Understanding and Enhancing Mamba-Transformer Hybrids for Memory Recall and Language Modeling

Anonymous ACL submission

Abstract

Hybrid models that combine state space models (SSMs) with attention mechanisms have shown strong performance by leveraging the efficiency of SSMs and the high recall ability of attention. However, the architectural design choices behind these hybrid models remain insufficiently understood. In this work, we analyze hybrid architectures through the lens of memory utilization and overall performance, and propose a complementary method to further enhance their effectiveness. We first examine the distinction between sequential and parallel integration of SSM and attention layers. Our analysis reveals several interesting findings, including that sequential hybrids perform better on shorter contexts, whereas parallel hybrids are more effective for longer contexts. We also introduce a data-centric approach of continually training on datasets augmented with paraphrases, which further enhances recall while preserving other capabilities. It generalizes well across different base models and outperforms architectural modifications aimed at enhancing recall. Our findings provide a deeper understanding of hybrid SSM-attention models and offer practical guidance for designing architectures tailored to various use cases[1].

1 Introduction

Recent advances in state-space models (SSMs), such as Mamba (Gu and Dao, 2023), have shown strong performance in language modeling, particularly in long-context tasks, while offering significantly greater efficiency than traditional Transformer (Vaswani et al., 2017) architectures (Dao and Gu, 2024; Waleffe et al., 2024; Zuo et al., 2024). However, unlike Transformers, which maintain a dynamically growing key-value (KV) cache to attend to all previous tokens, SSMs compress past information into a fixed-size hidden state, limiting their ability to model long-term dependencies and recall distant context (Park et al., 2024; Glorioso et al., 2024). To address this, recent work has explored *hybrid architectures* (Dong et al., 2024; Ren et al., 2024; Park et al., 2024) that integrate attention with SSMs, aiming to leverage the strengths of both: combining the expressive, high capacity memory of attention with the efficiency of SSM computation.

Despite promising results, there remains a limited understanding of how different architectural design choices affect performance in these hybrid models, and what specific roles SSM and attention components play. In this work, we aim to fill this gap by systematically analyzing the following three research questions: (*RQ1*) **Aggregation Strategies**: How do different ways of combining SSMs and attention affect performance and efficiency? (*RQ2*) **Component Roles**: What are the respective contributions and characteristics of SSMs and attention layers in hybrid models? (*RQ3*) **Data-Centric Enhancements**: Can performance be further improved through data-centric methods, beyond architectural design alone?

To investigate the first two questions (*RQ1*, *RQ2*), we conduct extensive pretraining experiments on 17 models spanning pure SSMs, Transformer, and hybrid variants (Figure 1). Prior work often uses inconsistent training and evaluation setups, making fair comparison difficult. We therefore design a unified experimental setup that standardizes training and evaluation, enabling a controlled analysis of individual components and architectural choices. All models share the same configurations, differing only in their core block design (SSM or attention). We evaluate them across three axes: language modeling, commonsense reasoning, and memory recall. Our analysis shows a strong correlation between long-context language modeling and commonsense reasoning, but weaker

[1]We will make our code and checkpoints publicly available.

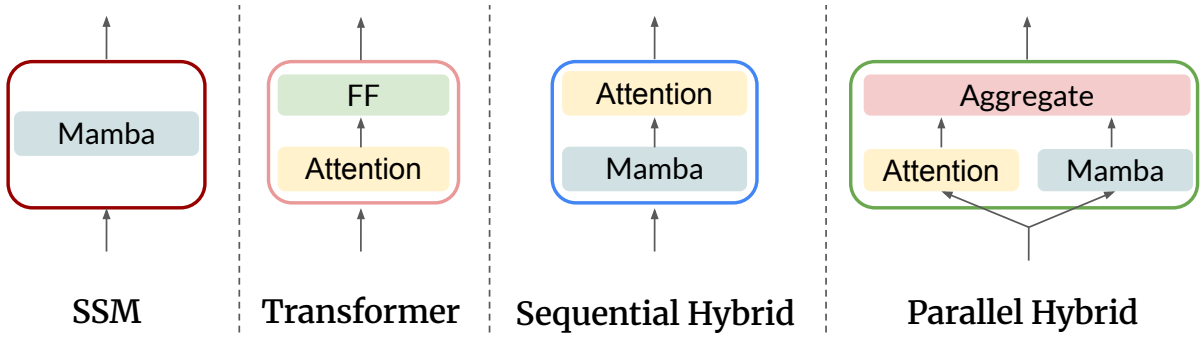

Figure 1: Comparison of different architectural designs: SSM, Transformer, Sequential Hybrid, and Parallel Hybrid. Each architecture consists of stacked *blocks* that incorporate Mamba and Attention layers. The key difference lies in how these layers are arranged: SSM uses only Mamba layers, Transformer uses only Attention layers, while the hybrid models combine both. Sequential Hybrid stacks Mamba and Attention layers within each block, whereas Parallel Hybrid applies them in parallel and aggregates their outputs. Feedforward (FF) layers are omitted in the hybrid models for clarity, as it varies by design.

links to memory recall. These results suggest that focusing solely on language modeling or reasoning benchmarks, as in prior work (Glorioso et al., 2024; Lieber et al., 2024; Ren et al., 2024), may miss critical aspects of memory performance. *Our study fills this gap by providing a comprehensive and standardized evaluation.*

Using our unified evaluation, we analyze *how aggregation strategies (sequential or parallel) affect performance and the roles of SSM and attention components*. Sequential hybrids, where one component processes input before the other, excel on short-context tasks because aligned representation spaces promote stable training. However, this alignment can limit expressiveness. In contrast, parallel hybrids keep separate embedding spaces and fuse outputs later, enabling greater representational diversity and stronger long-context performance. Among them, the parallel variant with a merge-attention layer, which attends over the outputs of the Mamba and the attention layers to produce a fused representation, achieves the strongest overall results.

Beyond architecture, we explore a *data-centric approach* to improve memory recall (*RQ3*). While previous works often finetune models on synthetic tasks like Needle-in-a-Haystack (NIAH) (Kamradt, 2023), which boosts recall but often harms performance on other metrics. To mitigate this, we show that continued training with paraphrased sentences, drawn from a distribution similar to the pretraining data, enhances recall with minimum or no degradation in commonsense reasoning. Compared to other datasets such as UltraChat (Ding et al., 2023), Based (Arora et al., 2024), or NIAH, this strategy achieves the best trade-off. Notably, it outperforms architectural methods aimed at enhancing recall, such as DeciMamba (Ben-Kish et al., 2024) (+12.7 avg), and generalizes well across a range of base models, scaling up to 2.8B parameter model.

2 Preliminary

In this section, we share details of how the Mamba layer from recent SSM models and the Attention layer in the Transformer differ, an overview of prior works on hybrid models, and outline our experimental architectures.

Mamba and Attention layers Both Mamba and attention layers transform an input sequence into an output sequence using a transformation matrix, but differ in how they process inputs. Mamba layers update a recurrent hidden state sequentially, incorporating one token at a time as a compressed summary of past inputs. In contrast, attention layers process the entire sequence simultaneously, attending to all preceding tokens to model dependencies. These approaches involve trade-offs: Mamba offers linear-time computation but may struggle with long-range dependencies, while attention layers capture such dependencies more effectively at the cost of quadratic time and memory. See Appendix A.1 for details and equations.

Hybrid Models To leverage the strengths of SSMs and attention, recent works have proposed hybrid architecture that integrate both components (Dong et al., 2024; Ren et al., 2024; Park et al., 2024). These models outperforms non-hybrid models, especially in long-context language modeling compared to attention-only models and recall

performance compared to pure SSMs.

Recent hybrid models vary along four design axes: (1) SSM layer type: Mamba is the most common (Gu and Dao, 2023; Ren et al., 2024; Dong et al., 2024; Glorioso et al., 2024), though alternatives like DeltaNet have also been effective (Yang et al., 2025). (2) Layer ratio: A 1:1 SSM-to-attention ratio is typical (Dong et al., 2024; Ren et al., 2024), though some prefer more SSM layers for efficiency (Glorioso et al., 2024; Lieber et al., 2024). (3) Attention type: To retain efficiency, many use SWA[2](Ren et al., 2024; Yang et al., 2025), combine SWA with full attention(Dong et al., 2024), or use full attention alone (Glorioso et al., 2024). (4) Integration strategy: Sequential fusion is most common (Park et al., 2024; Ren et al., 2024; Yang et al., 2025), but parallel fusion is also explored (Dong et al., 2024).

In this work, as our focus is on understanding affect of how to combine SSMs with attention layers and analyzing the role of each components in hyrid model performance, we focus on the fourth axes and keep other design choices fixed based on the recent strong baselines (Ren et al., 2024; Dong et al., 2024; Yang et al., 2025): (1) using Mamba as the SSM component, (2) a 1:1 ratio of attention to SSM layers, and (3) using SWA (Beltagy et al., 2020) as attention layer. See Appendix B for more related works.

Architectural Designs of Hybrid Blocks To analyze various hybrid model configurations, we design a set of *hybrid models*, each combining Mamba and Attention layers. These blocks are stacked to build the full model (Figure 1). Our designs vary along two main axes: (1) the integration strategy and (2) the placement of feed-forward (FF) layers. For integration, we explore: **sequential hybrid** where one layer's output feeds into the other, with two variants (Mamba → SWA and SWA → Mamba) and **parallel fusion** where both layers receive the same input, and their outputs are aggregated using one of several methods (simple averaging (Dong et al., 2024), a trainable projection layer (Behrouz et al., 2024), or a trainable merge-attention layer). Also, given that FF layers play an important role in Transformer models (Geva et al., 2020; Meng et al., 2022), we also experiment with the effect of different FF placements.

[2]SWA restricts attention to a fixed-size window around each token, improving scalability over full attention.

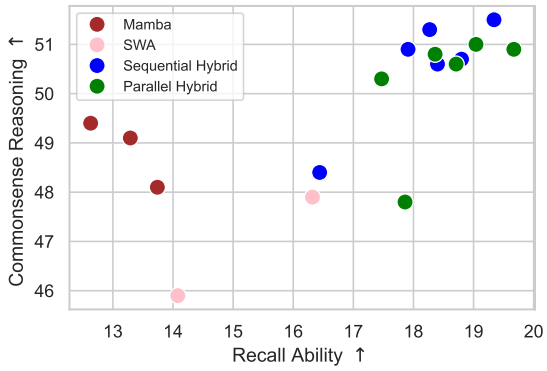

Figure 2: Comparison of different model architectures on Commonsense Reasoning (y-axis) vs. Recall Ability (x-axis). Commonsense Reasoning and Recall Ability are measured using answer accuracy. The models compared included Mamba-only, SWA-only, Hybrid (Sequential), and Hybrid (Parallel). For details of each model, see Figure 10 in Appendix C.1.

3 Designing a Unified Experimental Setup

While various works have proposed and demonstrated the effectiveness of hybrid models, their results are often difficult to compare to each other due to differences in training procedures, evaluation metrics, and the absence of released checkpoints. To enable fair and comprehensive analysis, in this section, we introduce a unified experimental setup to re-evaluate multiple models within this consistent framework of training (Section 3.1) and evaluation (Section 3.2). We observe that some prior works often overlook key metrics, which can obscure a model's overall performance, underscoring the need for extensive evaluation over multiple axes to understand model performance.

3.1 Training

We follow widely adopted training setups from recent works, primarily based on Ren et al. (2024), which provides detailed implementation code. All models are trained from scratch on 100B tokens from the SlimPajama dataset (Soboleva et al., 2023). Model sizes are kept consistent across architectural variants: approximately 430M parameters for base models and 1.3B for larger ones. All models use the same hyperparameters: batch size of 512, sequence length of 4K, learning rate of 4e-4, weight decay of 0.1, window size of 2k for SWA, and the AdamW optimizer (Loshchilov and Hutter, 2017).

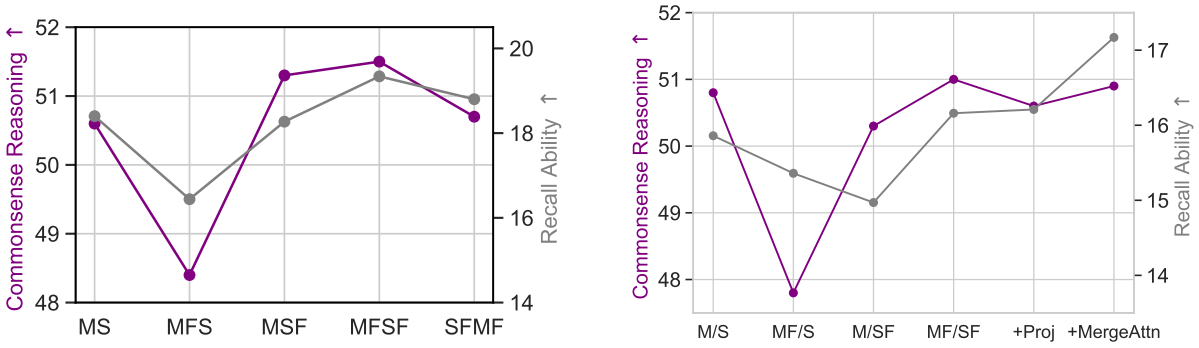

Figure 3: Performance comparison of commonsense reasoning accuracy and recall ability across different model architectures. (**a**) Results for sequential models. (**b**) Results for parallel models. For further details, refer to the first paragraph of Section 4.1.

3.2 Evaluation

Setup We evaluate hybrid models across three axes: (1) long-context language modeling, (2) commonsense reasoning, and (3) memory recall, following previous works on hybrid models. For language modeling, we report perplexity on the SlimPajama validation set using 16k-token sequences. Commonsense reasoning is assessed by averaging accuracy across five standard benchmarks: LAMBADA-OpenAI (Radford et al., 2019), HellaSwag (Zellers et al., 2019), PIQA (Bisk et al., 2020), ARC-Easy (Clark et al., 2018), and Winogrande (ai2, 2019). Recall ability is evaluated over average of eight datasets in Based benchmark (Arora et al., 2024), using the evaluation protocol of Yang et al. (2025). We further group them into short- and long-context subsets to study the influence of context length on recall performance. Details are in Appendix C.2.

Correlation Between Evaluation Axes We investigate how the three evaluation axes, language modeling, commonsense reasoning, and memory recall, relate across different architectural choices. We find that **strong performance on reasoning or language modeling does not necessarily imply strong memory recall.** While there is some positive correlation, it is relatively weak. Specifically, the pearson correlation coefficient between language modeling and commonsense reasoning is high (0.814), whereas recall correlates modestly with reasoning (0.697) and even less with language modeling (0.542). These trends are also visualized in Figure 2, which shows the correlation between recall ability (x-axis) and reasoning (y-axis). Notably, the clustering of models with similar architectures (indicated by color) suggests that architectural design has a greater impact on recall performance than overall reasoning ability. These findings highlight that prior works, which evaluate models solely on language modeling or reasoning benchmarks (Glorioso et al., 2024; Lieber et al., 2024; Ren et al., 2024), need a more comprehensive evaluation including memory-intensive tasks to more accurately assess model capabilities.

4 How Does the Architectural Design Affect Model Performance?

In this section, we present our experimental results across various model architectures (Section 4.1) and provide a detailed analysis of their structural design (Section 4.2).

4.1 Results

Figure 3 compares commonsense reasoning and recall performance across various *block* designs in both sequential and parallel model architectures. In both subfigures, the x-axis represents different block configurations. M indicates a Mamba layer, S a Sliding Window Attention (SWA) layer, and F a feed-forward (FF) layer. For example, MFSF represents a block with Mamba, FF, SWA, and FF layers in that order. In parallel models (Figure 3b), '|' denotes parallel branches (e.g., M|SF means Mamba on one side and SWA+FF on the other). Aggregation strategies are defined as follows: +PROJ uses a trainable projection layer; +MERGEATTN uses a trainable attention module, similar to the cross-attention layer in encoder-decoder models, but using Mamba's output embeddings as the Key and Value; the remaining variants use simple mean averaging. See Appendix D.1 for detailed performance and Appendix D.2 for com-

parison between hybrid and non-hybrid models.

Impact of SWA and Mamba Layer Order on Sequential Hybrid Performance We investigate how the order of SWA and Mamba layers affect sequential hybrid performance by comparing two configurations: MFSF (Mamba before SWA) and SFMF (SWA before Mamba). As shown in Figure 3a, MFSF consistently outperforms SFMF across tasks. This suggests that placing the Mamba layer first helps the model capture global dependencies early, while placing SWA first may bottleneck performance due to its limited attention window. However, when analyzing recall performance by context length (Figure 16 in Appendix D.3), SFMF performs better on shorter contexts. We attribute this to SWA effectively approximating full attention when the input length is within its window, enabling strong local representations that Mamba can refine. In summary, SFMF may benefit short-context tasks, but MFSF, architecture used in Ren et al. (2024), offers superior overall performance. We therefore adopt MFSF as our default sequential model architecture.

Effect of Aggregation Method in Parallel Hybrids Performance We study how different aggregation layers for combining SWA and Mamba output embeddings affect hybrid model performance. As shown in Figure 3b, we evaluate three strategies: +BOTH, PROJ, and MERGEATTN. MERGEATTN achieves the best overall performance, particularly in long-context language modeling (see Appendix D.4 for more results). We thus use MERGEATTN as the representative parallel model in subsequent analysis. Simple averaging (+BOTH) performs well on commonsense reasoning, consistent with observation in Dong et al. (2024), but we observe that it underperforms on recall; for strong recall, especially with long contexts, trainable aggregation methods like PROJ and MERGEATTN are more effective.

Sequential models excel in short contexts, parallel models excel in long ones When comparing recall performance of sequential and parallel models, we observe that sequential models tend to perform better in relatively shorter contexts whereas parallel combinations general show superior performance in longer contexts (Figure 4). We hypothesize that this trend arises from the differing degrees of interaction between the SWA and Mamba components. As parallel models has less interaction

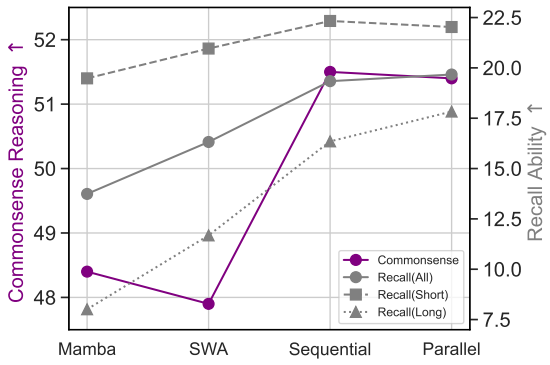

Figure 4: Performance of best performing models from each architecture in commonsense reasoning and recall ability, where divided by length of context.

between Mamba and SWA components, it prevents from collapsing into a shared mode of producing overly similar hidden states. It instead encourages each component to retain its distinct representational strength. In Section 4.2, we provide empirical evidence supporting this hypothesis.

Impact of Adding Feed-Forward Layers on Hybrid Model Performance Feed-formward (FF) layers play an important role in transformers (Geva et al., 2020; Meng et al., 2022), but their effect on hybrid models remains less explored. We find that adding FF layers to only one component, either Mamba or SWA, degrades performance in both sequential and parallel settings, while improvements appear only when FF layers are added to *both* components. We hypothesize that this degradation arises from feature misalignment: it is especially harmful in parallel architectures, where components maintain distinct representations and make aggregation harder, whereas sequential models integrate features into a shared space, mitigating some of these issues. This drop is particularly high when adding FFNs to Mamba, likely because its final layer (C in Equation 1) already functions similarly to an MLP (Sharma et al., 2024), making additional FFNs redundant or even detrimental. This aligns with prior findings that FFNs benefit SWA but not Mamba (Gu and Dao, 2023).

Generalization to 1.3B Trends observed at the 430M scale generally hold at 1.3B. Hybrid models consistently outperform non-hybrids. Among sequential hybrids, MFSF outperforms MS. In parallel setups, merge-attention as an aggregation layer shows higher performance, especially for long-context recall. Overall, merge-attention mechanisms show strong performance. Sequential hy-

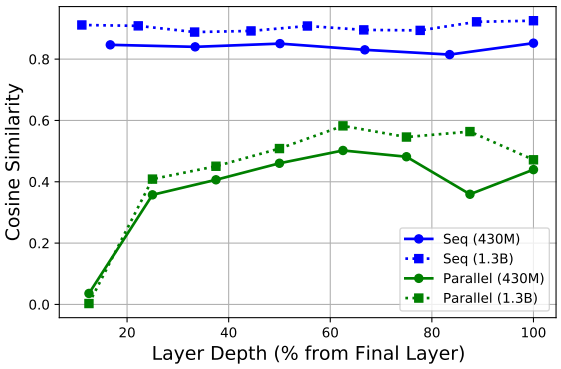

Figure 5: Cosine similarity between output embeddings of aligned SWA and Mamba layers (y-axis), plotted against layer depth, measured as percentage distance from the final layer (x-axis).

brids excel in short-context settings, while parallel hybrids perform better with longer contexts. See Appendix D.6 for detailed results.

4.2 Analysis

Similarity between SWA and Mamba Output Embeddings in Hybrid Models To better understand the interaction between SWA and Mamba in hybrid models, we analyze the cosine similarity of their output embeddings across block depths, aligned by position from the final block (Figure 5). Sequential hybrids show high similarity, especially in the larger 1.3B model, because outputs from one component feed into the next, naturally aligning their representations. Parallel hybrids show much lower similarity, particularly in early and middle layers, as both components process inputs independently and fuse outputs later. We hypothesize that this structural difference shapes performance: sequential hybrids benefit from stable, aligned representations for commonsense reasoning and short-context tasks but struggle with long-context reasoning. In contrast, parallel hybrids produce more diverse representations and, though sensitive to aggregation strategy, can outperform on complex long-context tasks when effectively trained. More analysis in Appendix D.7.

Identifying Critical Components in Hybrid Blocks Figure 6 shows performance degradation on commonsense (left) and recall (right) tasks when removing blocks by depth. Removing the first block causes the steepest drop, up to 90% on recall tasks, highlighting the crucial role of early layers. We further examine the importance of subcomponents within each block. In sequential models, the first subcomponent is most critical because it shapes the feature space, and later components align to it. In parallel models, the aggregation layer is most critical as it must merge the distinct representation spaces from Mamba and SWA, while either path alone can still infer the input distribution. See Appendix D.8 for further discussion.

Understanding the performance gains of MergeAttn Among the various configurations, parallel hybrids using an attention layer that merges output embeddings of SSM and attention achieve the best performance. To understand why these models tend to perform strongly, we analyze the models on how much each token is influenced by prior tokens, following the method in Ben-Kish et al. (2024); higher value indicates that they exhibit stronger attention to previous tokens. As shown in Figure 7, models with merge-attention show the highest average attention weights, suggesting that their improved performance arises because the Mamba layers effectively capture global dependencies, which the merge-attention mechanism then leverages to integrate information. See Appendix D.9 for more detail of calculation.

5 Dataset Strategy to Enhance Recall

We show that continually training models on datasets with paraphrased contexts, drawn from a distribution similar to the pretraining dataset, improves recall without sacrificing commonsense reasoning. Previous work focused on improving recall through *architectural changes*, such as hybrid models. Here, we investigate a *data-centric approach*, aiming to complement and further enhance these architectural advances.

Section 5.1 describes how we construct the training dataset and train the model. Section 5.2 shows that models trained on our dataset achieve the best trade-off between recall and reasoning, outperforming other dataset choices and DeciMamba (which introduces architectural changes) across scales up to 2.8B parameters. Section 5.3 analyzes design factors such as input length, dataset size, and model choice, demonstrating through extensive experiments that our simple approach generalizes well and consistently improves performance.

5.1 Experimental Setup

Paraphrasing Method We construct a paraphrased dataset using a subset of the training corpus (SlimPajama), based on the hypothesis that

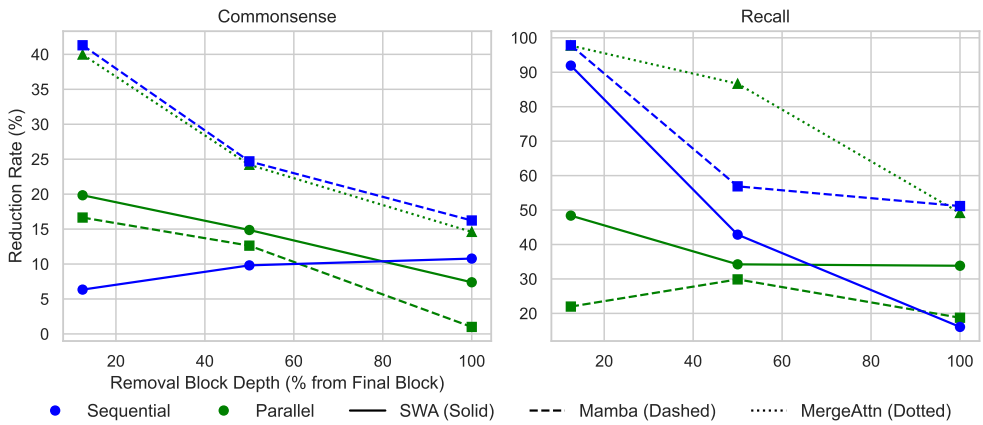

Figure 6: Performance degradation (y-axis) on commonsense (left) and recall (right) tasks as a function of the removed block's relative position from the final block (x-axis).

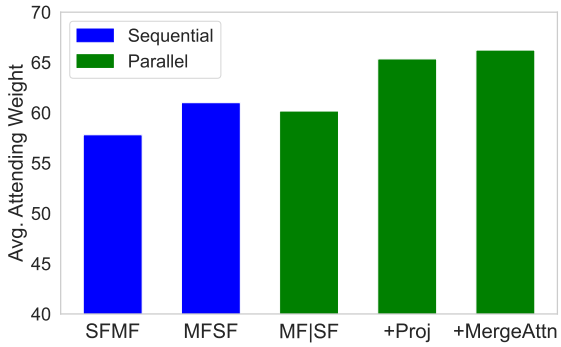

Figure 7: Average attending weight across different model architectures. Higher values indicate that the model attends more strongly to previous information.

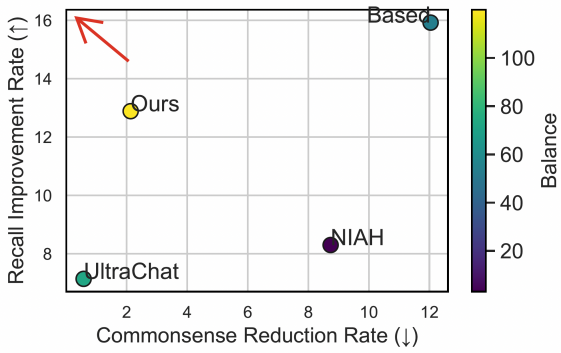

Figure 8: The upper-left region (indicated by the red arrow) represents the optimal balance between recall improvement and commonsense degradation.

the data should remain close in distribution to the original pretraining corpus to prevent degrading existing performance. To control the density of paraphrased content, we divide the data into 1k-token chunks. For each chunk, we use LLaMA 3.1–8B[3] to generate factual question-answer (QA) pairs. Following Arora et al. (2024), we convert these QA pairs into cloze-style paraphrased sentences. This yields pairs of the form (1k-token chunk, paraphrased sentence). To construct a training dataset, we concatenate multiple chunks and insert the corresponding paraphrased sentence at a random position following the chunk it was derived from. Based on the constructed dataset, we run a filtering process based on three criteria: (1) the model fails to generate a valid question and answer pair, (2) the generated answer is not present in the corresponding paragraph, or (3) the model fails to convert the example into a cloze-style task. See Appendix E.1 for more details.

[3]We use the released model from Hugging Face: `meta-llama/Meta-Llama-3-8B-Instruct`

Training Details After the initial pretraining phase,[4] we continue training the model using several different datasets, including recall-intensive datasets such as NIAH and SQuAD from Based, widely used SFT dataset UltraChat (Ding et al., 2023), and our paraphrased dataset. Following the setup of Ben-Kish et al. (2024), we train the models using a batch size of 32, a learning rate of 1e-4 for 10 epochs. We conduct experiments with both hybrid models and Mamba-only models.

5.2 Results

Our Dataset Strikes the Best Balance Figure 8 shows the balance between commonsense degradation and recall gains[5] for the 430M sequential hybrid model (MFSF) when trained on various datasets including NIAH (Kamradt, 2023), UltraChat (Waleffe et al., 2024), or SQuAD dataset from

[4]We also experimented with incorporating the paraphrase dataset during pretraining. However, we observed a degradation in performance when doing so (see Appendix E.2).

[5]In this section, we exclude SQuAD from Based when computing average recall, as it is part of the training data.

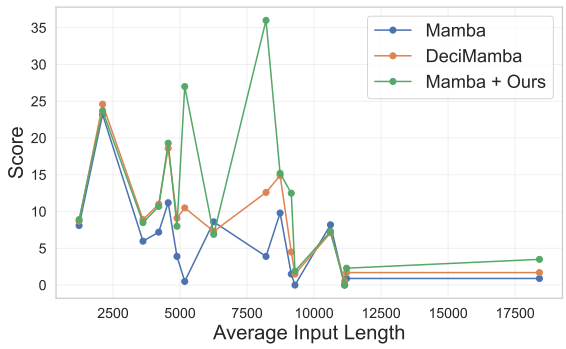

Figure 9: Performance (y-axis) of Mamba, DeciMamba, and Mamba trained with our dataset across LongBench datasets with varying input lengths (x-axis).

Based benchmark (Arora et al., 2024). Models trained on our paraphrased SlimPajama dataset consistently achieve the best balance. We attribute this to: (1) its alignment with the original pretraining distribution, preserving baseline performance; and (2) paraphrased content promoting the model to retain and utilize previous context. In contrast, recall-focused datasets like NIAH and Based significantly harm commonsense performance, while UltraChat offers only modest recall improvements. Appendix E.3 provides detailed performance. Similar patterns hold for Mamba models, including the released 2.8B version (Appendix E.4).

Comparison with DeciMamba We investigate whether a *data-centric* approach can outperform architectural modifications by comparing our approach with DeciMamba (Ben-Kish et al., 2024), which enhances recall by discarding less important tokens. Across 16 datasets in LongBench (Bai et al., 2023) using Mamba-2.8B, our approach achieves an average of +12.7 points overall, with particularly strong gains in QA tasks (+8.1 on average). As shown in Figure 9, our approach tends to consistently outperform DeciMamba on medium and long input lengths. These results suggest that our data-centric approach is not only complementary to architectural change but can also show strong standalone performance. Full results are provided in Appendix E.5.

5.3 Analysis

To understand the benefit and affect of such approach, we analyzed over various design choices.

Generalizes to Various Base Models We observe that **our method generalizes across different released variants of Mamba-2.8B**. Continual training yields performance gains for the base model (+1.7 in commonsense, +6.5 in recall), as well as for instruction-tuned (+1.3 in commonsense, +3.6 in recall) and preference-aligned models (+3.4 in commonsense, +0.8 in recall). Improvements are generally more pronounced for the base model. See Appendix E.6 for model details and results.

Longer chunk sizes yield stronger results We observe that models trained on longer sequences tend to achieve lower reduction rates on commonsense tasks and substantially higher gains on recall tasks, especially on long-context tasks (Figure 19 in Appendix)[6]. Training on shorter chunk sizes (e.g., 2k) tends to enhance performance on short-context recall but leads to high degradation on long-context tasks. This trend is robust across architectures, appearing in both sequential and parallel hybrids, and holds for different model sizes. For detailed results, refer to Appendix E.7.

Performance improves as the size of the training dataset increases We observe that training with larger datasets leads to clear gains in both commonsense reasoning and recall tasks: models trained with more tokens achieve a lower reduction rate on commonsense benchmarks and steadily higher improvements on recall performance[7]. Performance grows with the amount of training data and begins to converge around 80M-100M tokens. This trend holds across different model sizes and hybrid architectures. More details are in Appendix E.8.

6 Conclusion

In this paper, we focus on two main aspects: (1) studying how different architectural design choices (sequential, parallel) affect performance in hybrid models and the roles of individual components (SSM and Attention layers), and (2) exploring data-centric approaches to further improve model's recall ability. Our findings show that sequential models offer stable training but are limited in expressiveness, while parallel architectures better preserve the unique characteristics of each component, often leading to stronger performance. In particular, parallel hybrid models with merge-attention-based aggregation consistently perform well. We also demonstrate that continually pretraining the model on a paraphrased dataset effectively improves recall while maintaining overall model performance.

[6] All experiments used a fixed token count of 10M, discarding the final chunk if it does not align with the sequence length.

[7] All experiments use a fixed chunk size of 4k

Limitations

Due to computational constraints, we conducted our experiments on relatively small model scales, 430M and 1.3B parameters, trained with 100B tokens. Pretraining a 430M-parameter model on 8 A100 GPUs takes about one week, while a 1.3B-parameter model requires roughly two weeks, making it challenging to analyze larger-scale models. Notably, prior work on hybrid models has also primarily operated at similar scales (Ren et al., 2024; Dong et al., 2024; Yang et al., 2025). These resource limitations also restricted our ability to explore a broader range of hybrid model configurations and focus on the experimental setup described in the "Hybrid Model" section (Section 2). We leave more extensive analyses, such as incorporating additional components like gated DeltaNet, to future work.

References

2019. Winogrande: An adversarial winograd schema challenge at scale.

Simran Arora, Aman Timalsina, Aaryan Singhal, Sabri Eyuboglu, Xinyi Zhao, Ashish Rao, Atri Rudra, and Christopher Ré. 2024. Just read twice: closing the recall gap for recurrent language models.

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

Yushi Bai, Xin Lv, Jiajie Zhang, Hongchang Lyu, Jiankai Tang, Zhidian Huang, Zhengxiao Du, Xiao Liu, Aohan Zeng, Lei Hou, and 1 others. 2023. Longbench: A bilingual, multitask benchmark for long context understanding. *arXiv preprint arXiv:2308.14508*.

Ali Behrouz, Peilin Zhong, and Vahab Mirrokni. 2024. Titans: Learning to memorize at test time. *arXiv preprint arXiv:2501.00663*.

Iz Beltagy, Matthew E. Peters, and Arman Cohan. 2020. Longformer: The long-document transformer. *arXiv:2004.05150*.

Assaf Ben-Kish, Itamar Zimerman, Shady Abu-Hussein, Nadav Cohen, Amir Globerson, Lior Wolf, and Raja Giryes. 2024. Decimamba: Exploring the length extrapolation potential of mamba. *Preprint*, arXiv:2406.14528.

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

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

Tri Dao and Albert Gu. 2024. Transformers are SSMs: Generalized models and efficient algorithms through structured state space duality. In *International Conference on Machine Learning (ICML)*.

Ning Ding, Yulin Chen, Bokai Xu, Yujia Qin, Zhi Zheng, Shengding Hu, Zhiyuan Liu, Maosong Sun, and Bowen Zhou. 2023. Enhancing chat language models by scaling high-quality instructional conversations. *arXiv preprint arXiv:2305.14233*.

Xin Dong, Yonggan Fu, Shizhe Diao, Wonmin Byeon, Zijia Chen, Ameya Sunil Mahabaleshwarkar, Shih-Yang Liu, Matthijs Van Keirsbilck, Min-Hung Chen, Yoshi Suhara, Yingyan Lin, Jan Kautz, and Pavlo Molchanov. 2024. Hymba: A hybrid-head architecture for small language models.

Dheeru Dua, Yizhong Wang, Pradeep Dasigi, Gabriel Stanovsky, Sameer Singh, and Matt Gardner. 2019. Drop: A reading comprehension benchmark requiring discrete reasoning over paragraphs. *arXiv preprint arXiv:1903.00161*.

Mor Geva, Roei Schuster, Jonathan Berant, and Omer Levy. 2020. Transformer feed-forward layers are key-value memories. *arXiv preprint arXiv:2012.14913*.

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

Albert Gu and Tri Dao. 2023. Mamba: Linear-time sequence modeling with selective state spaces. *COLM*.

Mandar Joshi, Eunsol Choi, Daniel S Weld, and Luke Zettlemoyer. 2017. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. *arXiv preprint arXiv:1705.03551*.

Gregory Kamradt. 2023. Needle in a haystack- pressure testing llms. *Github*.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, and 1 others. 2019. Natural questions: a benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:453–466.

Hyunji Lee, Sejune Joo, Chaeeun Kim, Joel Jang, Doyoung Kim, Kyoung-Woon On, and Minjoon Seo. 2023. How well do large language models truly ground? *arXiv preprint arXiv:2311.09069*.

Opher Lieber, Barak Lenz, Hofit Bata, Gal Cohen, Jhonathan Osin, Itay Dalmedigos, Erez Safahi,

Shaked Meirom, Yonatan Belinkov, Shai Shalev-Shwartz, and 1 others. 2024. Jamba: A hybrid transformer-mamba language model. *arXiv preprint arXiv:2403.19887*.

Colin Lockard, Prashant Shiralkar, and Xin Luna Dong. 2019. 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)*, pages 3047–3056, Minneapolis, Minnesota. Association for Computational Linguistics.

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

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. 2022. Locating and editing factual associations in gpt. In *Neural Information Processing Systems*.

Jongho Park, Jaeseung Park, Zheyang Xiong, Nayoung Lee, Jaewoong Cho, Samet Oymak, Kangwook Lee, and Dimitris Papailiopoulos. 2024. Can mamba learn how to learn? a comparative study on in-context learning tasks. *International Conference on Machine Learning*.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know what you don't know: Unanswerable questions for squad. *Preprint*, arXiv:1806.03822.

Liliang Ren, Yang Liu, Yadong Lu, Yelong Shen, Chen Liang, and Weizhu Chen. 2024. Samba: Simple hybrid state space models for efficient unlimited context language modeling. *ICLR*.

Arnab Sen Sharma, David Atkinson, and David Bau. 2024. Locating and editing factual associations in mamba. *arXiv preprint arXiv:2404.03646*.

Daria Soboleva, Faisal Al-Khateeb, Robert Myers, Jacob R Steeves, Joel Hestness, and Nolan Dey. 2023. SlimPajama: A 627B token cleaned and deduplicated version of RedPajama.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, and 1 others. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

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

Roger Waleffe, Wonmin Byeon, Duncan Riach, Brandon Norick, Vijay Korthikanti, Tri Dao, Albert Gu, Ali Hatamizadeh, Sudhakar Singh, Deepak Narayanan, and 1 others. 2024. An empirical study of mamba-based language models. *arXiv preprint arXiv:2406.07887*.

Songlin Yang, Jan Kautz, and Ali Hatamizadeh. 2025. Gated delta networks: Improving mamba2 with delta rule. In *The Thirteenth International Conference on Learning Representations*.

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

Jingwei Zuo, Maksim Velikanov, Dhia Eddine Rhaiem, Ilyas Chahed, Younes Belkada, Guillaume Kunsch, and Hakim Hacid. 2024. Falcon mamba: The first competitive attention-free 7b language model. *arXiv preprint arXiv:2410.05355*.

A Preliminary

A.1 Mamba and Attention layers

Given an input sequence X, both Mamba layers and attention layers transform it into an output sequence Y via a transformation matrix M: M_{Mamba} (Equation 1) and M_{Attn} (Equation 2). The key difference lies in how they process inputs. Mamba layers update a recurrent hidden state h_t sequentially, incorporating one token x_t at a time. This hidden state serves as a compressed memory summarizing all past inputs. In contrast, Attention layers process the entire input sequence at once, attending to all tokens up to the current position, thereby capturing dependencies without recurrence. These design choices yield different trade-offs. Mamba is more computationally efficient due to its linear-time recurrence but may struggle with long-range dependencies. Attention layers, while effective at modeling token-wise relationships, incur quadratic time and memory complexity with sequence length.

$$Y_{\text{Mamba}} = M_{\text{Mamba}} X \quad \text{where} \tag{1}$$

$$Y_{\text{Mamba},t} = Ch_t, \quad h_t = Ah_{t-1} + Bx_t, \quad x_t \in X$$

$$Y_{\text{Attn}} = M_{\text{Attn}} X \quad \text{where} \tag{2}$$

$$Y_{\text{Attn}} = \text{softmax}\left(\frac{(W_Q X)(W_K X)^T}{\sqrt{d_k}}\right)(W_V X)$$

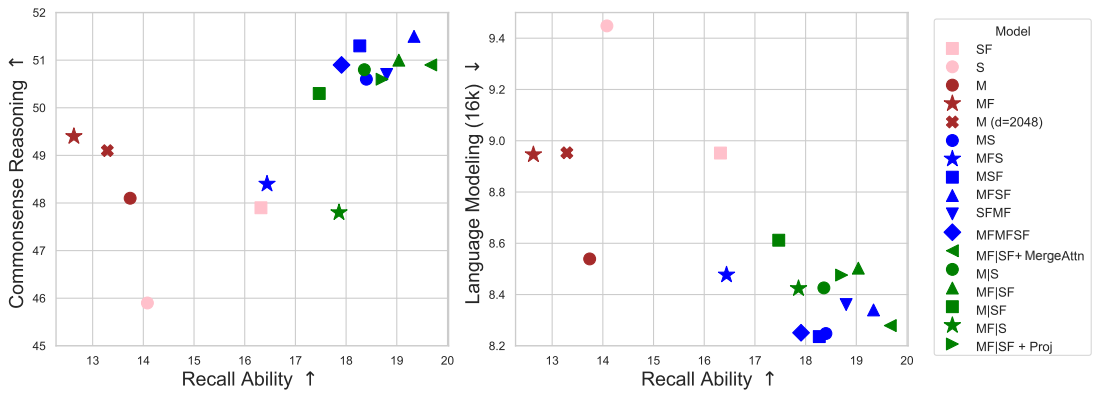

Figure 10: Comparison of detailed model architectures on Commonsense Reasoning (y-axis on left) and Language Modeling(y-axis on right) vs. Recall Ability (x-axis). Commonsense Reasoning and Recall Ability are measured using answer accuracy. The models compared included Mamba-only, SWA-only, Hybrid (Sequential), and Hybrid (Parallel).

B Related Works

B.1 Studies on SSMs

Prior work has explored state space models (SSMs) (Waleffe et al., 2024; Sharma et al., 2024), primarily focusing on their performance in language modeling tasks, particularly their ability to handle long-context dependencies. However, these studies typically examine pure SSM architectures and do not consider hybrid models. In this work, we conduct an empirical investigation of *hybrid architectures* that combine SSM and attention layers. Our goal is to understand the source of their performance gains and the distinct roles played by each component.

B.2 Recall Ability of Language Models

Enhancing a model's recall ability, also referred to as grounding ability, is a critical aspect of language modeling, especially in scenarios where the model must answer questions based on a given context, maintain strong coherence across parts of a conversation or document, or perform consistent reasoning over extended texts (Arora et al., 2024; Lee et al., 2023). This ability allows the model to retrieve relevant information accurately from given context, sustain contextual coherence, and generate factually grounded responses.

In this paper, we define recall ability as distinct from the general capability to model long contexts. Unlike next-token prediction, recall-intensive tasks require the model to retrieve specific values or answers from earlier in the context, demanding precise and accurate memory. Furthermore, evaluating recall ability is not limited to long-context tasks; it applies to any setting where exact retrieval from prior context is necessary.

Several studies have shown that SSM-based models often struggle with such recall-intensive tasks, as they must encode prior context into fixed-size hidden states. This architectural constraint leads to a bottleneck that limits their recall performance (Park et al., 2024; Ren et al., 2024; Dong et al., 2024).

C Designing a Unified Experimental Setup

C.1 Correlation Between Evaluation Axes

Figure 10 shows the detailed configuration of each points in Figure 2.

C.2 Dataset

We experiment over dataset from Arora et al. (2024) (Based benchmark) to calculate recall ability. Based benchmark is comprised of eight datasets: NQ (Kwiatkowski et al., 2019), TriviaQA (Joshi et al., 2017), DROP (Dua et al., 2019), FDA (Arora et al., 2023), SWDE (Lockard et al., 2019), and SQuAD (Rajpurkar et al., 2018). The NQ dataset is further subdivided by input length into NQ-512, NQ-1024, and NQ-2048. To compare the recall ability across different sequence lengths, we categorize the eight datasets into two groups: relatively short sequences (NQ-512, DROP, TriviaQA, SQuAD) and relatively long sequences (NQ-1024, NQ-2048, FDA, SWDE). Using the LLaMA-2 tokenizer (Touvron et al., 2023), which was also used during training, the average input length is around 1k tokens for the short-sequence group and around 2.5k tokens for the long-sequence group.

The Slimpajama dataset and evaluation datasets are released under Apache 2.0 license. We used the datasets for research purpose.

D How does the architectural design affect model performance?

D.1 Performance at the 430M scale

Table 1 presents the performance of models at a 430M parameter scale. Figures 11 and 12 show the language modeling performance, measured in terms of perplexity on the SlimPajama validation set, for sequential and parallel hybrid models, respectively.

D.2 Hybrid models outperform non-hybrid models in recall and commonsense reasoning

Results in Figure 4 show the performance of non-hybid models (Mamba, SWA) and hybrid models (Sequential, Parallel). We observe consistent gains in both commonsense reasoning and recall performance in hybrid models over non-hybrid ones in line with prior works (Ren et al., 2024; Dong et al., 2024; Park et al., 2024). Notably, we observe that recall shows a substantially larger improvement, with an average increase of 29.5%, compared to a 7.3% gain in commonsense reasoning. These improvements are especially prominent in long-context scenarios. In contrast, in short-context settings, performance differences are less pronounced, and hybrid models perform similarly to the SWA baseline.

D.3 SWA as the Initial Component Improves Short Context Recall

Figure 16 presents the average recall performance for both short and long sequences in the sequential architecture. The SFMF configuration demonstrates stronger performance on shorter sequences. We hypothesize that this is because, in short contexts, the input length fits within the window size of the SWA module, allowing it to approximate full attention more effectively.

D.4 Trainable Aggregation Layers Improve Performance on Long Contexts

Figure 12 presents perplexity on the SlimPajama validation dataset across different chunk sizes. Models equipped with trainable aggregation layers, specifically MF|SF (+proj) and

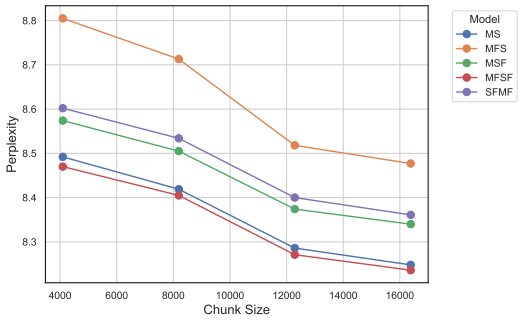

Figure 11: Peplexity over sequence length for sequential hybrids

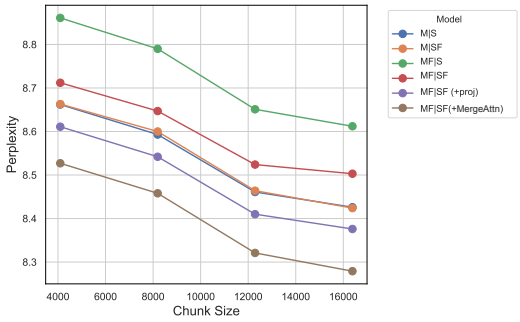

Figure 12: Peplexity over sequence length for parallel hybrids

MF|SF (+MergeAttn), consistently outperform others across varying context lengths. These models show strong recall performance, with particularly notable improvements in longer ones (Figure 13).

D.5 Impact of Adding Feed-Forward Layers on Hybrid Model Performance

Prior work has shown the importance of feed-forward (FF) layers in transformers (Geva et al., 2020; Meng et al., 2022). Thereby, we investigate their impact on hybrid models. Interestingly, adding FF layers to only one component, either Mamba or SWA, degrades performance in both sequential and parallel settings, and performance improves only when FF layers are added to both components. In the sequential setup (Figure 3a), the baseline MS outperforms MFS and MSF, but is lower than MFSF. Similarly, in the parallel setup (Figure 3b), M|S show higher performance over MF|S and SF|M but lower than SF|MF.

We hypothesize that this degradation stems from feature misalignment. The effect is more pronounced in parallel architectures, where individual component characteristics are preserved, making it harder to aggregate misaligned features. In contrast, sequential models integrate features into a shared

	Commonsense Reasoning						Recall Ability								
Model Type	LAM.	Hella.	PIQA	ARC	Wino.	***Avg.***	NQ-S	NQ-M	NQ-L	Drop	FDA	SWDE	TQA	SQD	***Avg.***
M	31.7	43.1	68.2	45.2	52.6	48.1	9.6	8.7	7.5	11.4	1.8	14.1	38.5	18.4	13.7
MF	34.2	41.6	68.6	51.8	51.0	49.4	9.2	8.8	6.3	10.4	1.1	12.3	36.0	17.0	12.6
S	30.6	37.6	64.6	45.3	51.3	45.9	9.9	9.2	6.4	12.4	3.2	18.4	31.9	13.4	13.1
SF	35.4	38.7	65.7	48.7	51.0	47.9	10.8	7.9	6.8	12.3	15.5	16.5	40.8	20.0	12.6
Sequential Hybrid															
MS	40.8	44.0	67.2	50.0	50.9	50.6	12.6	12.2	8.0	11.6	14.3	26.4	41.7	20.6	18.4
MFS	34.9	41.8	67.6	44.5	53.2	48.4	10.1	8.8	7.6	11.8	14.6	21.6	37.8	19.3	16.4
MSF	39.0	43.3	67.9	52.5	53.8	51.3	12.3	11.5	7.0	11.7	16.4	24.8	41.9	20.6	18.3
MFSF	38.5	44.2	69.1	51.7	54.0	**51.5**	13.5	12.3	8.0	11.2	16.5	28.6	43.4	21.2	19.3
SFMF	37.4	42.8	68.6	52.1	52.5	50.7	12.8	11.6	7.6	12.2	15.5	25.6	43.2	22.0	18.8
Parallel Hybrid															
M|S	40.1	42.7	67.9	50.1	53.1	50.8	11.5	10.5	7.3	11.9	16.2	26.7	42.8	20.0	18.4
MF|S	24.5	42.5	68.3	52.3	51.7	47.8	11.3	11.3	7.2	11.6	15.5	25.6	40.9	19.6	17.9
M|SF	37.5	41.2	67.6	51.4	53.7	50.3	11.0	10.0	6.9	10.9	14.9	24.5	41.7	19.9	17.5
MF|SF (Avg)	39.3	42.8	67.9	52.8	52.3	51.0	11.7	11.9	8.0	12.3	16.8	28.0	42.1	19.9	18.8
MF|SF (Proj)	38.0	42.6	69.4	51.0	52.0	50.6	11.9	12.4	8.4	12.7	16.9	28.6	42.3	20.7	19.2
MF|SF (MergeAttn)	39.3	44.3	69.0	51.9	52.3	51.4	12.8	12.9	9.0	11.9	17.7	29.6	43.1	20.3	**19.7**

Table 1: Model performance at the 430M scale. Model Type: M = Mamba, S = SWA, F = FF layer. The order reflects the design sequence within each block. In parallel hybrids, "|" denotes parallel branches (e.g., M|SF means Mamba on one side, SWA+FF on the other). Tasks: LAM. = LAMBADA-OpenAI, Hella. = HellaSwag, ARC = ARC-Easy, Wino. = Winogrande, NQ-S = NQ-512, NQ-M = NQ-1024, NQ-L = NQ-2048, TQA = TriviaQA, SQD = SQuAD. Bold indicates the highest average performance. In both cases, the best models use hybrid architectures with merge-attention.

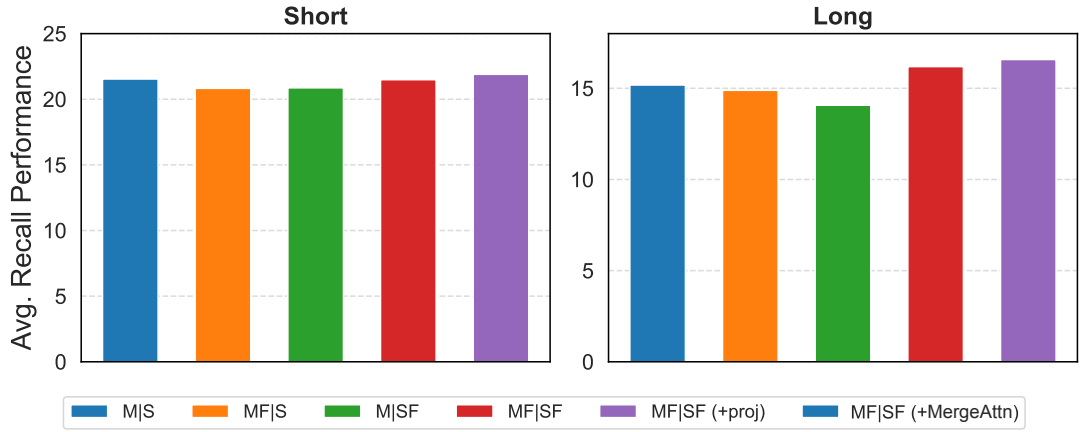

Figure 13: Comparison of average recall performance across short and long input contexts for parallel hybrids

space, mitigating this effect. Also, the performance drop is especially large when adding FFNs to the Mamba layer, possibly because its final layer (C in Equation 1) already functions similarly to an MLP (Sharma et al., 2024), making an additional FFN redundant or even detrimental. This aligns with prior findings that FFNs benefit SWA but not Mamba (Gu and Dao, 2023).

D.6 Performance at the 1.3B Scale

Table 2 presents the performance of models at the 1.3B parameter scale. Due to computational constraints, we limited our experiments to configurations that demonstrated strong performance at the 430M scale. We observe consistent trends across both scales. Hybrid models outperform their non-hybrid counterparts. Among sequential architectures, the MFSF model achieves the best performance. Additionally, parallel architectures that use merge-attention layers for fusion generally yield the highest performance. We also observe a similar pattern from 430M scale (Figure 4) when comparing short- vs. long-sequence settings in recall ability in 1.3B scale (Figure 15).

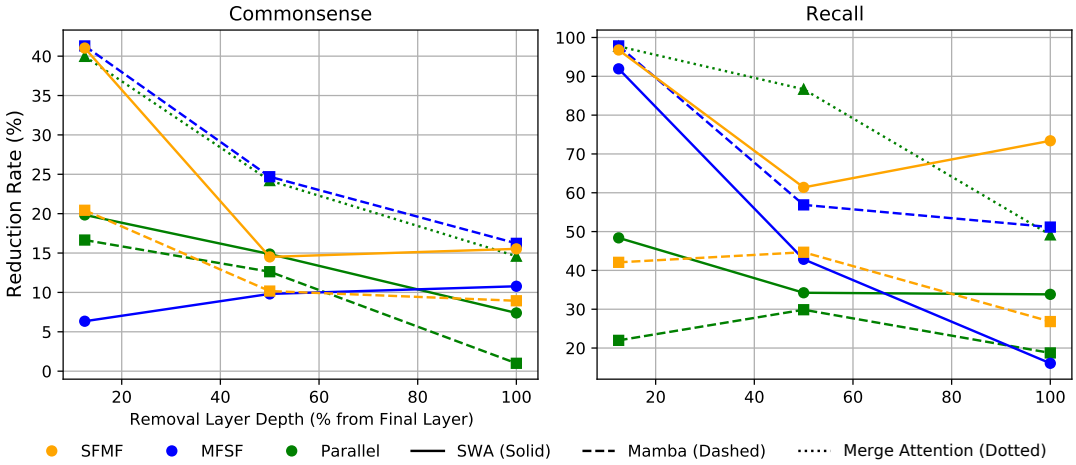

Figure 14: Performance degradation (y-axis) on commonsense (left) and recall (right) tasks as a function of the removed block's relative position from the final block (x-axis) for sequential(SFMF), sequential(MFSF) and parallel(+MergeAttn) architecture.

	Commonsense Reasoning						Recall Ability								
Model Type	LAM.	Hella.	PIQA	ARC	Wino.	***Avg.***	NQ-S	NQ-M	NQ-L	Drop	FDA	SWDE	TQA	SQD	***Avg.***
M	45.3	52.7	72.1	67.6	54.9	58.5	15.8	13.0	10.6	16.9	4.1	14.8	50.8	23.6	18.7
SF	47.2	49.8	69.5	65.9	53.4	57.1	18.0	16.0	10.7	19.1	10.3	29.0	52.0	24.9	22.5
Sequential Hybrid															
MS	48.9	48.8	69.9	65.3	54.9	57.6	17.9	15.4	10.3	19.3	45.9	26.4	53.8	23.1	26.5
MFSF	52.9	52.6	71.9	68.5	55.4	60.3	17.6	15.6	10.9	20.0	46.2	38.6	53.7	24.7	28.4
Parallel Hybrid															
MF|SF (Avg)	53.5	51.9	71.4	64.1	56.1	59.4	19.1	16.0	12.4	18.6	47.8	35.8	53.3	24.6	28.5
MF|SF (MergeAttn)	54.4	53.7	71.7	68.0	57.4	**61.0**	17.9	16.9	11.8	19.4	48.4	39.7	51.7	26.0	**29.0**

Table 2: Model performance at the 1.3B scale. Due to computational constraints, we evaluate only those configurations that performed well at the 430M scale. Model Type: M = Mamba, S = SWA, F = FF layer. The order reflects the design sequence within each block. In parallel hybrids, "|" denotes parallel branches (e.g., M|SF means Mamba on one side, SWA+FF on the other). Tasks: LAM. = LAMBADA-OpenAI, Hella. = HellaSwag, ARC = ARC-Easy, Wino. = Winogrande, NQ-S = NQ-512, NQ-M = NQ-1024, NQ-L = NQ-2048, TQA = TriviaQA, SQD = SQuAD. Bold indicates the highest average performance. In both cases, the best models use hybrid architectures with merge-attention.

D.7 Similarity between SWA and Mamba Output Embeddings in Hybrid Models

We observe that sequential hybrids exhibit high similarity between SWA and Mamba outputs, especially in the larger 1.3B model, while parallel hybrids show much lower similarity, particularly in early and middle layers. This difference arises from the design: sequential hybrids pass outputs from one component to the next, naturally aligning their embedding distributions. In contrast, parallel hybrids process the same input independently, with their outputs aggregated later, leading to more distinct representations.

This structural difference impacts performance. Sequential hybrids maintain a consistent representational space, enabling stable training and strong results on tasks requiring commonsense reasoning or handling shorter contexts. However, they struggle with longer-context tasks that require richer representations. Parallel hybrids, while more sensitive to aggregation strategies due to the divergence in output spaces, can achieve better performance on complex tasks when trained effectively by leveraging the complementary strengths of both components.

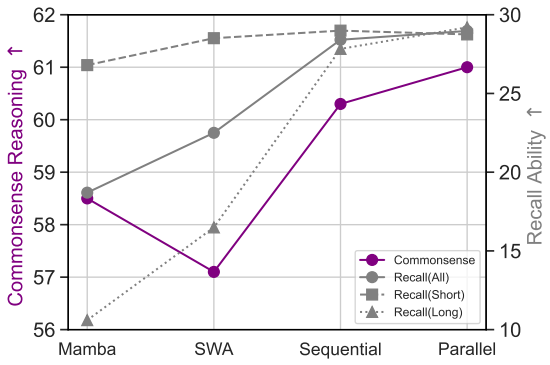

Figure 15: Performance of best performing 1.3B scale models from each architecture in commonsense reasoning and recall ability, where divided by length of context.

D.8 Identifying Critical Components in Hybrid Blocks

Figure 6 presents the performance reduction rates (y-axis) for commonsense tasks (left) and recall tasks (right) as a function of the block removed (x-axis, represented as the percentage depth from the final block). Across most configurations, the removal of the first block results in the highest performance degradation, indicating that early blocks are typically the most critical. This trend is particularly pronounced in recall tasks, where removing the first block often leads to performance drops of around 90%.

To better understand the importance of components within each block, we analyze how the removal of specific subcomponents affects performance across architectures. In sequential architectures, the first subcomponent in each block plays the most important role. In contrast, in parallel architectures, the aggregation mechanism rather than individual components like Mamba or SWA, is the most impactful. In more details, in sequential architectures, such as MFSF, where the Mamba layer is placed first, removing this initial layer leads to significant degradation, while removing the SWA layer has a milder effect. Conversely, in SFMF, which places the SWA layer first, the most substantial drop occurs when the SWA layer is removed, with the Mamba layer being less impactful (Figure 14). These results suggest that the position of the layer (i.e., being the first) has a greater influence on performance than the specific type of layer (Mamba vs. SWA). For parallel architectures, the impact of removing individual Mamba or SWA layers is less severe. Instead, the greatest degradation occurs when aggregation mechanisms such as merge-attention or projection layers are changed to a simple average.

The findings can also be related to the distribution shift caused by each component. Sequential architectures exhibit the strongest distributional shift in the first component, making it consistently important regardless of whether it is Mamba or SWA component. After this initial transformation, subsequent components tend to collapse into similar distributions, reducing their relative impact. In contrast, in parallel architectures, both the Mamba and SWA components process the same input independently. As a result, the distributional shifts introduced by each path are less pronounced, and the model can still form a reasonable representation of the input even if one component is removed. However, the aggregation mechanism causes the largest distributional shift in parallel architectures. Replacing it with simpler methods, such as averaging, can distort the combined representation from the two components, resulting in significant performance degradation.

D.9 Calculating average attending weights

We calculate Mamba hidden attention maps following Ben-Kish et al. (2024). The average attending weight is calculated with a randomly selected 100 samples of the validation set of Slimpajama of a 4k chunk. We average over all tokens and all layers.

E Dataset Strategies to Enhance Recall

E.1 Filtering the Paraphrased Dataset

We apply a filtering process to the paraphrased dataset based on the following criteria: (1) the model fails to generate a valid question and answer pair, (2) the generated answer is not present in the corresponding paragraph, or (3) the model fails to convert the example into a cloze-style task, such as when the answer does not appear at the end of the sentence. Instances that do not meet these criteria are discarded, and the processing continues with the remaining examples. For all experiments, we maintained approximately 3k training instances in the training dataset to ensure a fair comparison.

E.2 Introducing Paraphrased Data: Early vs. Late

We investigate the impact of introducing paraphrased datasets at different stages of pretraining. When added early, performance deteriorates: al-

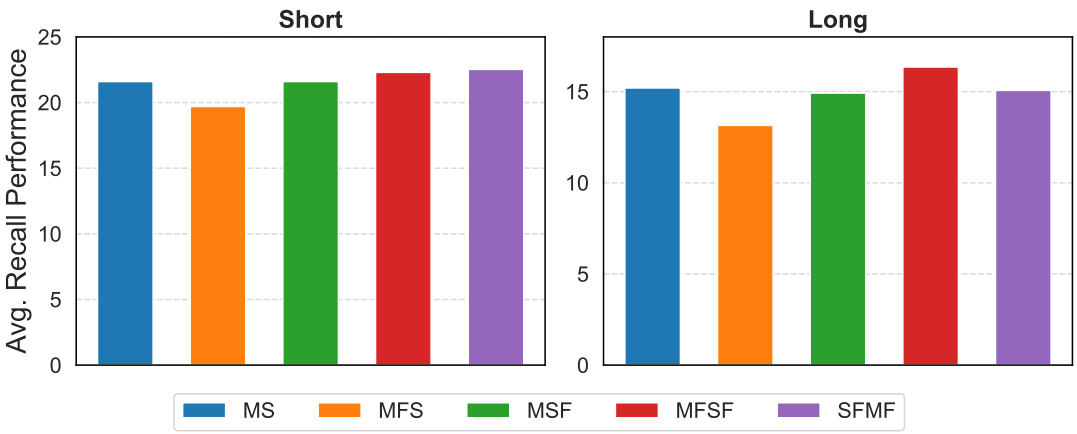

Figure 16: Comparison of average recall performance across short and long input contexts for sequential hybrids

Training Dataset	Commonsense	Recall
Original	51.5	19.1
Based (SQuAD)	45.3	22.3
NIAH	50.4	21.6
UltraChat	47.0	20.6
Ours	51.2	20.4

Table 3: Performance on commonsense reasoning and recall ability after training on the datasets listed in the "Training Dataset" column.

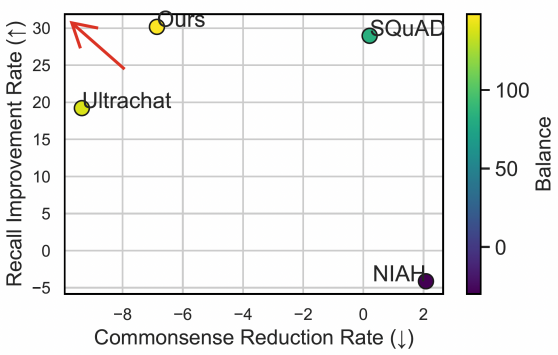

Figure 17: The upper-left region (indicated by the red arrow) represents the optimal balance between recall improvement and commonsense degradation. When training a pretrained 430M Mamba across various datasets, models trained on our dataset (paraphrased SlimPajama) consistently achieve the best balance compared to when training on other datasets.

though training loss decreases steadily, validation loss increases, suggesting overfitting. We hypothesize this is due to the model's limited language modeling ability in the early stages, making it more sensitive to data quality. Additionally, deduplication plays a critical role in preventing overfitting. In contrast, introducing paraphrased data later in continual training stage, as the model is stable, we observe that it consistently improves the recall performance.

E.3 Our Dataset Achieves the Best Balance Across Various Training Datasets

Table 5 shows the performance on commonsense reasoning and recall ability after training on datasets on SQuAD dataset from Based, NIAH, UltraChat, and Ours (paraphrased slimpajama dataset). Please note that we remove the SQuAD dataset when averaging recall ability.

E.4 Ours Also Shows Good Balance on Mamba-Only Models

Figure 17 illustrates that models trained on our dataset (paraphrased SlimPajama) tend to achieve an optimal balance, compared to those trained on alternative datasets such as NIAH, Based, or UltraChat, when finetuned on top of the pretrained 430M Mamba model.

Along with hybrid models, we observe notable improvements in recall ability with minimal or no degradation in commonsense reasoning or language modeling performance when training non-hybrid models (models using only Mamba or only SWA layers) with a dataset of 4k sequence length and 40 million tokens. The mamba-only model showed a recall improvement rate of 29.5%, whereas the SWA-only model showed a more modest improvement of 17.7%. This suggests that the Mamba-only model, despite initially exhibiting weak recall performance due to underdeveloped recall capabilities during pretraining, has significant potential for recall when further trained. Prior to our additional training, the SWA-only model outperformed the Mamba-only model in recall (SWA:

13.8, Mamba: 12.5). However, after training, the Mamba-only model learned to better retain and recall information, resulting in a recall performance of 17.7, surpassing that of the SWA-only model (16.8). Furthermore, this improvement in recall did not come at the cost of commonsense reasoning. The Mamba-only model shows a 6.87% increase in commonsense reasoning, whereas the SWA-only model shows a 2.96% decline in commonsense reasoning ability. These results suggest that our method not only benefits hybrid models but also improves the performance of various model architectures, particularly those utilizing Mamba layers.

E.5 Comparison with DeciMamba

We evaluate performance on the LongBench dataset to compare our training approach with DeciMamba, using the same base model (Table 6). Model trained with our dataset consistently yields stronger results, particularly on QA datasets, with an average improvement of +8.1 points.

E.6 Generalization Across Different Mamba-2.8B Variants

Table 4 presents the performance of various base models trained using our paraphrased dataset. To ensure a fair comparison, we evaluate three variants of the Mamba-2.8B model: Mamba-2.8B, Mamba-2.8B-Ultrachat, and Mamba-2.8B-Zephyr. Our results show consistent improvements in both commonsense reasoning and recall performance when using the paraphrased dataset. Notably, the gains are most pronounced when using the original Mamba-2.8B model as the base, suggesting that models with fewer prior instruction-tuning steps may benefit more from paraphrased augmentation.

E.7 Length of Training Dataset

As shown in Figure 18, for both sequential and parallel architectures and various model sizes, **continually training with longer chunk size result in lower reduction rate on commonsense tasks and higher improvements on recall tasks**.

Upon closer inspection (Table 5), shorter chunk sizes (e.g., 2k) significantly boost performance on short-context recall tasks but lead to notable degradation on long-context tasks. This effect is particularly pronounced in parallel models. We hypothesize that this is because, as shown in Section D.4, parallel hybrid retains layer-wise characteristics more strongly than sequential models. Additionally, the gap in performance is more substantial for recall tasks (range of around -1% to 7%) than for commonsense tasks (range of around -1% to 3%).

E.8 Number of training dataset

Figure 20 shows the reduction rate of commonsense performance (left) and the improvement rate of recall performance (right) by the number of training token (x-axis), trained with a chunk size of 4k. As the training data size increases, we observe a general improvement in both commonsense and recall performance with convergence of around 80M to 100M tokens.

F CheckList

F.1 Potential Risks

Although our experiments are conducted on publicly available datasets, we do not apply additional data cleaning. As a result, the pretrained model may produce unexpected or unintended outputs due to noise or biases present in the data.

F.2 LLM Usage

We used the free version of ChatGPT-4o to assist with grammar checking during the writing of this paper.

		Commonsense Reasoning						Recall Ability							
Base Model	Type	LAM.	Hella.	PIQA	ARC	Wino.	***Avg.***	NQ-S	NQ-M	NQ-L	Drop	FDA	SWDE	TQA	***Avg.***
Mamba		69.1	49.5	75.3	64.1	63.2	63.7	31.0	28.1	21.7	20.9	29.6	41.0	64.6	33.8
	+ Ours	67.0	64.8	76.0	68.3	62.4	65.4	41.0	37.3	27.5	31.5	32.2	41.0	71.4	40.3
Mamba-U		67.0	70.5	78.6	65.9	65.2	65.6	36.3	35.0	27.7	25.7	34.3	50.1	70.5	39.9
	+ Ours	65.9	69.7	78.3	69.8	64.0	66.9	42.6	39.4	30.8	30.8	33.6	52.4	74.8	43.5
Mamba-Z		67.9	71.2	78.4	66.2	65.0	65.6	36.8	35.1	27.8	26.1	32.8	51.6	70.4	40.1
	+ Ours	66.8	69.7	77.8	70.2	67.8	69.0	42.4	38.5	30.6	31.3	34.2	34.9	74.3	40.9

Table 4: Performance of Mamba-2.8B when continually trained on our paraphrased dataset, evaluated across different base model variants. We observe consistent improvements in both commonsense reasoning and recall capabilities, with gains more pronounced for stronger base models (e.g., Mamba). "Mamba-U" and "Mamba-Z" refer to Mamba-2.8B-UltraChat and Mamba-2.8B-Zephyr, respectively.

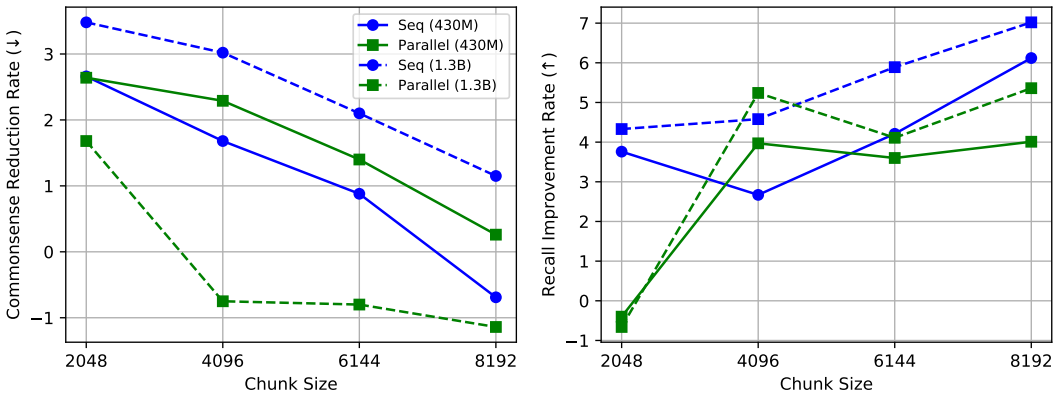

Figure 18: Commonsense reasoning reduction rate(left) and recall improvement rate(right) by changing the chunk size of the training dataset (x-axis).

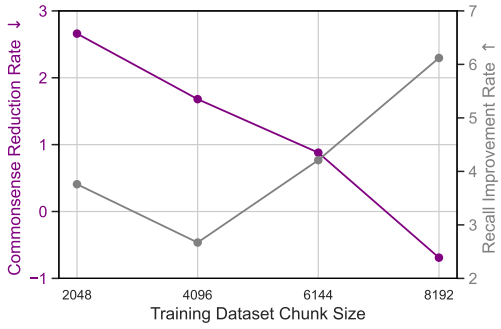

Figure 19: Commonsense reasoning reduction rate and recall improvement rate by changing the chunk size of the training dataset (x-axis).

Length	Commonsense	Recall (Short)	Recall (Long)	Recall (All)
Original	51.5	22.33	16.35	19.34
2k	50.1	24.3	16.4	19.8
4k	50.6	23.6	16.5	19.6
6k	51.0	23.7	17.0	19.9
8k	51.9	24.4	17.3	20.3

Table 5: Average commonsense and recall performance for short and long contexts as training dataset length (Length Column) varies in sequential hybrid (MFSF) training.

Benchmark	**Avg Len**	Mamba	DeciMamba	Mamba + Ours
2wikimqa	4887	3.9	**9.1**	8.0
Hotpotqa	9151	1.5	4.5	**12.5**
Musique	11214	0.9	1.7	**2.3**
Narrative QA	18409	0.9	1.7	**3.5**
Qasper	3619	5.97	**8.9**	8.5
Multifield QA	4559	11.2	18.6	**19.3**
GovReport	8734	9.8	14.9	**15.2**
QMSum	10614	**8.2**	7.1	7.3
MultiNews	2113	23.2	**24.6**	23.7
TriviaQA	8209	3.9	12.6	**36.0**
SAMSum	6258	**8.6**	7.3	6.9
TREC	5177	0.5	0.5	**27.0**
LCC	1235	8.1	8.7	**8.9**
RepoBench-p	4206	7.2	**11.0**	10.7
Passage Count	11141	0.0	**0.5**	0.0
Passage Ret. en	9289	0.0	1.5	**1.9**

Table 6: Performance over LongBench. Results of DeciMamba are from the paper (Ben-Kish et al., 2024). **Mamba+Ours** is model continual trained with our paraphrased dataset on the same base model (instruction-tuned Mamba-2.8b model). Ours tend to show high performance, especially on QA datasets.

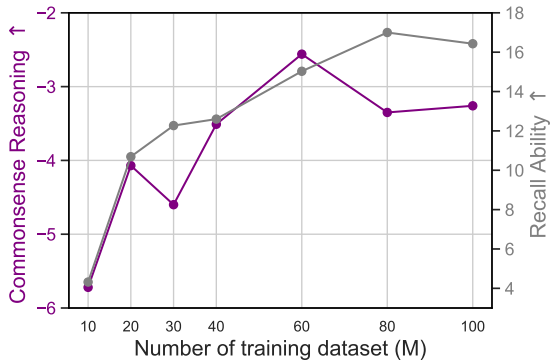

Figure 20: Commonsense reasoning and recall ability when changing the number of training dataset (x-axis).