{R}^{P \times N}$, and the cached moments are shared per group.

for $i = 1, \ldots, L$ **do**
  $\boldsymbol{S}_{C_i},\ \boldsymbol{\mu_i},\ \boldsymbol{\Sigma_i} \leftarrow \mathbf{0},\ \mathbf{0},\ \mathbf{0}$
  for $t \in C_i$ **do**
    $\boldsymbol{v}_t, \boldsymbol{B}_t, \boldsymbol{D}_t, \Delta_t \leftarrow \boldsymbol{W}_v \boldsymbol{x}_t,\ \boldsymbol{W}_B \boldsymbol{x}_t,\ \boldsymbol{W}_D \boldsymbol{x}_t,\ \operatorname{softplus}(\boldsymbol{W}_\Delta \boldsymbol{x}_t)$
    $\boldsymbol{q}'_t, \boldsymbol{k}'_t \leftarrow \boldsymbol{W}_{q'} \boldsymbol{x}_t,\ \boldsymbol{W}_{k'} \boldsymbol{x}_t$
    $\boldsymbol{S}_{C_i} \leftarrow \exp(\Delta_t A)\, \boldsymbol{S}_{C_i} + \Delta_t\, \boldsymbol{v}_t \boldsymbol{B}_t^\top$ ▷ Mamba-2 SSD state
    $\boldsymbol{\mu_i} \leftarrow \boldsymbol{\mu_i} + \frac{1}{m} \boldsymbol{k}'_t, \quad \boldsymbol{\Sigma_i} \leftarrow \boldsymbol{\Sigma_i} + \frac{1}{m} \boldsymbol{k}'_t \boldsymbol{k}'^\top_t$ ▷ moments of keys
    for $\ell = 1, \ldots, i$ **do**
      $\boldsymbol{o}_{t,\ell} \leftarrow \boldsymbol{S}_{C_\ell} \boldsymbol{D}_t, \quad r^{\mu}_{t,\ell} \leftarrow \boldsymbol{\mu_\ell}^\top \boldsymbol{q}'_t, \quad r^{\Sigma}_{t,\ell} \leftarrow \boldsymbol{q}'^\top_t \boldsymbol{\Sigma_\ell}\, \boldsymbol{q}'_t - (r^{\mu}_{t,\ell})^2$
    end for
    $\lambda_{t,\ell} \leftarrow \gamma_t \operatorname{softmax}_\ell(\tau_\mu\, r^{\mu}_{t,\ell}) + (1 - \gamma_t) \operatorname{softmax}_\ell(\tau_\Sigma\, r^{\Sigma}_{t,\ell})$ ▷ Eq. (9)
    $\boldsymbol{o}_t \leftarrow \sum_{\ell \le i} \lambda_{t,\ell}\, \boldsymbol{o}_{t,\ell}$ ▷ aggregate readouts, Eq. (4)
  end for
end for
return $\boldsymbol{o}_1, \ldots, \boldsymbol{o}_T$

value $\boldsymbol{v}_t$, input matrix $\boldsymbol{B}_t$ (the write key), output matrix $\boldsymbol{D}_t$ (the read query), and step size Δ_t. The segment state $\boldsymbol{S}_{C_i} \in \mathbb{R}^{P \times N}$ (P the head dimension, N the SSM state size) follows the Mamba-2 SSD recurrence with per-head scalar decay $\exp(\Delta_t A)$. In Mamba-2, the cached moments are shared per group. The aggregation step is identical to Eq. (9).

D ADDITIONAL ANALYSIS

Memory Cost. Tab. 9 reports the recurrent memory retained to decode each token, relative to the single fixed-state (Vanilla) cache. All multi-state methods cache the $L = T/m$ segment states, which dominate the cost. Position adds no cached moments, and Meanpool adds a negligible first moment ($\le 0.1\%$), while MERLIN additionally caches a $d_r \times d_r$ second moment. This extra cost is controlled by d_r and is at most 12.6% for Gated DeltaNet and 4.2% for Mamba-2 (whose moments are shared per group) at d_r=128, shrinking quadratically as d_r decreases.

Parameter Overhead. We show the number of parameters that each aggregation method adds to the backbone under default model configurations ($m = 256, d_r = 128$). The parameter overhead of MERLIN roughly equals the Meanpool baseline (4.5% for Gated DeltaNet-1.3B and 2.2% for

Table 9: **Relative Decode Memory Cache** ($\times$ the single fixed-state Vanilla cache) at a 16K context, by segment size m (number of states $L{=}T/m$) and moment dimension d_r. All multi-state methods cache the L segment states. Position caches no moments, Meanpool caches a first moment ($\leq 0.1\%$), and MERLIN caches a $d_r \times d_r$ second moment. MERLIN's overhead is larger for Gated DeltaNet (per-head router) than Mamba-2 (per-group router).

(a) Gated DeltaNet-1.3B ($d_k \times d_v = 256 \times 512$)

Method	m (states L)			
	512 (32)	256 (64)	128 (128)	64 (256)
Vanilla	1×	1×	1×	1×
Position	32.0×	64.0×	128.0×	256.0×
Meanpool	32.0×	64.0×	128.0×	256.0×
MERLIN (d_r=16)	32.1×	64.1×	128.3×	256.5×
MERLIN (d_r=32)	32.3×	64.5×	129.0×	258.1×
MERLIN (d_r=64)	33.0×	66.0×	132.1×	264.1×
MERLIN (d_r=128)	36.0×	72.1×	144.1×	288.3×

(b) Mamba-2-780M ($P \times N = 64 \times 128$)

Method	m (states L)			
	512 (32)	256 (64)	128 (128)	64 (256)
Vanilla	1×	1×	1×	1×
Position	32.0×	64.0×	128.0×	256.0×
Meanpool	32.0×	64.0×	128.0×	256.0×
MERLIN (d_r=16)	32.0×	64.0×	128.1×	256.2×
MERLIN (d_r=32)	32.1×	64.2×	128.3×	256.7×
MERLIN (d_r=64)	32.3×	64.7×	129.4×	258.7×
MERLIN (d_r=128)	33.3×	66.7×	133.4×	266.7×

Mamba-2-780M). The Position baseline is relatively lightweight, with only 1.2% and 0.6% overhead for Gated DeltaNet-1.3B and Mamba-2-780M respectively.

Table 10: **Parameter Overhead of the Aggregation Mechanisms.** Parameters added in the aggregation pathway on top of each shared backbone. We assume default model configurations where $m = 256, d_r{=}128$.

Backbone	Aggregation	Added	%
GatedDeltaNet-1.3B	Vanilla	0	0.0%
	POSITION	16.8M	1.2%
	MEANPOOL	66.1M	4.5%
	MERLIN	66.3M	4.6%
Mamba-2-780M	Vanilla	0	0.0%
	POSITION	4.8M	0.6%
	MEANPOOL	18.9M	2.2%
	MERLIN	18.9M	2.2%

Qualitative Examples. Tab. 11 illustrates the aggregation weight assignment on individual segments given retrieval-augmented generation samples in HELMET Yen et al. (2025) (two Natural Questions, two TriviaQA, all at 16K context). For each query we show the gold segment that contains the ground truth as well as four randomly sampled distractor (irrelevant) segments, and the aggregation weight $\lambda_{t,i}$ each receives in the last layer. MERLIN assigns highest weights on the gold segment, well above the recency-driven POSITION router and above MEANPOOL.

E LIMITATIONS

Due to compute constraints, our experiments cover two backbones (Gated DeltaNet-1.3B and Mamba-2-780M) at a 16K context length, and we mid-train multi-state models from single-state checkpoints. We compare MERLIN against Vanilla, Position, and Meanpool aggregation with uniform segmentation. Studying larger models, longer contexts, and a broader set of aggregation baselines is beyond our current compute budget, and we leave it to future work.

Table 11: **Qualitative Examples.** We evaluate the GDN-based models on 4 random RAG samples (2 from Natural Questions and 2 from TriviaQA) in HELMET (Yen et al., 2025) 16K. We show the last layer's aggregation weights assigned to the ground truth and distractor segments for each aggregation method. MERLIN assigns highest weights to the ground truth segment compared to the Meanpool and Position baselines.

Index	Type	Candidate segment	POSITION	MEANPOOL	MERLIN
Question: "what is another name for the water cycle?" **Answer:** hydrological cycle					
9	Distractor	...of the world's water supply, about 321,000,000 cubic miles of it...	0.006	0.018	0.017
22	Distractor	...water vapour or aqueous vapor is the gaseous phase of water...	0.005	0.019	0.016
33	**Ground Truth**	...*also known as the hydrologic cycle or the* ***hydrological cycle****, the continuous movement of water...*	0.005	0.030	**0.055**
35	Distractor	...cultural eutrophication of lakes is primarily due to phosphorus...	0.005	0.008	0.005
54	Distractor	...loch or fen; each has a special meaning, though modern usage...	0.005	0.010	0.007
Question: "where are the Washington Redskins based out of?" **Answer:** the Washington metropolitan area					
8	Distractor	...Franchise history: the Daniel Snyder ownership era (1999–present)...	0.004	0.010	0.012
28	Distractor	...Daniel Snyder ...W. Smith, the founder of Federal Express...	0.004	0.016	0.013
37	**Ground Truth**	...*a professional football team based in* ***the Washington metropolitan area****; they compete in the NFL...*	0.004	0.034	**0.050**
39	Distractor	...History. The Redskins relocated from Boston, Massachusetts...	0.004	0.018	0.031
50	Distractor	...a home for that team; its highest-ranking teams as of 2015...	0.004	0.030	0.027
Question: "in which Charles Dickens novel does the character Mr M Choakumchild appear?" **Answer:** Hard Times					
16	Distractor	...the illiterate poor chipped in ha'pennies to have each new monthly episode...	0.004	0.010	0.024
26	**Ground Truth**	...*where he wrote "Bleak House" (1852–53) and* ***"Hard Times"*** *(1854)...*	0.004	0.021	**0.060**
42	Distractor	...Mrs Nickleby is based on his mother, though she did not recognize herself...	0.004	0.012	0.010
45	Distractor	...the novel "Nicholas Nickleby", and Mrs Micawber in "David Copperfield"...	0.004	0.023	0.010
53	Distractor	...Dickens in films and TV: Lennox Pawle (1935)...	0.004	0.019	0.007
Question: "which chain of European retailers began near a crossroads in Annecy, France in 1958?" **Answer:** Carrefour					
6	Distractor	...the family also holds a stake in 3 Suisses, Norauto and others...	0.007	0.021	0.016
25	**Ground Truth**	...***Carrefour*** *S.A. is a French multinational retailer...*	0.007	0.019	**0.049**
31	Distractor	...within its metropolitan boundaries; the Cagliari–Sarroch port system...	0.007	0.013	0.016
37	Distractor	...17 September 2012 ...Jacqueline de Jong; an internment camp...	0.007	0.008	0.006
42	Distractor	...some trades are also present along the avenue de Verdun...	0.007	0.011	0.007