COMPETING LEARNING IN HYBRID ARCHITECTURES

Anonymous authors
Paper under double-blind review

ABSTRACT

Many modern language models have adopted hybrid architectures that incorporate both linear-time recurrence and softmax attention. Modern recurrence layers provide a highly expressive mixing mechanism while ensuring high throughput by compressing contextual information into a fixed-size state, but underperform softmax attention in contextual information retrieval. In this paper, we reveal a *Competing Learning* phenomenon where different kinds of mixers compete to learn the same function, and when any layer learns a function, it suppresses the corresponding learning signal for other layers. Layer attribution analysis reveals that under typical data distributions, modern recurrence learns retrieval functions faster than softmax attention, causing attention to fail to learn retrieval in most cases. Stopping weight updates in recurrence in the middle of training helps attention learn retrieval. Based on these results, we propose a staged training procedure for pre-training hybrid architectures on natural language. Empirical results on models up to 1B parameters show that our method results in substantial improvement in language modeling loss, common-sense reasoning, as well as a wide range of retrieval-related tasks across different context lengths.

1 INTRODUCTION

Recently, many open-weight large language models (LLMs) have adopted *hybrid architectures* that combine two distinct types of token mixers: linear recurrence (e.g., state space models (SSMs) and linear attention) and softmax attention (Vaswani et al., 2017). Such models have demonstrated strong language modeling capabilities while offering improved training and inference efficiency, scaling to models with up to trillions of parameters. Representative examples include Kimi K3 (Team Kimi et al., 2026), GLM-5.3-Flash (GLM Team, 2026), Qwen3.8-Flash-Next (Qwen Team, 2026), and Nemotron-3-Ultra (NVIDIA Team, 2026).

Prior work has shown that, in well-trained LLMs, different token mixers—and/or different heads within the same mixer—can specialize in distinct mixing patterns. For instance, Xiao et al. (2025) show that a subset of attention heads in Transformer models (Vaswani et al., 2017), referred to as *retrieval heads*, learn long-distance mixing patterns, whereas other heads primarily capture short-distance interactions. Similarly, Bick et al. (2025) show that different heads within a linear recurrence layer exhibit retrieval behaviors over different distances. In hybrid architectures, Chen et al. (2026) further show that assigning long-distance retrieval to softmax attention and short-distance mixing to recurrent layers is beneficial across a wide range of downstream tasks. Together, these findings motivate a division of labor in hybrid models, where softmax attention specializes in long-distance mixing while recurrent layers specialize in short-distance mixing.

In this work, we identify *Competing Learning*, a phenomenon in which layers that acquire retrieval capabilities earlier inhibit the development of these capabilities in other layers. In the hybrid models we study, recurrent layers tend to gain this early advantage, taking over retrieval-related functions that softmax attention layers are intended to specialize in. Consequently, training hybrid models using standard practices results in softmax attention's retrieval learning being limited by recurrent layers, leading to suboptimal division of labor.

Concretely, we provide an analysis based on layer-wise contribution attribution on synthetic retrieval tasks to identify the speed of retrieval learning of different mixers in a hybrid model. The analysis provides evidence to illustrate how the learning of retrieval functions in softmax attention is inhibited by the retrieval learning of recurrent layers. Furthermore, manually stopping recurrent layers

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.

from learning retrieval at different points in the training process reveals that softmax attention starts learning retrieval as soon as recurrent layers' learning is stopped. We also discover that under typical training distributions, with more short-distance patterns than long-distance ones, recurrence learns retrieval faster than softmax attention, resulting in suboptimal division of labor.

To address this issue, we propose Better Layer Division (BLD), a training pipeline that introduces a preliminary training stage before standard pretraining to give full-attention layers an early advantage in learning retrieval. This intervention is designed to steer Competing Learning in favor of full-attention layers, encouraging them to assume the long-range retrieval role during subsequent pretraining. We evaluate BLD on widely used benchmarks (e.g. Common-sense reasoning (CSR), recall-intensive tasks, LongPPL (Fang et al., 2025).) using hybrid models based on Kimi Delta Attention (KDA)[1] (Kimi Team et al., 2025) at multiple scales, and observe improvements in benchmarks including CSR, recall-intensive tasks, RULER, LongPPL and training loss over vanilla ones.

Our main contributions are as follows:

- We investigate the training dynamics of layer-wise functional specialization in hybrid models, focusing on the allocation of retrieval-related functions between recurrent and full-attention layers.
- We identify and characterize Competing Learning through empirical analyses, revealing how early differences in retrieval learning influence subsequent layer specialization.
- We propose BLD, a training pipeline that guides functional specialization by freezing the parameters of selected components during the early stage of pretraining, and evaluate its effectiveness on KDA-based hybrid models at multiple scales.

2 Related Work

Hybrid Architectures Recent LLMs increasingly adopt hybrid architectures that combine softmax attention with linear attention, including GLM-5.3-Flash (GLM Team, 2026), Kimi K3 (Team Kimi et al., 2026), Qwen3.8-Flash-Next (Qwen Team, 2026), and Nemotron-3-Ultra (NVIDIA Team, 2026). Such designs combine the retrieval strength of softmax attention with the computational efficiency of linear attention. Prior work has shown that linear or recurrent models can underperform softmax attention on precise in-context retrieval (Arora et al., 2024; Bick et al., 2025; Jelassi et al., 2024). Several works improve the memory or retrieval capabilities of linear attention through architectural or state-space enhancements (Yang et al., 2023; 2025b; Qin et al., 2024; Shen et al., 2026). In contrast, our work does not enhance the retrieval capacity of linear attention itself, but improves the pre-training of hybrid models without architectural modifications. Hybrid architectures instead retain softmax-attention layers for long-range retrieval while using efficient linear attention for most computation (Kimi Team et al., 2025). Our work focuses on better realizing this architectural complementarity during pre-training. Most of our experiments are conducted on Kimi-Linear-style models based on Kimi Delta Attention (KDA) (Kimi Team et al., 2025).

Labor Division in Non-Hybrid Models Prior work has shown that only a small subset of attention heads develops strong long-context retrieval behavior (Wu et al., 2025). DuoAttention exploits this functional specialization by distinguishing retrieval heads from streaming heads for efficient inference (Xiao et al., 2025). Other works leverage already-formed retrieval heads to construct post-training signals for improving long-context performance and contextual faithfulness (Huang et al., 2025; Ma & Okazaki, 2026). Recent studies further analyze how retrieval heads are learned or evolve during context extension and long-context continual pre-training (Zhao et al., 2024; Liang et al., 2026). These works mainly study or exploit specialization among homogeneous softmax-attention heads, rather than the allocation of retrieval capability across heterogeneous attention components. In hybrid models, however, such allocation is critical: if all retrieval heads emerge in linear-attention components, the intended architectural advantage of combining efficient linear attention with the stronger retrieval capability of softmax attention would be largely undermined.

Labor Division in Hybrid Models Prior work has observed functional specialization between attention and recurrent components in hybrid models, with attention playing a particularly important

[1] Widely-used RNN architecture adopted by frontier models (e.g. Kimi K3, GLM-5.3-Flash).

role in exact retrieval and recall (Bick et al., 2025; Afendulev et al., 2026; Wang et al., 2025). These works mainly characterize specialization after it has emerged, while other studies exploit such specialization to construct hybrid models from pretrained Transformers (Chen et al., 2026; Bick et al., 2026; Li et al., 2026). In contrast, we study how this division of labor forms during hybrid-model pre-training and how it can be actively optimized. In the work most closely related to ours, Qiao et al. (2026) show that efficient-attention modules can weaken the optimization pressure for full-attention layers and delay retrieval-head formation. Our work further identifies the underlying competition in retrieval-specific learning signals: linear-attention and softmax-attention layers both gradually acquire retrieval capabilities, and layers that learn retrieval earlier reduce the retrieval-specific prediction error, thereby weakening the learning signal available to other layers. We refer to this as the Competing Learning phenomenon and use it to guide pre-training toward stronger retrieval specialization in softmax-attention layers.

3 Competing Learning in Hybrid Attention Models

Central to our paper is the Competing Learning phenomenon, which is stated as follows:

Competing Learning Phenomenon. During hybrid-model pre-training, recurrent and softmax-attention layers both gradually acquire in-context retrieval capabilities. If some layers partially learn retrieval early in training, they reduce the retrieval-specific prediction error—and thus the learning signal—available to other layers, slowing their acquisition of this capability.

To show the causes and effects of this phenomenon, we conduct a series of layer attribution and gradient intervention experiments to monitor the retrieval learning of each layer in a hybrid model. In the following sections, we first propose a synthetic retrieval task (Section 3.1). Then, we propose a novel metric for determining how much a layer contributes to the model's retrieval abilities (Section 3.2). Finally, we train a series of hybrid models on this task and inspect the contributions of each layer to monitor their retrieval learning over time (Section 3.3).

Concretely, we conduct the following two sets of experiments:

- Section 3.3.1: We provide the experimental evidence for the Competing Learning phenomenon with weight freezing.
- Section 3.3.2: We show how the distribution of short and long-distance patterns chooses the winner of Competing Learning phenomenon.

Altogether, we demonstrate the existence of Competing Learning and KDA layers' huge advantage in training data with short-distance patterns. Since short-distance patterns are more frequent than long-distance patterns in natural language pretraining (Futrell et al., 2015), this leads to a suboptimal labor division in direct pretraining of hybrid models.

3.1 Spaced Retrieval

In order to evaluate retrieval ability without interference from complicated pattern in natural language, we craft a synthetic task called "Spaced Retrieval" in the following format:

$$\begin{aligned} \text{Input:} \quad & \$\mathtt{X}_1\mathtt{X}_2\cdots\mathtt{X}_t \underbrace{\sqcup\cdots\sqcup}_{g \text{ spaces}} \$ \\ \text{Output:} \quad & \mathtt{X}_1\mathtt{X}_2\cdots\mathtt{X}_t \end{aligned}$$

where g is the gap between the source and target sequence. Each $\mathtt{X}_i$ is a token sampled from $\{0, 1, \ldots, 9\}$ uniformly at random. We can adjust t, g to change the difficulty of this task. We use $t = 5$ and $g \in \{64, 8192\}$ for the training and evaluation in following experiments.

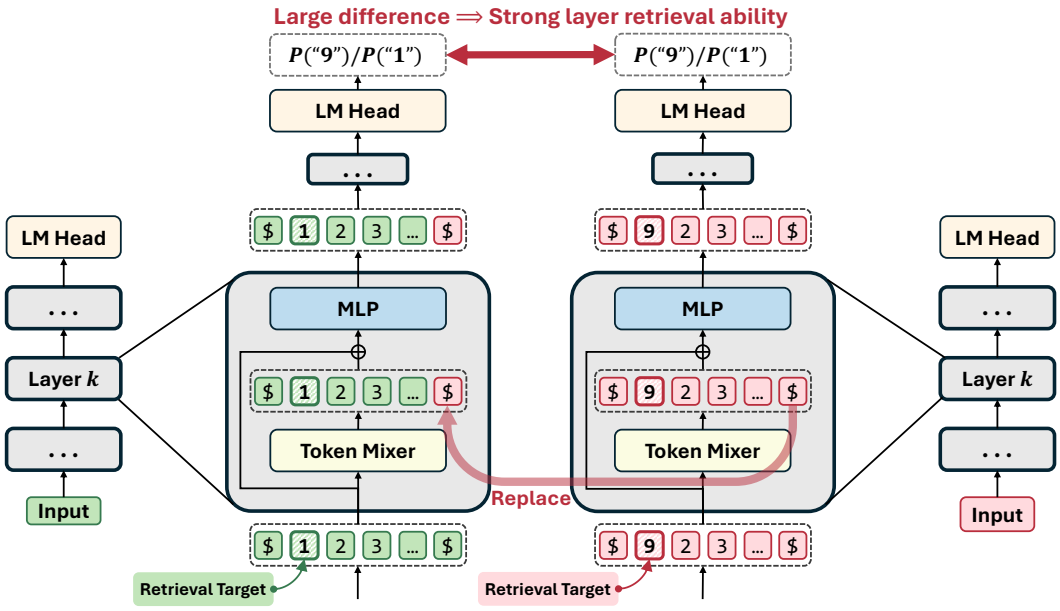

Figure 1: Illustration of our layer retrieval ability evaluation method (described in Section 3.2). As an example, the sequence to copy is "123", and the layer under inspection is layer k. The difference in probability change tells us the contribution of layer k in retrieving the target token "9".

3.2 Estimating Layer Retrieval Ability Score

Model Notations Let $\mathbf{H}_{l-1}(x) \in \mathbb{R}^{N\times d}$ be the residual stream entering the l-th layer for input x. Each layer computes

$$\mathbf{U}_l(x) = \mathrm{Mixer}_l\big(\mathbf{H}_{l-1}(x)\big), \tag{1}$$
$$\mathbf{R}_l(x) = \mathbf{H}_{l-1}(x) + \mathbf{U}_l(x), \tag{2}$$
$$\mathbf{H}_l(x) = \mathbf{R}_l(x) + \mathrm{MLP}_l\big(\mathbf{R}_l(x)\big), \tag{3}$$

where Mixer_l is a recurrence or softmax attention layer. We write $\mathbf{h}_l(x,t) = [\mathbf{U}_l(x)]_t$ for its output at token position t, before residual addition. Let $\mathbf{z}(x,t)$ denote the final logits from the model's native output head, and define the answer margin of two tokens a, b as

$$m(\mathbf{z}; a, b) = [\mathbf{z}]_a - [\mathbf{z}]_b. \tag{4}$$

Retrieval Ability Estimation Consider two input prefixes x^A and x^B of equal length j, differing only in the source digit to be retrieved: a in x^A and b in x^B. Their mixer outputs at layer l are denoted by $\mathbf{U}_l(x^A)$ and $\mathbf{U}_l(x^B)$. We replace only the output at the last unmasked position, denoting U_l by U:

$$\begin{aligned}\mathbf{U}(x^A) &= \big[\mathbf{U}_1(x^A), \ldots, \mathbf{U}_{j-1}(x^A), \mathbf{U}_j(x^A)\big],\\ \widetilde{\mathbf{U}}^{A\leftarrow B} &= \big[\mathbf{U}_1(x^A), \ldots, \mathbf{U}_{j-1}(x^A), \mathbf{U}_j(x^B)\big],\end{aligned} \tag{5}$$

where $\mathbf{U}_{l,t}(x)$ denotes the mixer output at position t.

During the forward pass of x^A, we substitute $\widetilde{\mathbf{U}}^{A\leftarrow B}$ for $\mathbf{U}(x^A)$ before residual addition, retaining the residual input from x^A. The current layer's MLP and all subsequent computations proceed normally with unchanged parameters. We denote the resulting logits at position j by $\widetilde{\mathbf{z}}^{A\leftarrow B}_{l,j}$.

The **retrieval ability** score of layer l for this input pair and answer position is

$$\boxed{C_l^{A\leftarrow B} = m\big(\mathbf{z}(x^A, j); a, b\big) - m\big(\widetilde{\mathbf{z}}^{A\leftarrow B}_{l,j}; a, b\big).} \tag{6}$$

This score indicates the reduction in correct-answer preference after replacing the layer's output with a misleading output with a different answer. Therefore, the layers with importance in retrieving have large scores, which means that when they are interfered with, the model can hardly retrieve.

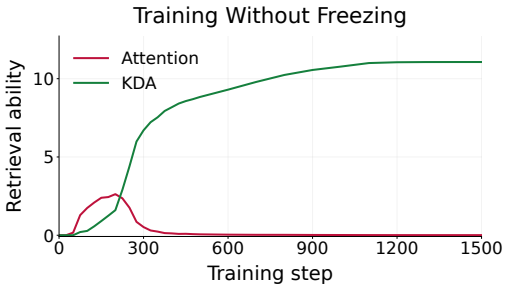

Figure 2: The average retrieval ability of the two types of layer in a hybrid model trained on Spaced Retrieval.

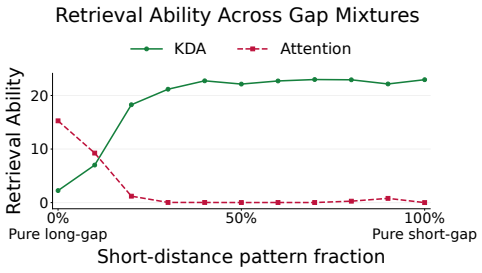

Figure 3: The retrieval ability of the two different kinds of layer in a hybrid model, trained on Spaced Retrieval with different data mixtures.

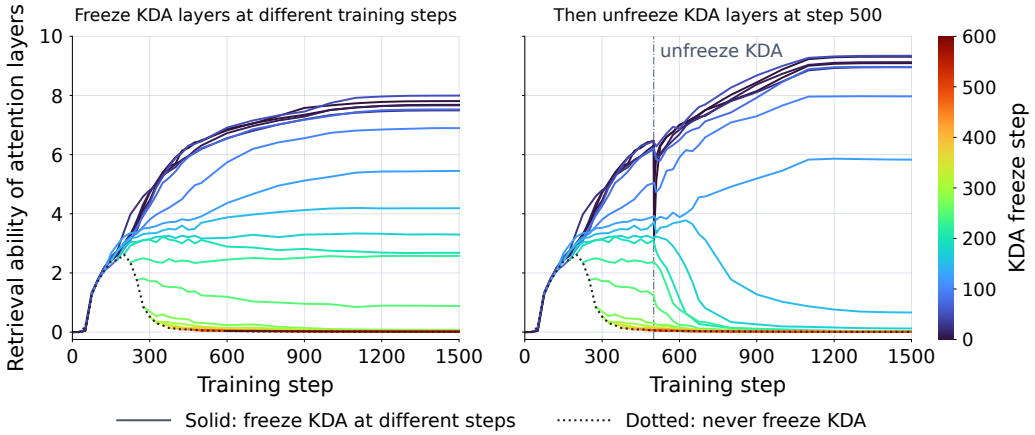

Figure 4: Demonstration of Competing Learning. The curves indicate the retrieval ability of attention layers in a hybrid model consisting of attention and recurrent layers (KDA), trained on our retrieval task, Spaced Retrieval (Section 3.1).

3.3 Analysis Experiments

Experimental Configuration We train several hybrid models on the Spaced Retrieval task. Each model has 3.6M parameters and consists of two KDA (a widely-used linear recurrence architecture) layers and two attention layers. We use 1:1 mixtures of examples with $g = 64$ and $g = 8192$ (except for Figure 3). During training, we only compute the loss on output tokens. Here, we report the results for $g = 8192$, but we emphasize that the results of $g = 64$ are similar. The results for $g = 64$ and more experimental details can be found in Appendix A.

3.3.1 Suppressing Learning in One Token Mixer Stimulates Learning in Another

Without intervention, KDA layers become dominant in terms of retrieval ability in this setting, as shown in Figure 2. To investigate how their dominance affects the attention layers, we freeze all KDA layers at different training steps and continue training the remaining model components. As shown in the left part of Figure 4, the attention layers' retrieval ability increases sharply immediately after each freezing intervention. This observation is consistent with the hypothesis that, once KDA layers acquire some retrieval capability, they suppress the acquisition of the same capability by attention layers, providing evidence for the Competing Learning phenomenon.

Furthermore, we explore how unfreezing KDA layers at a later training step influences learning competition. In the right plot of Figure 4, runs in which KDA is frozen earlier than step 125 have higher retrieval ability than when the KDA remains frozen for the rest of the training, while runs in which

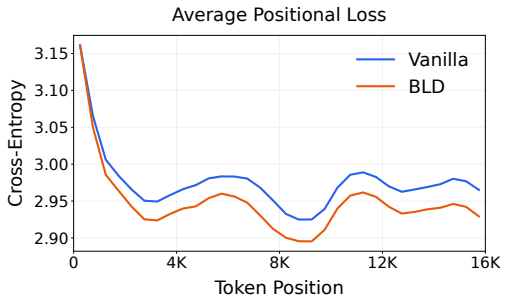

Figure 5: Average position-wise loss of 510M BLD/vanilla model across all examples.

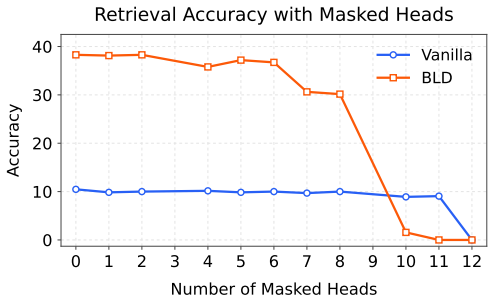

Figure 6: The retrieval accuracy when masking a fraction of the heads.

KDA is frozen later than step 125 have lower retrieval ability than when the KDA remains frozen. Moreover, these changes appear almost immediately after KDA is unfrozen. This clear polarization may be attributed to the fact that attention layers in checkpoints where freezing began earlier than step 125 have enough retrieval advantages by freezing KDA layers long enough. These advantages are sufficient to make attention layers win in Competing Learning when the freezing intervention is lifted. Therefore, their retrieval ability increases after unfreezing. By contrast, attention layers in checkpoints intervened later than step 125 do not have a sufficiently large advantage to win in Competing Learning, so their retrieval ability decreases again after unfreezing.

Takeaways The intervention results provide direct evidence for the Competing Learning phenomenon: KDA and attention layers compete to learn retrieval, and temporarily freezing the dominant KDA layers can shift—and, with sufficient duration, even reverse—the winner of this competition.

3.3.2 Different Training Data Prefer Different Architectures

We further investigate how the distribution of gap (g) in the training data influences the learnability of retrieval in different mixers. To this end, we train the hybrid model using different mixture proportions of short-distance ($g = 64$) and long-distance ($g = 8192$) examples. As shown in Figure 3, attention layers become more important as the proportion of long-distance examples increases, whereas KDA layers are more likely to dominate when the long-distance patterns do not appear overwhelmingly more than short-distance patterns. This is because KDA has a uniformly initialized forget gate, which gives this architecture a short-distance inductive bias.

Takeaways Different mixers learn retrieval at different speeds, depending on the data distribution. While both mixers can acquire the same retrieval abilities, the mixer that learns faster will become dominant, limiting the learning of the other mixer.

Therefore, applying the traditional training recipe to a hybrid model can lead to a suboptimal division of labor. Because short-distance patterns occur more frequently than long-distance patterns in natural language (Futrell et al., 2015), the training objective may encourage KDA to dominate retrieval, suppressing the training of attention layers. This issue is less problematic in homogeneous architectures such as Transformers, where all layers share the same architecture and, consequently, have comparable potential for retrieval.

4 Experiments on LLM Pretraining

In this section, we show that we can leverage insights about Competing Learning to achieve improved LLM performance in natural language pre-training.

4.1 METHOD: BETTER LAYER DIVISION

Based on the analysis results in Section 3.3.1, we propose to freeze the RNN layers' weights during the early stage of pretraining, thereby giving the attention layers a first-mover advantage in Competing Learning. Importantly, this intervention establishes a corrected division of labor between the two components, which is subsequently preserved and further reinforced throughout the remainder of pretraining. This better labor division translates to improved model performance.

4.2 EXPERIMENTAL DETAILS

Model Configuration We train a family of KDA–attention hybrid models ranging from 180M to 1B total parameters, including embeddings. All models share a repeating four-layer pattern consisting of one attention layer followed by three KDA layers, maintaining a fixed 3:1 KDA-to-attention ratio across scales. The layers are stacked in the form of "Attention-KDA-KDA-KDA", and all layers are using NoPE (not adopting RoPE, which is reported to be better than using RoPE in KDA-based hybrid architecture (Kimi Team et al., 2025)). We scale the models by varying their depth and hidden width. At each scale, BLD and vanilla pre-training use identical architectures. Detailed model configurations are provided in Appendix B.

Training Configuration We compare BLD against vanilla pre-training by training models from scratch with token budgets scaled proportionally to their non-embedding parameter counts, following the Chinchilla scaling principle (training tokens are nearly $20\times$ of parameters) (Hoffmann et al., 2022). All models are trained on DCLM (Li et al., 2024), a high-quality open-source pre-training corpus, using the Qwen3 tokenizer (Yang et al., 2025a) and a context length of 16,384 tokens. For BLD, the KDA blocks remain frozen for approximately $\frac{1}{10}$ of the subsequently unfrozen pretraining part. The vanilla baseline updates all parameters, using the same training data, token budgets, and optimization schedule.

We use AdamW during the initial portion of pre-training, followed by Muon with auxiliary AdamW after unfreezing, with identical optimization schedules for BLD and the vanilla baseline. Detailed hyperparameters and token budgets are provided in Appendix C.

Evaluation Tasks We evaluate models across common-sense reasoning (CSR), recall-intensive tasks, RULER, and language modeling, without fine-tuning. See Appendix D for more details.

- CSR is measured by the average score on seven benchmarks: ARC-Easy, ARC-Challenge (Clark et al., 2018), HellaSwag (Zellers et al., 2019), PIQA (Bisk et al., 2020), BoolQ (Clark et al., 2019), WinoGrande (Sakaguchi et al., 2019), LAMBADA (Paperno et al., 2016).
- Recall is evaluated on six tasks from the Based (Arora et al., 2024) evaluation suite: SWDE (Lockard et al., 2019), FDA (Arora et al., 2023), SQuAD (Rajpurkar et al., 2018), TriviaQA (Joshi et al., 2017), DROP (Dua et al., 2019), and Natural Questions (Kwiatkowski et al., 2019).
- RULER (Hsieh et al., 2024) is evaluated with average scores across the 57 evaluated RULER task–length combinations spanning 1K to 16K tokens.
- Language modeling is evaluated using word-level perplexity on WikiText-2 (Merity et al., 2016) and selected-token perplexity on LongPPL (Fang et al., 2025).

4.3 MAIN RESULTS

Downstream Tasks As shown in Table 1, BLD models achieve notably lower LM losses. They also show a modest improvement in CSR performance, while exhibiting substantial gains on retrieval-oriented tasks, including Recall and RULER. Moreover, the PPL metrics, which reflect overall long-context modeling capability, consistently favor BLD by a large margin. The specific subtask scores are included in Appendix E.

Size	Model	LM Loss ↓	CSR ↑	Recall ↑	RULER ↑	WikiText ↓	LongPPL ↓
180M	Vanilla	3.5903	**37.76**	10.39	2.84	54.00	258.47
180M	BLD	**3.5702**	37.49	**14.12**	**8.82**	**50.38**	**108.64**
250M	Vanilla	3.2844	38.31	21.54	17.32	36.52	57.22
250M	BLD	**3.2642**	**41.49**	**25.33**	**20.98**	**34.27**	**17.96**
390M	Vanilla	3.1611	41.91	26.73	22.32	29.10	17.09
390M	BLD	**3.1441**	**44.25**	**31.27**	**26.89**	**28.11**	**11.16**
510M	Vanilla	3.0553	**44.80**	30.50	25.25	26.02	14.56
510M	BLD	**3.0461**	44.49	**34.70**	**29.42**	**25.57**	**9.21**
750M	Vanilla	2.9773	45.94	33.52	26.53	23.28	12.68
750M	BLD	**2.9705**	**46.61**	**37.52**	**32.82**	**22.75**	**8.08**
1B	Vanilla	2.8744	48.69	36.50	32.75	20.52	7.69
1B	BLD	**2.8680**	**49.45**	**38.14**	**36.04**	**20.15**	**6.50**

Table 1: Zero-shot evaluation results of BLD (ours) versus vanilla pre-training. Size denotes total parameters, including embeddings, rounded to the nearest 10M. CSR and Recall are macro averages over seven and six tasks, respectively. RULER averages 57 task–length entries at 1K–16K. All three scores are percentages. Bold indicates the better result at each model size.

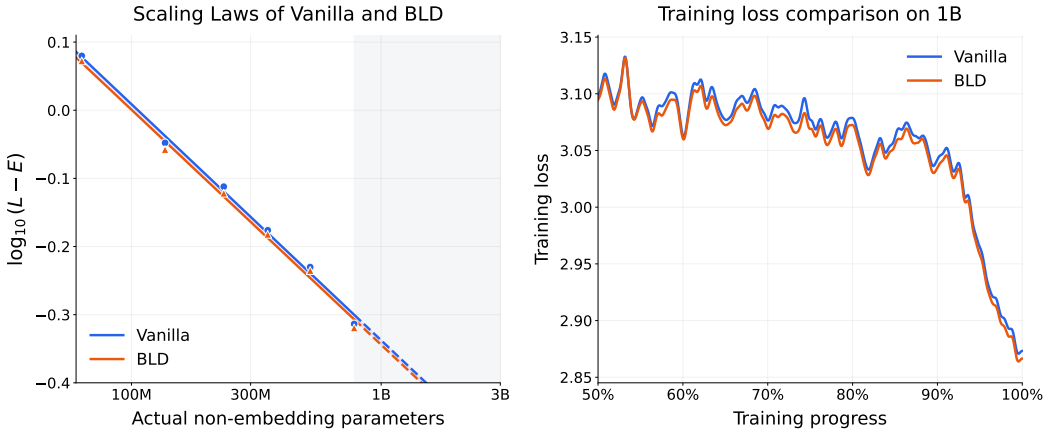

Figure 7: Left: Loss as a function of model size, fitted with power-plus-constant scaling laws. Right: The training-loss curve of the 1B vanilla and BLD models.

Context Utilization We evaluate the long-context utilization of our models as Sun et al. (2024) by computing LM loss as a function of context length. Specifically, we calculated next-token prediction loss on `Books3`,`BookCorpus2` (Gao et al., 2020) with context length of 16K. Figure 5 shows that BLD model has a lower loss especially in the later position, highlighting better context utilization abilities by BLD models. The results of all model sizes and other details are included in Appendix F.

Scaling Laws We also fit the loss as a power-law function of model size to verify that BLD is scalable to larger model scales (shown in Figure 7), and the fitted curves suggest that BLD retains its loss advantage beyond the evaluated model scales.

4.3.1 Transferability to Other RNN Architectures

We also applied BLD to two other popular recurrent mixers, Gated DeltaNet (GDN) (Yang et al., 2025b) and Gated Linear Attention (GLA) (Yang et al., 2023). These models have the same configuration as the model in Section 4.2, but with KDA layers replaced with either of these layers. Training configuration is also the same as in Section 4.2. Further details are reported in Appendix G.

Takeaways The advantages of BLD, lower training loss and better downstream task performance, are transferable to other linear recurrence architectures.

Architecture	Size	Model	LM Loss ↓	CSR ↑	Recall ↑	RULER ↑	WikiText ↓	LongPPL ↓
GDN	400M	Vanilla	3.1947	43.23	17.33	12.33	31.25	152.84
GDN	400M	BLD	**3.1624**	**43.90**	**32.05**	**25.42**	**28.69**	**11.77**
GLA	400M	Vanilla	3.2911	41.19	24.58	23.98	34.57	32.69
GLA	400M	BLD	**3.2113**	**43.20**	**30.12**	**28.54**	**30.65**	**13.26**

Table 2: Zero-shot evaluation results of BLD (ours) versus vanilla pre-training with GDN and GLA hybrid models with 3:1 recurrence-to-attention ratio.

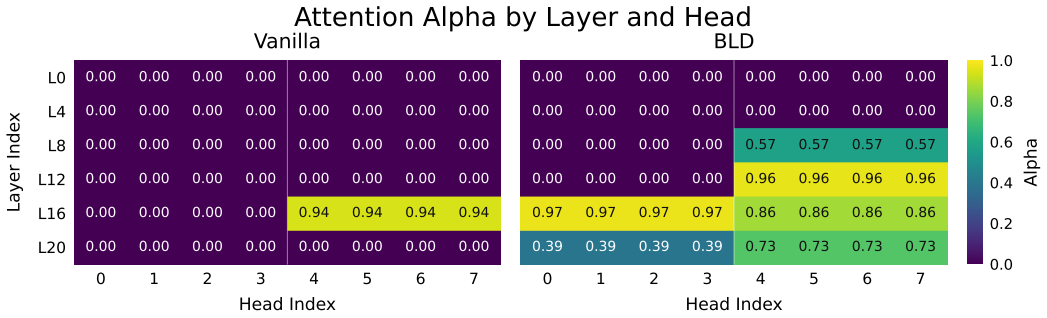

Figure 8: The alpha values of attention heads. With GQA, query heads 0–3 and 4–7 share a KV head within each group and therefore share the same KV-head-level alpha value.

4.3.2 Evidence for Labor Division Change in Attention Scores

We further provide direct evidence that BLD induces stronger functional differentiation. Specifically, following Xiao et al. (2025), we analyze 510M KDA-attention hybrid models by training *alpha values* to distinguish retrieval heads, which are responsible for long-distance information retrieval, from streaming heads, which primarily perform short-distance token mixing.

Observations Figure 8 shows the alpha values after 1000 steps training of BLD and vanilla model. Alpha measures a head's reliance on full-context attention, with larger values indicating greater reliance. The attention heads in the BLD model exhibit higher alpha values than those in the vanilla model, indicating that BLD contains more heads that attend to long-range tokens. Figure 6 reports retrieval accuracy as attention heads are progressively masked in ascending order of their alpha values. For BLD, the accuracy remains largely unchanged as low-alpha heads are masked and drops sharply only when the final three heads, which have the highest alpha values, are removed. **This suggests that these three heads play a critical role in retrieval, and we therefore identify them as retrieval heads. In contrast, the vanilla model exhibits a substantial accuracy drop only when its single highest-alpha head is masked, indicating that it contains only one retrieval head.** Taken together, these results suggest that BLD promotes a greater degree of functional specialization, with more attention heads specializing in long-range retrieval. This behavior directly aligns with the motivation underlying our method. More details are included in Appendix H.

5 Conclusion

In this work, we first identify the Competing Learning phenomenon and provide supporting experimental evidence. We then show that directly pretraining hybrid models can lead to a suboptimal division of labor among different architectural components. To address this issue, we propose BLD and validate its effectiveness through scalable experiments. By promoting a more effective division of labor, BLD consistently improves model capabilities across a wide range of dimensions.

Nevertheless, BLD represents only a preliminary approach to addressing the Competing Learning phenomenon in the training of hybrid models. We believe that a deeper understanding of the Competing Learning phenomenon will provide further insights into the division of labor within hybrid architectures and inspire the development of more advanced training strategies that better account for this phenomenon and achieve stronger performance.

AI USE STATEMENT

In this work, we used generative AI tools for research execution. We have not used generative AI tools for research ideation, and LLM data-generation is not applicable to this work. Additionally, we used generative AI tools for polishing the manuscript. We have reviewed all AI-assisted work. LLM-generated code was verified and tested for correctness by 2 authors. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

We provide the experimental settings needed to reproduce our results in the appendix. Appendix A details the synthetic retrieval task, model configuration, training procedure, and freezing interventions. Appendices B and C describe the language-model architectures, training data, token budgets, optimization settings, and parameter-freezing scope. Evaluation protocols and detailed results are provided in Appendices D and E. Appendices F and G describe the position-wise loss analysis and the GDN/GLA experiments, respectively.

We will publicly release the training and evaluation code, relevant experimental data, and model checkpoints to facilitate reproduction.

REFERENCES

Kirill Afendulev, Alexey Dontsov, Elena Tutubalina, and Anton Korznikov. What attention recalls and recurrence controls in hybrid language models. *arXiv preprint arXiv:2609.04434*, 2026.

Joshua Ainslie, James Lee-Thorp, Michiel de Jong, Yury Zemlyanskiy, Federico Lebron, and Sumit Sanghai. GQA: Training generalized multi-query transformer models from multi-head checkpoints. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 4895–4901, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.298. URL https://aclanthology.org/2023.emnlp-main.298/.

Simran Arora, Brandon Yang, Sabri Eyuboglu, Avanika Narayan, Andrew Hojel, Immanuel Trummer, and Christopher Ré. Language models enable simple systems for generating structured views of heterogeneous data lakes. *arXiv preprint arXiv:2304.09433*, 2023.

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:2402.18668*, 2024.

Aviv Bick, Eric Xing, and Albert Gu. Understanding the skill gap in recurrent language models: The role of the gather-and-aggregate mechanism. *arXiv preprint arXiv:2504.18574*, 2025.

Aviv Bick, Eric P Xing, and Albert Gu. Retrieval-aware distillation for transformer-ssm hybrids. *arXiv preprint arXiv:2602.11374*, 2026.

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. Piqa: Reasoning about physical commonsense in natural language. In *Thirty-Fourth AAAI Conference on Artificial Intelligence*, 2020.

Yingfa Chen, Zhen Leng Thai, Zihan Zhou, Zhu Zhang, Xingyu Shen, Shuo Wang, Chaojun Xiao, Xu Han, and Zhiyuan Liu. Hybrid linear attention done right: Efficient distillation and effective architectures for extremely long contexts. *arXiv preprint arXiv:2601.22156*, 2026.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. Boolq: Exploring the surprising difficulty of natural yes/no questions. In *NAACL*, 2019.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv:1803.05457v1*, 2018.

Dheeru Dua, Yizhong Wang, Pradeep Dasigi, Gabriel Stanovsky, Sameer Singh, and Matt Gardner. Drop: A reading comprehension benchmark requiring discrete reasoning over paragraphs. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 2368–2378, 2019.

Lizhe Fang, Yifei Wang, Zhaoyang Liu, Chenheng Zhang, Stefanie Jegelka, Jinyang Gao, Bolin Ding, and Yisen Wang. What is wrong with perplexity for long-context language modeling? In *International Conference on Learning Representations*, volume 2025, pp. 94541–94563, 2025.

Richard Futrell, Kyle Mahowald, and Edward Gibson. Large-scale evidence of dependency length minimization in 37 languages. *Proceedings of the National Academy of Sciences*, 112(33):10336–10341, 2015. doi: 10.1073/pnas.1502134112. URL https://www.pnas.org/doi/abs/10.1073/pnas.1502134112.

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.

GLM Team. Glm-5.3-flash: Frontier intelligence, flash cost, 2026. URL https://z.ai/blog/glm-5.3-flash.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, et al. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*, 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.

Lei Huang, Xiaocheng Feng, Weitao Ma, Yuchun Fan, Xiachong Feng, Yangfan Ye, Weihong Zhong, Yuxuan Gu, Baoxin Wang, Dayong Wu, et al. Improving contextual faithfulness of large language models via retrieval heads-induced optimization. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 16896–16913, 2025.

Luyang Huang, Shuyang Cao, Nikolaus Parulian, Heng Ji, and Lu Wang. Efficient attentions for long document summarization. In *Proceedings of the 2021 conference of the north American chapter of the association for computational linguistics: Human language technologies*, pp. 1419–1436, 2021.

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.

Keller Jordan, Yuchen Jin, Vlado Boza, Jiacheng You, Franz Cesista, Laker Newhouse, and Jeremy Bernstein. Muon: An optimizer for hidden layers in neural networks, 2024. URL https://kellerjordan.github.io/posts/muon/.

Mandar Joshi, Eunsol Choi, Daniel S Weld, and Luke Zettlemoyer. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1601–1611, 2017.

Kimi Team, Yu Zhang, Zongyu Lin, Xingcheng Yao, Jiaxi Hu, Fanqing Meng, Chengyin Liu, Xin Men, Songlin Yang, Zhiyuan Li, et al. Kimi linear: An expressive, efficient attention architecture. *arXiv preprint arXiv:2510.26692*, 2025.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, et al. Natural questions: a benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:453–466, 2019.

Jeffrey Li, Alex Fang, Georgios Smyrnis, Maor Ivgi, Matt Jordan, Samir Gadre, Hritik Bansal, Etash Guha, Sedrick Keh, Kushal Arora, Saurabh Garg, Rui Xin, Niklas Muennighoff, Reinhard Heckel, Jean Mercat, Mayee Chen, Suchin Gururangan, Mitchell Wortsman, Alon Albalak, Yonatan Bitton, Marianna Nezhurina, Amro Abbas, Cheng-Yu Hsieh, Dhruba Ghosh, Josh Gardner, Maciej Kilian, Hanlin Zhang, Rulin Shao, Sarah Pratt, Sunny Sanyal, Gabriel Ilharco, Giannis Daras, Kalyani Marathe, Aaron Gokaslan, Jieyu Zhang, Khyathi Chandu, Thao Nguyen, Igor Vasiljevic, Sham Kakade, Shuran Song, Sujay Sanghavi, Fartash Faghri, Sewoong Oh, Luke Zettlemoyer, Kyle Lo, Alaaeldin El-Nouby, Hadi Pouransari, Alexander Toshev, Stephanie Wang, Dirk Groeneveld, Luca Soldaini, Pang Wei Koh, Jenia Jitsev, Thomas Kollar, Alexandros G. Dimakis, Yair Carmon, Achal Dave, Ludwig Schmidt, and Vaishaal Shankar. Datacomp-lm: In search of the next generation of training sets for language models, 2024.

Yanhong Li, Songlin Yang, Shawn Tan, Mayank Mishra, Rameswar Panda, Jiawei Zhou, and Yoon Kim. Distilling to hybrid attention models via kl-guided layer selection. In *International Conference on Learning Representations*, volume 2026, pp. 108531–108555, 2026.

Yupu Liang, Shuang Chen, Guanwei Zhang, Shaolei Wang, and Suncong Zheng. Revealing the learning dynamics of long-context continual pre-training. *arXiv preprint arXiv:2604.02650*, 2026.

Colin Lockard, Prashant Shiralkar, and Xin Luna Dong. Openceres: When open information extraction meets the semi-structured web. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 3047–3056, 2019.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

Youmi Ma and Naoaki Okazaki. From interpretability to performance: Optimizing retrieval heads for long-context language models. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 27731–27747, 2026.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models, 2016.

NVIDIA Team. Nemotron 3 ultra: Open, efficient mixture-of-experts hybrid mamba-transformer model for agentic reasoning, 2026. URL https://arxiv.org/abs/2606.15007.

Denis Paperno, Germán Kruszewski, Angeliki Lazaridou, Ngoc-Quan Pham, Raffaella Bernardi, Sandro Pezzelle, Marco Baroni, Gemma Boleda, and Raquel Fernández. The lambada dataset: Word prediction requiring a broad discourse context. In *Proceedings of the 54th annual meeting of the association for computational linguistics (volume 1: Long papers)*, pp. 1525–1534, 2016.

Ziqing Qiao, Yinuo Xu, Chaojun Xiao, Zhou Su, Zihan Zhou, Yingfa Chen, Xiaoyue Xu, Xu Han, and Zhiyuan Liu. Rethinking the role of efficient attention in hybrid architectures. *arXiv preprint arXiv:2606.15378*, 2026.

Zhen Qin, Songlin Yang, Weixuan Sun, Xuyang Shen, Dong Li, Weigao Sun, and Yiran Zhong. Hgrn2: Gated linear rnns with state expansion. *arXiv preprint arXiv:2404.07904*, 2024.

Qwen Team. Qwen3.8-Flash-Next: A new architecture, towards ultimate cost-efficiency, August 2026. URL https://qwen.ai/blog?id=qwen3.8-flash-next.

Pranav Rajpurkar, Robin Jia, and Percy Liang. Know what you don't know: Unanswerable questions for squad. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pp. 784–789, 2018.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Winogrande: An adversarial winograd schema challenge at scale, 2019. URL https://arxiv.org/abs/1907.10641.

Noam Shazeer. Glu variants improve transformer. *arXiv preprint arXiv:2002.05202*, 2020.

Xingyu Shen, Yingfa Chen, Zhen Leng Thai, Xu Han, Zhiyuan Liu, and Maosong Sun. Statex: Enhancing rnn recall via post-training state expansion. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 21339–21353, 2026.

Yu Sun, Xinhao Li, Karan Dalal, Jiarui Xu, Arjun Vikram, Genghan Zhang, Yann Dubois, Xinlei Chen, Xiaolong Wang, Sanmi Koyejo, et al. Learning to (learn at test time): Rnns with expressive hidden states. *arXiv preprint arXiv:2407.04620*, 2024.

Team Kimi, Tongtong Bai, Yifan Bai, Yiping Bao, Jianfeng Cai, Xinyuan Cai, Peizhou Cao, Yuxuan Cao, Ziwei Chai, Y Charles, et al. Kimi k3: Open frontier intelligence. *arXiv preprint arXiv:2607.24653*, 2026.

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.

Dustin Wang, Rui-Jie Zhu, Steven Abreu, Yong Shan, Taylor Kergan, Yuqi Pan, Yuhong Chou, Zheng Li, Jibin Wu, Ge Zhang, et al. A systematic analysis of hybrid linear attention. *arXiv preprint arXiv:2507.06457*, 2025.

Wenhao Wu, Yizhong Wang, Guangxuan Xiao, Hao Peng, and Yao Fu. Retrieval head mechanistically explains long-context factuality. In *International Conference on Learning Representations*, volume 2025, pp. 62143–62156, 2025.

Guangxuan Xiao, Jiaming Tang, Jingwei Zuo, Junxian Guo, Shang Yang, Haotian Tang, Yao Fu, and Song Han. Duoattention: Efficient long-context llm inference with retrieval and streaming heads. In *International Conference on Learning Representations*, volume 2025, pp. 37228–37253, 2025.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025a.

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

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hellaswag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 2019.

Biao Zhang and Rico Sennrich. Root mean square layer normalization. *Advances in neural information processing systems*, 32, 2019.

Xinyu Zhao, Fangcong Yin, and Greg Durrett. Understanding synthetic context extension via retrieval heads. *arXiv preprint arXiv:2410.22316*, 2024.

A SPACED RETRIEVAL EXPERIMENTS

A.1 DATA AND ARCHITECTURE

We use the Spaced Retrieval task from Section 3.1:

$$\text{Input:} \quad \$\mathtt{X}_1\mathtt{X}_2\cdots\mathtt{X}_t \underbrace{\sqcup\cdots\sqcup}_{g \text{ spaces}} \$$$

$$\text{Output:} \quad \mathtt{X}_1\mathtt{X}_2\cdots\mathtt{X}_t$$

Here $t = 5$, each X_i is a digit, and $g \in \{64, 8192\}$ is the number of spaces separating the key from the retrieval query. We add 0–20 spaces before the first delimiter to vary the key's position. Training and evaluation, including the modified keys used for attribution, use disjoint keys. To compare $g = 64$ and $g = 8192$, we keep each example identical except for g. Tables 3 and 4 summarize the data, model, and optimization settings.

Table 3: Synthetic retrieval data and model configuration.

Setting	Value
Key length / g	Five digits; $g \in \{64, 8192\}$
Leading spaces	0–20
Vocabulary	12 symbols: ten digits, delimiter, and space/padding
Training / evaluation	50,000 distinct keys / 2,000 held-out pairs per g
Decoder blocks	KDA–attention–KDA–attention
Hidden / FFN width	256 / 512
Mixer heads / attention KV heads	2 / 2
KDA head dimension / value expansion	128 / 2
Short convolution width	4
KDA output gate	Enabled (sigmoid) (Kimi Team et al., 2025)
position encoding	all layers have no rotary embeddings
Embeddings	Tied input/output embeddings

A.2 OPTIMIZATION AND EVALUATION

Table 4: Synthetic training and attribution evaluation settings.

Setting	Value
Optimizer / updates	AdamW (Loshchilov & Hutter, 2017) / 1,500
Batch	32 examples per update
Learning rate	Peak 3×10^{-4}; final 10^{-5}
Schedule	100 warmup updates, followed by cosine decay
Adam coefficients	$(\beta_1, \beta_2) = (0.9, 0.95)$
Weight decay / gradient clip	0.1 / global norm 1.0
Precision	BF16
Loss mask	Cross-entropy on the five answer tokens only

Source digits, delimiters, leading spaces, and gap tokens do not contribute to the training loss.

For attribution, we select one answer digit per held-out pair, with 400 pairs assigned to each of the five answer positions. The counterfactual input changes only the corresponding digit in the source key. Each input prefix ends just before the selected answer digit, and any earlier answer digits are identical in the two inputs. We average attribution scores across these pairs, rather than performing five interventions on every pair.

A.3 BIDIRECTIONAL MIXER-OUTPUT INTERVENTION

For each input pair, we compute the retrieval ability score defined in Section 3.2 twice, once in each direction ($A \leftarrow B$ and $B \leftarrow A$), and take their average. We then average over all evaluation pairs:

$$C_l = \frac{1}{2}\mathbb{E}_{(A,B)}\left[m(\mathbf{z}^A; a, b) - m(\widetilde{\mathbf{z}}_l^{A\leftarrow B}; a, b) + m(\mathbf{z}^B; b, a) - m(\widetilde{\mathbf{z}}_l^{B\leftarrow A}; b, a)\right]. \tag{7}$$

In each direction, we replace the selected layer's mixer output at the prediction position with that from the paired input, then recompute the prediction without changing the model weights.

The score is a change in a *logit margin*. Signed values are retained. Independently measured layer effects need not add to a total model effect: they are neither disjoint contributions nor normalized shares. In the baseline and freeze/unfreeze plots, a mixer-family curve averages its two layer scores. The mixture endpoint plot instead sums the two scores within each family before averaging across seeds.

A.4 FREEZING AND UNFREEZING INTERVENTIONS

Table 5: Synthetic freezing and unfreezing protocol.

Setting	Value
Frozen parameters	Entire KDA blocks 0 and 2, including mixer, MLP, and normalization
Trainable components	Attention blocks and shared parameters
Updates completed before freezing (s)	0, 1, 10, 25, 50, 75, 100, 125, 150, 175, 200, 225, 250, 275, 300, 325, 350, 375, 400, 425, 450, 475, 500, 600
Permanent freeze	Keep KDA blocks frozen through update 1,500

For each s in Table 5, we train normally for s updates, then freeze the KDA blocks starting with update $s + 1$. Thus, $s = 0$ means freezing from the start, and the score recorded at update s is measured before freezing has any effect. Frozen blocks still participate in the forward pass and allow gradients to reach earlier trainable components.

In the unfreezing experiment, we resume KDA updates at update 501 and continue training to update 1,500. We use only $s \leq 500$, excluding $s = 600$. The $s = 500$ run has no frozen updates and serves as a continuation control; we also compare with the baseline that never freezes KDA.

Each run continues from the saved training state, preserving the optimizer, learning-rate schedule, random state, and data order. Unfreezing does not reset these states or restart warmup.

A.5 DIFFERENT TRAINING DATA PREFER DIFFERENT ARCHITECTURES

We vary the proportion of short-distance examples to study how the training distribution affects retrieval learning, as in Section 3.3.2. Let p denote the fraction of examples with $g = 64$; the remaining fraction $1 - p$ uses $g = 8192$. Thus, $p = 0$ uses only long-distance examples, whereas $p = 1$ uses only short-distance examples.

Table 6: Training-data proportions and settings for the data-distribution experiment.

Setting	Value
Short-distance fraction p	0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1
Training distances	$g = 64$ and $g = 8192$
Optimizer updates	1,500
Examples per update	32
Evaluation distances	$g = 64$ and $g = 8192$

Within each optimizer update, short- and long-distance examples are processed separately. Their gradients are weighted by the number of examples and accumulated before a single parameter update. We use at least three random seeds for each setting and report the average final-checkpoint retrieval scores in Figure 3.

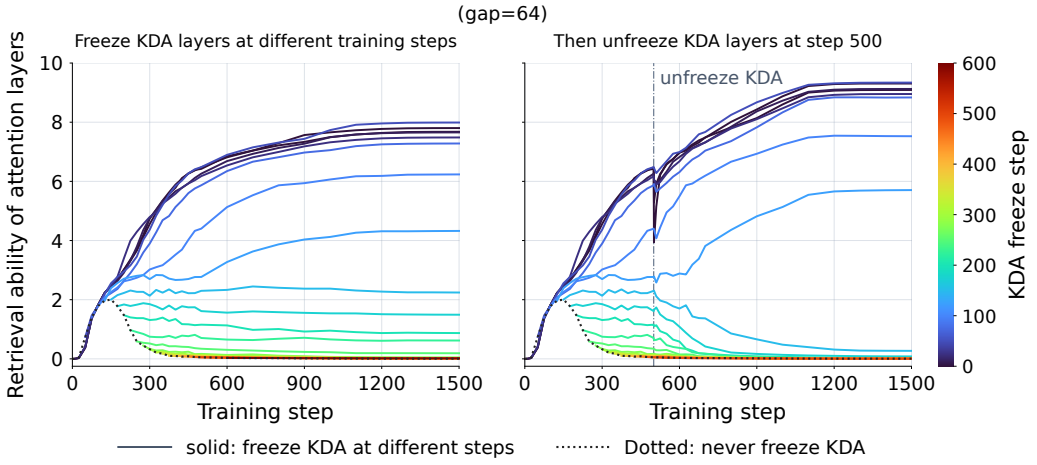

Figure 9: Attention retrieval ability under KDA freezing and unfreezing at $g = 64$, using the same setup as Figure 4.

B Model Configurations

B.1 Parameter accounting and model identity

Table 7: KDA hybrid model configurations. Parameter counts are rounded to the nearest million; tied embeddings are counted once. Query and key/value head counts are shown separately for softmax attention.

Model size	Total params (M)	Non-embedding params (M)	Decoder blocks	Hidden width	Feed-forward width	KDA heads	Attention heads query / key-value
180M	181	63	8	768	1792	6	12/3
250M	254	136	16	768	2048	6	6/2
390M	391	234	16	1024	2688	8	8/2
510M	508	351	24	1024	2688	8	8/2
750M	755	520	16	1536	4096	12	12/3
1B	1015	780	24	1536	4096	12	12/3

We label models by their total parameter counts, including token embeddings, throughout this appendix: 180M, 250M, 390M, 510M, 750M, and 1B. Labels are rounded to the nearest 10M. Table 7 reports total and nonembedding parameter counts rounded to the nearest million. Training token-to-parameter ratios and scaling fits use the unrounded nonembedding parameters.

All models repeatedly stack the four-block sequence Attention–KDA–KDA–KDA, giving a 3:1 ratio of KDA to softmax-attention blocks at every model size.

The size-dependent settings are listed in Table 7.

C Training Details

C.1 Data and freezing during early pretraining

Throughout pretraining, we use DCLM (Li et al., 2024) with 16,384-token sequences. Texts are tokenized, terminated with EOS, concatenated, and split into fixed-length sequences; incomplete remainders of preprocessing batches are discarded. Table 9 contrasts the early and remaining portions of pretraining. Vanilla and BLD use the same data views, token budgets, and optimizer schedules at each model size.

Table 10 summarizes the token budgets.

Table 8: Architecture settings shared across KDA language models.

Setting	Value
Tokenizer / vocabulary	Qwen3 (Yang et al., 2025a) / 153,000 entries
Repeating block pattern	Attention–KDA–KDA–KDA; KDA-to-attention ratio 3:1
Embeddings / FFN	Tied input/output embeddings / SwiGLU (Shazeer, 2020)
Normalization	RMSNorm (Zhang & Sennrich, 2019), $\epsilon = 10^{-6}$
KDA head dimension	128
KDA value expansion / convolution width	2 / 4
KDA output gate	Enabled (sigmoid) (Kimi Team et al., 2025)
Softmax attention	Grouped-query attention (Ainslie et al., 2023); no projection bias or sliding window
Positional embeddings	Rotary embeddings disabled

Table 9: Freezing scope during early and remaining pretraining.

Setting	Early pretraining	Remaining pretraining
DCLM data view	Global shard 01	Global shards 02–10
Token fraction	Approximately $1/11$ (9.09%)	Approximately $10/11$ (90.91%)
Vanilla parameters	All trainable	All trainable
BLD: frozen components	Entire KDA blocks: mixer, MLP, and normalization	None
BLD: trainable components	Attention blocks, tied embeddings, and final normalization	All components
Weight / optimizer initialization	Train from scratch	Continue from current weights; reset optimizer and schedule

Table 10: Training token budgets, rounded for readability. M and B denote million and billion tokens.

Model size	Early pretraining tokens	Remaining pretraining tokens
180M	120M	1.20B
250M	280M	2.80B
390M	480M	4.80B
510M	720M	7.20B
750M	1.07B	10.65B
1B	1.60B	16.00B

Early pretraining uses approximately $2.05N$ tokens, followed by approximately $20.5N$ tokens in the remaining pretraining, where N is the number of nonembedding parameters, following the Chinchilla scaling principle (Hoffmann et al., 2022).

C.2 Optimization, batching, and restarts

Table 11: Optimization settings shared by Vanilla and BLD.

Setting	Early pretraining	Remaining pretraining
Optimizer	AdamW (Loshchilov & Hutter, 2017)	Muon (Jordan et al., 2024) with auxiliary AdamW
Peak learning rate	10^{-4}	Muon: 0.02; AdamW: 0.001
Minimum learning rate	10^{-4} (no terminal decay)	Muon: 0.001; AdamW: 5×10^{-5}
Schedule	Linear warmup, then constant	Linear warmup, constant interval, then cosine decay
Adam coefficients	$(0.9, 0.95)$	$(0.9, 0.95)$
Muon momentum / iterations	Not applicable	0.95 / 5 Newton–Schulz iterations
Weight decay / clip	0.1 / global norm 1.0	0.1 / global norm 1.0
Precision / seed	BF16 / 0	BF16 / 0
Global batch (sequences)	16	32
Tokens per update	262,144	524,288
Microbatch per GPU	1 sequence	2 sequences

Table 12: Size-dependent schedules and batching. Warmup and decay durations are in optimizer updates; accumulation is unchanged throughout pretraining.

Size	Early warmup	Remaining warmup	Final decay	GPUs	Accum.
180M	23	229	229	1	16
250M	53	534	534	1	16
390M	92	916	916	2	8
510M	137	1,373	1,373	2	8
750M	198	2,032	2,032	2	8
1B	305	3,052	3,052	8	2

Muon updates eligible hidden matrix weights; embeddings, the output head, and remaining parameters use auxiliary AdamW. All optimizer groups share the same multiplicative schedule. Both methods reset optimizer and schedule state when the early portion ends.

D Evaluation Details

D.1 Checkpoint and metric selection

We evaluate the final pretraining checkpoint without fine-tuning, using zero demonstrations for the downstream language-model evaluation tasks. The decoding strategy for generation tasks is greedy decoding.

Language-model loss. We report the mean training loss over the final 100 optimizer updates.

Common-sense reasoning. CSR is the macro average of seven tasks: ARC-Easy and ARC-Challenge (Clark et al., 2018), HellaSwag (Zellers et al., 2019), and PIQA (Bisk et al., 2020) use normalized accuracy; BoolQ (Clark et al., 2019), WinoGrande (Sakaguchi et al., 2019), and LAMBADA (Paperno et al., 2016) use accuracy.

Recall. We use the Based evaluation suite (Arora et al., 2024) and average the `contains` scores over SWDE (Lockard et al., 2019), FDA (Arora et al., 2023), SQuAD (Rajpurkar et al., 2018), TriviaQA (Joshi et al., 2017), DROP (Dua et al., 2019), and Natural Questions (Kwiatkowski et al., 2019) with a 2,048-token context.

RULER. We evaluate RULER (Hsieh et al., 2024) at 1K, 2K, 4K, 8K, and 16K tokens, with 100 examples per task–length combination. These lengths include 10, 11, 12, 12, and 12 tasks, respectively; QA-1 is included from 2K and QA-2 from 4K. The overall score averages all 57 task–length combinations. CSR, Recall, and RULER scores are reported as percentages.

WikiText-2. We report word-level perplexity on WikiText-2 (Merity et al., 2016) using the evaluator's `word_perplexity` metric.

LongPPL. Following LongPPL (Fang et al., 2025), we report pooled perplexity over tokens selected using fixed reference-model annotations, as detailed below.

D.2 LongPPL reproduction

Table 13: LongPPL reproduction settings.

Setting	Value
Dataset / sample	GovReport (Huang et al., 2021) / 50 documents
Sampling seed / token limit	42 / 32,768 per document
Input formatting	Raw text; no chat template or additional BOS
Reference model	Qwen3-8B (Yang et al., 2025a); fixed annotations across compared models
Short context / stride	$K = 4096$ / $d = 1024$
Selected targets	3,920 tokens across 49 documents
Alignment	Same texts and target positions; tokenizer alignment checked

For token i, let ℓ_i^{long} and ℓ_i^{short} be its reference log probabilities under the document prefix and a short-context window. The selected-token indicator is

$$M_i = \mathbb{1}\{\ell_i^{\text{long}} - \ell_i^{\text{short}} > 2\}\mathbb{1}\{\ell_i^{\text{long}} > -2\}. \tag{8}$$

Target tokens within each stride interval share the same left context boundary, so their context lengths range from K to $K + d - 1$ preceding tokens. Table 13 lists the fixed sample and reference settings. The reported metric is

$$\text{LongPPL}(\theta) = \exp\left[-\frac{\sum_i M_i \log p_\theta(x_i \mid x_{<i})}{\sum_i M_i}\right]. \tag{9}$$

We pool selected-token negative log likelihoods before exponentiation; we do not average document perplexities. The document without selected tokens does not enter this denominator.

E Detailed Results and Additional Analyses

E.1 Downstream task results

Tables 14 and 15 report individual CSR and Recall scores. The CSR mean here covers the seven tasks listed above, including LAMBADA, and uses the same aggregation as the main tables. Table 16 breaks RULER down by context length; Tables 17–19 provide task-wise averages. All entries use the same final checkpoints as Tables 1 and 2.

Figure 10 compares Vanilla and BLD at each context length across the six KDA model sizes.

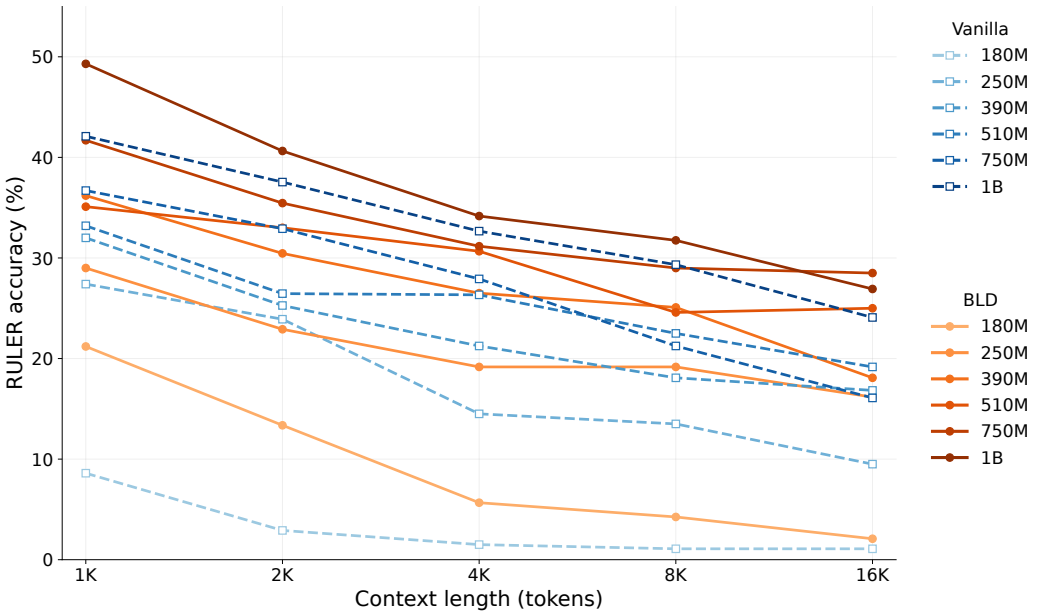

Figure 10: RULER accuracy by context length for six KDA model sizes. Blue dashed lines denote Vanilla; orange solid lines denote BLD. Darker shades indicate larger models. Each point averages the tasks evaluated at that length.

Table 14: Seven-task CSR results (percent). LAMBADA reports accuracy. Means use unrounded scores from all seven tasks. GDN and GLA denote the transfer configurations in Appendix G. Unless otherwise specified, all models are KDA-based hybrid models.

Size / mixer	Method	ARC-E	ARC-C	Hella.	PIQA	BoolQ	Wino.	LAMBADA	Mean
180M	Vanilla	36.36	22.44	27.38	58.27	58.87	51.07	9.94	37.76
180M	BLD	35.65	22.44	27.64	59.30	55.84	49.96	11.59	37.49
250M	Vanilla	38.43	23.98	29.80	58.98	45.05	51.38	20.57	38.31
250M	BLD	41.16	23.72	30.33	62.35	60.58	48.70	23.56	41.49
390M	Vanilla	43.73	24.74	34.31	64.20	47.68	49.25	29.46	41.91
390M	BLD	43.43	24.57	35.35	65.40	58.75	50.75	31.48	44.25
510M	Vanilla	46.51	26.11	36.93	66.27	58.01	50.43	29.34	44.80
510M	BLD	47.31	25.17	37.42	66.97	55.69	51.22	27.61	44.49
750M	Vanilla	48.06	25.51	40.68	67.19	53.76	51.14	35.26	45.94
750M	BLD	48.91	25.26	40.93	68.50	53.73	51.85	37.12	46.61
1B	Vanilla	51.26	27.65	45.11	69.75	53.82	52.96	40.25	48.69
1B	BLD	51.09	27.90	45.91	69.10	55.69	55.25	41.22	49.45
GDN (400M)	Vanilla	44.44	24.40	34.41	64.53	56.73	51.07	27.03	43.23
GDN (400M)	BLD	44.23	24.66	34.77	64.91	58.69	51.54	28.51	43.90
GLA (400M)	Vanilla	43.43	24.06	32.43	64.25	52.23	49.80	22.10	41.19
GLA (400M)	BLD	44.53	24.74	33.32	63.44	59.48	50.36	26.57	43.20

Table 15: Recall results using the contains metric (percent). Means use unrounded scores. Unless otherwise specified, all models are KDA-based hybrid models.

Size / mixer	Method	SWDE	FDA	SQuAD	TriviaQA	DROP	NQ	Mean
180M	Vanilla	9.09	0.54	13.40	21.74	13.08	4.50	10.39
180M	BLD	13.95	13.43	14.68	23.52	14.23	4.91	14.12
250M	Vanilla	27.90	17.15	21.88	33.12	19.45	9.72	21.54
250M	BLD	29.79	30.85	25.00	35.25	18.93	12.13	25.33
390M	Vanilla	34.74	26.04	27.38	38.68	20.17	13.34	26.73
390M	BLD	38.34	40.83	30.09	41.35	19.84	17.14	31.27
510M	Vanilla	36.72	34.21	28.92	43.36	21.99	17.77	30.50
510M	BLD	44.46	46.55	30.06	44.25	23.38	19.51	34.70
750M	Vanilla	40.59	41.47	30.90	47.45	21.66	19.04	33.52
750M	BLD	49.23	50.54	33.31	49.41	21.56	21.03	37.52
1B	Vanilla	44.55	42.74	33.91	52.19	24.77	20.84	36.50
1B	BLD	48.15	43.83	36.76	53.08	24.20	22.81	38.14
GDN	Vanilla	17.55	3.27	20.95	36.43	16.29	9.50	17.33
GDN	BLD	43.74	40.74	30.06	41.88	19.50	16.34	32.05
GLA	Vanilla	36.18	27.59	22.52	33.95	17.63	9.60	24.58
GLA	BLD	40.86	41.20	27.78	38.15	17.97	14.73	30.12

Table 16: RULER scores by context length (percent). Parentheses give task counts; All averages the 57 task–length entries. Unless otherwise specified, all models are KDA-based hybrid models.

Size / mixer	Method	1K (10)	2K (11)	4K (12)	8K (12)	16K (12)	All (57)
180M	Vanilla	8.60	2.91	1.50	1.08	1.08	2.84
180M	BLD	21.20	13.36	5.67	4.25	2.08	8.82
250M	Vanilla	27.40	23.91	14.50	13.50	9.50	17.32
250M	BLD	29.00	22.91	19.17	19.17	16.17	20.98
390M	Vanilla	32.00	25.27	21.25	18.08	16.83	22.32
390M	BLD	36.20	30.45	26.50	25.08	18.08	26.89
510M	Vanilla	33.20	26.45	26.33	22.50	19.17	25.25
510M	BLD	35.10	33.00	30.67	24.58	25.00	29.42
750M	Vanilla	36.70	32.91	27.92	21.25	16.08	26.53
750M	BLD	41.70	35.45	31.17	29.00	28.50	32.82
1B	Vanilla	42.10	37.55	32.67	29.33	24.08	32.75
1B	BLD	49.30	40.64	34.17	31.75	26.92	36.04
GDN (400M)	Vanilla	18.60	13.27	11.33	10.17	9.42	12.33
GDN (400M)	BLD	32.30	26.45	23.83	24.75	21.00	25.42
GLA (400M)	Vanilla	29.00	26.55	23.42	25.08	16.92	23.98
GLA (400M)	BLD	34.30	30.64	28.58	27.58	22.75	28.54

Table 17: Task-wise RULER scores averaged over evaluated lengths (percent). V: Vanilla; B: BLD. QA-1 averages four lengths, QA-2 three, and other tasks five.

Task	180M V	180M B	250M V	250M B	390M V	390M B
Single-1	8.20	34.80	96.20	100.00	100.00	98.40
Single-2	8.60	40.40	41.00	31.40	62.80	72.80
Single-3	0.00	5.40	28.60	69.00	49.00	78.60
Multi-key-1	6.60	11.20	15.80	20.80	26.40	32.40
Multi-key-2	0.00	0.00	0.20	0.40	0.80	3.20
Multi-key-3	0.00	0.00	0.00	1.60	0.80	2.00
Multi-value	0.00	0.00	0.00	0.00	0.00	0.00
Multi-query	0.00	0.00	0.00	0.00	0.00	0.00
Variable tracking	0.00	0.00	0.00	0.00	0.00	0.00
Frequent words	0.00	0.00	0.20	0.00	0.00	3.20
QA-1	4.50	4.00	9.50	11.00	10.50	9.75
QA-2	9.00	9.33	13.00	12.00	10.33	13.67

Table 18: Task-wise RULER scores averaged over evaluated lengths (percent). V: Vanilla; B: BLD. QA-1 averages four lengths, QA-2 three, and other tasks five.

Task	510M V	510M B	750M V	750M B	1B V	1B B
Single-1	100.00	98.80	99.60	100.00	99.80	100.00
Single-2	67.80	99.00	79.60	90.40	83.00	92.80
Single-3	73.80	60.20	60.60	81.40	71.60	77.00
Multi-key-1	26.60	48.40	39.20	61.00	61.80	58.40
Multi-key-2	1.00	2.80	2.80	5.40	26.60	30.80
Multi-key-3	1.60	3.80	2.20	6.60	2.60	6.60
Multi-value	0.00	0.00	0.00	4.20	0.40	6.80
Multi-query	0.00	0.00	0.00	0.40	0.40	1.20
Variable tracking	0.00	0.00	0.00	0.00	0.00	0.00
Frequent words	0.40	2.60	0.20	0.60	0.40	9.40
QA-1	10.25	13.25	14.25	17.75	19.50	24.50
QA-2	14.00	15.33	11.33	16.67	18.67	13.67

E.2 Scaling fits

We use actual nonembedding parameter counts N as the model-size variable. A power-plus-constant fit with a shared floor and method-specific amplitudes and exponents has the form

$$L_m(N) = E + A_m(N/10^9)^{-\alpha_m}, \qquad m \in \{\text{Vanilla}, \text{BLD}\}. \tag{10}$$

The archived fitting procedure minimizes unweighted squared loss residuals jointly over both methods, with nonnegative floor/amplitudes and positive exponents. It examines the final-100-update average.

Table 20 reports the fitting using the same checkpoints as Table 1. Figure 7 uses the final-100-update fit. Its right panel shows the 1B model's training loss over 50–100% of the total pretraining token budget, smoothed for display by a Gaussian filter with standard deviation 100 updates in the remaining portion and reflected boundaries.

F Position-wise Language-Modeling Loss

Table 21 describes the diagnostic underlying Figure 5. Figure 11 shows the results of the same diagnostic across all model sizes.

G Transfer to GDN and GLA

Table 22 reports the GDN (Yang et al., 2025b) and GLA (Yang et al., 2023) replacement architectures. The 400M designation in Table 2 identifies the reference KDA configuration. They match KDA in depth, width, and token budget. Optimization and full-block freezing/unfreezing follow the 390M KDA configuration. Data budgets remain fixed rather than being rescaled to each replacement's parameter count. Detailed downstream scores appear in Tables 14–16.

Table 19: Task-wise RULER scores averaged over evaluated lengths (percent). V: Vanilla; B: BLD. QA-1 averages four lengths, QA-2 three, and other tasks five.

Task	GDN V	GDN B	GLA V	GLA B
Single-1	99.60	99.60	94.80	99.80
Single-2	18.20	86.00	84.80	93.40
Single-3	1.00	44.20	55.00	79.20
Multi-key-1	13.20	38.00	27.20	33.40
Multi-key-2	0.00	1.20	0.60	2.60
Multi-key-3	0.00	2.60	1.00	1.60
Multi-value	0.00	0.00	0.00	0.00
Multi-query	0.00	0.00	0.00	0.00
Variable tracking	0.00	0.00	0.00	0.00
Frequent words	0.40	0.20	0.00	0.20
QA-1	4.00	11.75	4.50	7.50
QA-2	8.33	14.33	10.67	15.33

Table 20: Shared-floor fits for the original-checkpoint six-size series. RMSE columns are Vanilla/BLD.

Loss window	E	A_V	α_V	A_B	α_B	RMSE V/B
Final 100	2.38867	0.45960	0.34649	0.45266	0.34502	0.01321/0.01424

H Attention-Head Identification and Retrieval Evaluation

Head identification. We apply a DuoAttention-style identification procedure to the final 510M Vanilla and BLD checkpoints. Each model has six attention layers, with two KV heads and eight query heads per layer. We learn one gate $\alpha_{\ell,g} \in [0,1]$ per KV head (12 gates in total), shared by its four query heads. Before the attention output projection, each head interpolates between full causal and streaming attention:

$$\widetilde{\mathbf{o}}_{\ell,h} = \alpha_{\ell,g(h)}\mathbf{o}^{\text{full}}_{\ell,h} + (1-\alpha_{\ell,g(h)})\mathbf{o}^{\text{stream}}_{\ell,h}. \tag{11}$$

Streaming attention retains the first 128 tokens and the most recent 256 tokens, subject to causality. All pretrained parameters are frozen; KDA layers retain their original computation without gates.

Each model serves as its own full-attention teacher. We optimize the gates using mean squared error between teacher and student final hidden states at reference-answer token positions, plus $0.05\,\mathrm{mean}(|\alpha|)$. Both branches receive reference answers during identification; this is hidden-state distillation, not retrieval fine-tuning of the base model. Gates start at one and are clamped to $[0,1]$ after each update. Table 23 summarizes the main settings.

Masking and evaluation. Within each model, we rank all 12 KV heads by increasing learned alpha and progressively replace full attention with streaming attention, starting with the lowest-alpha heads. Remaining heads use full attention; the continuous alpha values are used only for ranking. No further training is performed. Thus, a "masked head" denotes a KV head whose four query heads use restricted context, rather than a head with zeroed output. The evaluated budgets are $0, 1, 2, 4, 5, 6, 7, 8, 10, 11, 12$ KV heads, obtained by rounding masking ratios of $0\%, 10\%, \ldots, 100\%$.

Table 21: Position-wise language-modeling diagnostic settings.

Setting	Value
Dataset	First 50 nonempty documents from each local Books3 and BookCorpus2 cache from The Pile (Gao et al., 2020) (100 total)
Input processing	Independent documents; no added special tokens; cached token IDs and checked tokenization
Input / prediction limit	16,384 input tokens / 16,384 predictions per document
Execution	Full decoder prefix; chunked output projection; original position settings
Precision	BF16 model execution; FP32 logits

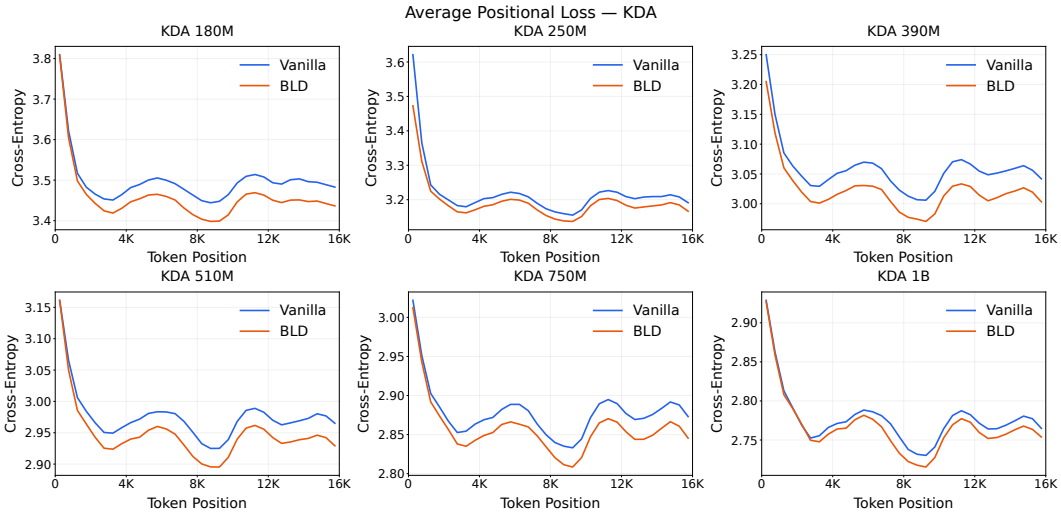

Figure 11: Average loss of all models at each position across all examples.

Table 22: GDN and GLA transfer configurations. Parameter and token counts are rounded; M and B denote million and billion.

Setting	GDN	GLA
Blocks / hidden / FFN width	16 / 1,024 / 2,688	16 / 1,024 / 2,688
Attention block indices	0, 4, 8, 12	0, 4, 8, 12
Attention Q / KV heads	8 / 2	8 / 2
Recurrent blocks / heads	12 / 8	12 / 8
Recurrent dimensions	Head dimension 128	Key expansion 1
Value expansion	2	2
Convolution width / output gate	4 / enabled	4 / enabled
Rotary embeddings	Disabled	Disabled
Total parameters	400M	401M
Nonembedding parameters	244M	244M
Early / remaining updates	1,831 / 9,155	1,831 / 9,155
Early pretraining tokens	480M	480M
Remaining pretraining tokens	4.80B	4.80B

Table 23: Settings for attention-head identification.

Setting	Value
Optimization steps / batch size	1,000 / 1 sequence
Optimizer	AdamW, no weight decay
Peak / minimum learning rate	0.02 / 0.002
Schedule	200-step warmup and final 200-step decay
Sequence length	16,384 tokens
Passkeys per sequence	10, each containing 32 words
Background / insertion depth	BookSum / 5–95%

We use the same separately generated set of 64 documents and 10 passkeys per document for both models and every budget (640 queries per condition). Each passkey contains 32 random NATO-alphabet words embedded in BookSum text. The 16K length includes the document and reference question–answer pairs. At evaluation, each question is answered independently from the document cache using greedy decoding, without reference answers or earlier question–answer pairs. The plotted accuracy is the percentage of queries whose *first generated token* matches the reference.