ReMamba: Equip Mamba with Effective Long-Sequence Modeling

Anonymous ACL submission

Abstract

While the Mamba architecture demonstrates superior inference efficiency and competitive performance on short-context natural language processing (NLP) tasks, empirical evidence suggests its capacity to comprehend long contexts is limited compared to transformer-based models. In this study, we investigate the long-context efficiency issues of the Mamba models and propose ReMamba, which enhances Mamba's ability to comprehend long contexts. ReMamba incorporates selective compression and adaptation techniques within a two-stage re-forward process, incurring minimal additional inference costs overhead. Experimental results on the LongBench and L-Eval benchmarks demonstrate ReMamba's efficacy, improving over the baselines by 3.2 and 1.6 points, respectively, and attaining performance almost on par with same-size transformer models.

1 Introduction

Transformers (Vaswani et al., 2017), which form the backbone of most LLMs, encounter substantial challenges when dealing with long texts. The quadratic computational demands and the linear memory costs of the attention mechanism become prohibitive as the text length grows. This complexity poses a significant barrier to effectively modeling long texts, which is crucial for the development of LLMs. To address this, Mamba is proposed as a solution (Gu and Dao, 2024). Mamba models utilize a recurrent inference mode that ensures linear time complexity and compress information into the fixed state size, resulting in constant memory demands during inference. Furthermore, Mamba models eliminate the need for positional encoding, theoretically allowing them to handle inputs of any length. Mamba performs competitively against transformers on downsteam tasks. Shortly after, Mamba2 was introduced, simplifying the structured A matrix of Mamba to enable faster training and enlarged state size (Dao and Gu, 2024).

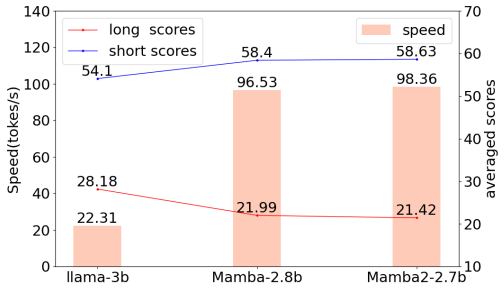

Figure 1: A comparison of pretrained Mamba models and Transformers of equivalent size across speed, short-context, and long-context performance metrics. Speed is measured under conditions of 6k input tokens and 1k output tokens. "short scores" represents the average accuracy across six tasks (HellaSwag, PIQA, Arc-E, Arc-C, WinoGrande, OpenbookQA) evaluated within the LM evaluation harness (Gao et al., 2023). "long scores" corresponds to the average scores on the LongBench-E benchmark (Bai et al., 2024). Notably, all LongBench evaluations employ a maximum token length of 2k to align with the model's training configuration.

Despite these advantages, some studies reveal that Mamba models do not perform as well as expected when dealing with long texts reaching 2k tokens or more (Waleffe et al., 2024). As depicted in Figure 1, our experimental findings reveal that the pretrained Mamba model surpasses pretrained Transformers of comparable size, such as llama-3b (Geng and Liu, 2023), on short-context tasks. Conversely, a substantial performance degradation is observed for Mamba on long-context tasks relative to Transformers. This performance disparity underscores a significant limitation of Mamba models in practical long-context applications.

This long-context deficency issue of Mamba is usually attributed to its RNN-like nature. This kind of architecture exhibits limitations in preserving crucial information from earlier input sequences as the context length increases due to the fixed-size memory (Wen et al., 2024; Yang et al., 2024b). Hy-

brid architectures (Lieber et al., 2024; Ren et al., 2024; Park et al., 2024) have sought to mitigate this issue by integrating attention mechanisms from transformers. However, these approaches often lead to decreased computational efficiency and increased memory consumption.

The key challenge in Mamba is the excessive degradation of distant information. ReMamba addresses this by employing an effective compression strategy that condenses the information and reduces the context length. Specifically, it selects the top-k hidden states during the first forward pass and integrates them into the state space using Mamba's selective mechanism in the second pass. ReMamba introduces minimal computational overhead (one additional forward pass) and maintains low, constant memory consumption. Experimental results demonstrate that our approach significantly improves Mamba's long-context performance, bringing it close to the performance of transformers. Our ReMamba model achieves a 3.2 improvement over the baseline on LongBench (Bai et al., 2024) and 1.6 improvement on L-Eval (An et al., 2023). Furthermore, our methodology exhibits transferability to Mamba2, yielding a 1.6 improvement on LongBench.

2 Related work

2.1 Mamba

The state space model chooses the time-invariant $\hat{A}$ (state transition matrix) and $\hat{B}$ (input coefficient matrix) thus lacking expressiveness and flexibility. Mamba (Gu and Dao, 2024) proposes to make $\hat{A}$ and $\hat{B}$ dynamically depend on inputs.

Recall that in one Mamba layer l , SSM states S are transformed as follows:

$$\Delta_{t-1}^{l} = \text{Softplus}\left(\text{Proj}_1(h_{t-1}^{l-1})\right), \tag{1a}$$

$$B_{t-1}^{l} = \text{Proj}_2\left(h_{t-1}^{l-1}\right), \tag{1b}$$

$$\hat{A}^{l}, \hat{B}_{t-1}^{l} = \text{discretize}\left(A^{l}, B_{t-1}^{l}, \Delta_{t-1}^{l}\right), \tag{1c}$$

$$h_{t-1}^{\prime l} = \text{Proj}_3\left(h_{t-1}^{l-1}\right), \tag{1d}$$

$$S_{t}^{l} = \hat{A}^{l} \otimes S_{t-1}^{l} + \hat{B}_{t-1}^{l}\left(h_{t-1}^{\prime l}\right)^{T}. \tag{1e}$$

Here, $h_{t-1}^{l-1} \in R^{H}$ represents the output hidden state of Mamba at layer $l-1$ and time step $t-1$. The Softplus function is denoted by Softplus, and Proj_1, Proj_2, and Proj_3 are abbreviations for multiple space projection operations.

Furthermore, $\Delta_{t-1}^{l} \in R^{H'}$ is the discrete time step corresponding to the selective mechanism in Mamba, where H' is the intermediate hidden size. The continuous and discrete state transformation matrices at layer l are given by $A^{l}, \hat{A}^{l} \in R^{H' \times N}$, respectively. The continuous and discrete input coefficient matrices are denoted by $B_{t-1}^{l}, \hat{B}_{t-1}^{l} \in R^{N \times 1}$. The state size is represented by N. The discretization method for computing $\hat{A}$ and $\hat{B}$ is indicated by "discretize". The vector $h_{t-1}^{\prime l} \in R^{H' \times 1}$ and the SSM state is represented by $S_{t}^{l} \in R^{H' \times N}$. The symbol $\otimes$ denotes element-wise multiplication, and $\hat{B}_{t-1}^{l}\left(h_{t-1}^{\prime l}\right)^{T}$ represents matrix multiplication. It is important to note that the definitions of A^{l} and $\hat{A}^{l}$ presented here differ from their original definitions due to Mamba's simplification to diagonal matrices.

2.2 Mamba2

Dao and Gu (2024) theoretically proves the connections between structured state space models and attention mechanisms. They also simplify structured matrix $\hat{A}$ further into scalar-times-identity structure and thus develop a new state space duality (SSD) framework with multi-head patterns similar to transformers.

2.3 Long Context Mamba and Transformers

Positional interpolation has been widely used as a technique to extend the context length of transformers (Chen et al., 2023; Peng et al., 2024; Ding et al., 2024). But they are specialized for transformers.

Mamba has been found to struggle in maintaining performance beyond its pretraining context length without additional training. LongMamba (Peiyuan, 2024) made the first successful attempt to extend Mamba's context length through a few hours of long-context fine-tuning. DeciMamba (Ben-Kish et al., 2024) aims to address the context extension problem of Mamba in a training-free manner, proposing a method to progressively reduce sequence length across layers by empirically removing unimportant tokens.

However, our experiments demonstrate that long-context fine-tuned Mamba still lags behind long-context fine-tuned transformers of the same size, despite using the same data. Moreover, DeciMamba2.8b appears to be insufficiently effective when evaluated on two widely used long-context benchmarks.

3 Pilot Study

KV cache compression is widely used in transformers to reduce memory consumption and improve inference speed. However, prompt compression often results in performance degradation compared to the full context lengths generation in transformers. Unlike transformers, Mamba does not employ a KV cache; instead, it utilizes a fixed-size state space in each layer to preserve context memory. A potential issue with Mamba is its tendency to forget distant information. In our pilot study, we hypothesize that the state space update in Mamba is insufficient for effectively compressing context information, and that techniques like prompt compression could help relieve this issue.

To explore this, we apply a simple prompt compression method: we replace part of the context tokens with a few randomly selected hidden states from the last layer of Mamba, creating a shorter prompt (referred to as random Mamba). This approach is similar to that of (Ge et al., 2024), which uses soft prompts for information compression. Intuitively, this random compression method may lead to significant information loss and degraded performance. However, our results show that the average scores for different context lengths on LongBench between normal Mamba and random Mamba are similar when both are trained on the same long-context dataset. Furthermore, random Mamba outperforms normal Mamba at certain context lengths, as shown in Figure 3 of 5.4.1. The random_select(SFT) represents the fine-tuned random Mamba, while Mamba(SFT) represents the fine-tuned normal Mamba.

This observation suggests that information loss in Mamba when handling long contexts is substantial. To relieve this, we propose selective compression and selective adaptation through leveraging Mamba's state space update mechanism.

4 Methodology

ReMamba consists of two forward stages. In the first stage, three feed-forward networks are employed to help determine the significance of hidden states from Mamba's final layer. These hidden states are selected based on their importance scores. The second stage integrates these compression hidden states with the input context, adapting Mamba's selective mechanism to incorporate them into the state space.

Our proposed method draws some spirits from techniques employed in KV cache compression (Mu et al., 2023; Ge et al., 2024; Yang et al., 2024a; Chevalier et al., 2023; Hwang et al., 2024; Gao et al., 2024) by leveraging the language model itself to aggregate information via hidden states and employing a scoring mechanism to select the most salient representations. Nevertheless, different from transformers, ReMamba's compression strategy focuses on two key objectives: 1) compressing and selectively retaining crucial information to minimize information degradation, and 2) reducing the frequency of state space updates to further alleviate the information loss.

The selection method employed in ReMamba is a simplified key-query-value-based approach, commonly used in Retrieval-Augmented Generation and summarization tasks (Lewis et al., 2020; Mao et al., 2022). In this context, we select the most important hidden states from the last layer of Mamba to mitigate information loss during the compression process.

4.1 Stage1 : Selective Compression

Selective compression involves selectively compressing the input prompt by leveraging the final layer hidden states of the Mamba model to decrease state updates and consolidate information.

Suppose the sequence length is L and the context token embeddings are $\{t_i\}_{i=1}^{L}$. We define the relative range to be compressed as $range := (s, e)$, where $e = s + p$, with s and e denoting the relative start and end positions, respectively, and p representing the relative length to compress. These values satisfy $0 \leq s, p, e \leq 1$. The index set of the context to compress is $\mathcal{R} := [S, E]$, where $S = L \cdot s + 1$ and $E = L \cdot (s + p)$. Consequently, the length of the prompt to compress is $L' = E - S + 1$. For convenience, we use $\mathcal{R}$ to represent both the set of indices and the set of actual tokens within the context to be compressed. Furthermore, we define the compression ratio ρ and compress the selected context $\mathcal{R}$ into $K := |\mathcal{R}| \cdot \rho$ hidden representations.

In Figure 2, the compression hyperparameter settings are: $s = 0.2$, $p = 0.4$, $range = (0.2, 0.6)$, $\mathcal{R} = [3, 6]$, $\rho = 0.5$, $K = 2$. In our experiments, we find that $s = 0$ yields the best results, which can be attributed to the casual language modeling nature of Mamba (this will be discussed in more details in A.3).

As shown in the Stage 1 of Figure 2, we denote the last layer's output hidden states as $\{h_i\}_{i=1}^{L}$, where each $h_i \in R^H$ with H representing the hid-

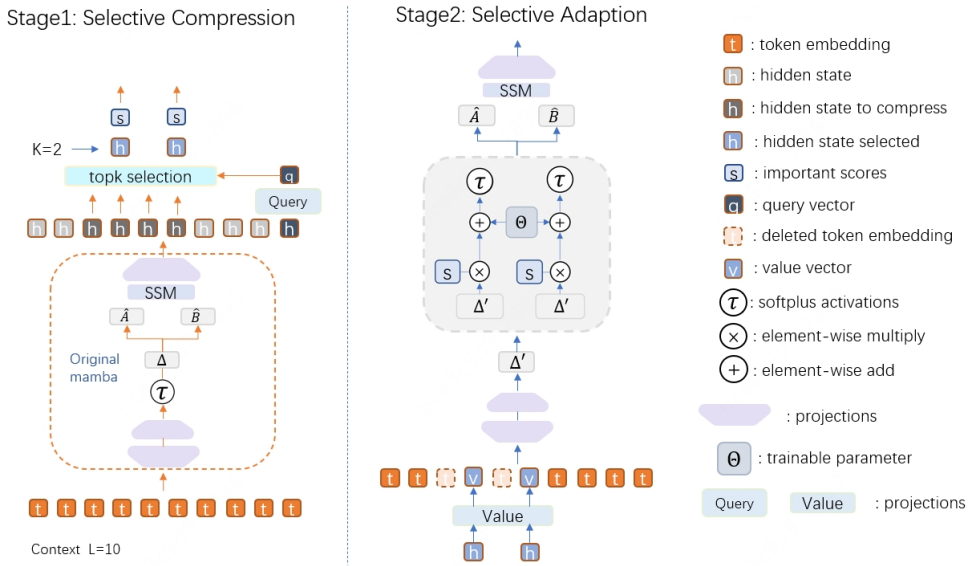

Figure 2: ReMamba architecture. We just show one layer and leave out the A, B and discrete method here. For Stage 2, only those value vectors selected need to go through selective adaption. Normal token embeddings just flow as usual. We select top-K (here is top-2) hidden states in the last layer according to their importance scores calculated with the last hidden state h_L. And we incorporate the scores into the gradient utilizing the selective mechanism in Mamba.

den size. We then transform the last hidden state h_L into a query hidden state, namely q, through a feed-forward layer named $Query$. Additionally, the hidden states to be compressed, denoted as $\{h_i\}_{i=S}^{E}$, are transformed into $\{k_i\}_{i=S}^{E}$ via a Key layer (this transformation is not shown in Figure 2). Finally, the cosine similarity scores, $Cos = \{cos_i\}_{i=S}^{E}$, are computed to serve as importance scores for the hidden states $\{h_i\}_{i=S}^{E}$. The calculation of q, k_i, and cos_i is formulated as follows:

$$\begin{aligned} q &= \text{Query}(h_L) \\ \{k_i\}_{i=S}^{E} &= \text{Key}(\{h_i\}_{i=S}^{E}) \\ cos_i &= \frac{k_i \cdot q}{\max(\|k_i\|_2 \cdot \|q\|_2, \epsilon)} \end{aligned} \tag{2}$$

where k_i represents the transformed hidden state at position i, and cos_i computes the cosine similarity between q and k_i. The constant ϵ prevents division by zero.

We select the top-K hidden states h_j, where $j \in G$, from the hidden states $\{h_i\}_{i=S}^{E}$ based on their importance scores, denoted by Cos. The index set G is defined as:

$$G = \underset{A \subset \{S, S+1, \ldots, E\}, |A|=K}{\arg\max} \sum_{i \in A} cos_i \tag{3}$$

Note that the original order of these indices is preserved.

In our model, after selecting the top-K hidden states h_j, we apply a feed-forward layer, $Value$, to project them into the token embedding hidden space:

$$\{v_i\}_{i=1}^{K} = V(\{h_j\}, j \in G) \tag{4}$$

Their corresponding cosine similarity scores are $\{cos'_i\}_{i=1}^{K}$. We then replace the token embeddings $\{t_i\}_{i=S}^{E}$ $(\mathcal{R})$ with $\{v_i\}_{i=1}^{K}$. Consequently, the new input embeddings for Mamba are replaced by:

$$T_{\text{new}} = \text{Cat}(\{t_i\}_{i=1}^{S-1}, \{v_i\}_{i=1}^{K}, \{t_i\}_{i=E+1}^{L}) \tag{5}$$

$$= \{t'_i\}_{i=1}^{L-L'+K} \tag{6}$$

where Cat denotes the concatenation operation. The length of T_{new} is $L - L' + K$, resulting in a significantly shorter input sequence for the second forward pass compared to the first.

4.2 Stage 2: Selective Adaption

One significant challenge in using top-K selection based on importance scores is its non-differentiability, which impedes the ability to train such models effectively. Here we propose a framework that integrates importance scores into the selective mechanisms of the Mamba model.

For hidden states (embeddings) that do not require compression in stage 1, namely $\{t_i\}_{i=1}^{S-1}$ and

$\{t_i\}_{i=E+1}^{L}$, the standard Mamba algorithm is applied during the second forward pass. For embeddings at selected positions, specifically $\{t_i'\}_{i=S}^{S+K-1}$ or equivalently $\{v_i\}_{i=1}^{K}$, Equation 1a is reformulated as follows:

$$\begin{aligned} \alpha &= \mathrm{ReLU}(cos_{t-1}^{'}) \\ {\Delta_{t-1}^{l}}^{'} &= \mathrm{Proj}_1(h_{t-1}^{l-1}) \\ \delta &= {\Delta_{t-1}^{l}}^{'} \cdot \alpha + \Theta^l \\ \Delta_{t-1}^{l} &= \mathrm{Softplus}(\delta) \end{aligned} \tag{7}$$

where $\Theta^l \in R^{H'}$ is a layer-wise trainable offset parameter controlling scale intensity. ReLU is the activation function. Intuitively, hidden states with low importance scores should minimally impact model computations. Therefore, we approximate this behavior by setting their corresponding Δ values close to zero. Ideally, directly multiplying Δ by α would be more precise, but this necessitates modifications to the selective scan algorithm, leading us to adopt the simpler approach.

4.3 Training

Following the forward encoding processes, standard causal language generation is applied using the Mamba architecture. During training, newly introduced parameters within the selective compression mechanism are optimized. These parameters, except for Θ which is initialized to all zeros, are initialized with a subset of the weights from the first layer's in_proj matrix. Additionally, for parameters in Mamba, the dt_proj matrix is fully trained, while in_proj, out_proj, embeddings, and lm_head are updated using Low-Rank Adaptation (LoRA) (Hu et al., 2022). In our best implementation, to emphasize the significance of specific information, gradients flowing into the importance scores are scaled proportionally to these scores. This approach intuitively prioritizes the training of more critical representations.

5 Experiments

5.1 Experimental Setups

Our model is designed for long-context question-answering tasks, necessitating a substantial corpus of long-context instruction tuning data. To this end, we leverage the OpenOrca dataset (Mukherjee et al., 2023) and LongAlpaca-12k (Chen et al., 2024). The former comprises a rich collection of ChatGPT-augmented FLAN data alignments, while the latter is a long-context alignment dataset. We initially filter long instruction tuning instances from OpenOrca and concatenate them with LongAlpaca. To accommodate device memory constraints, prompts are truncated to a maximum length of 6,000 tokens. This process yields approximately 200,000 long-context training examples. To augment training data diversity, the initial 300,000 standard instances from OpenOrca are incorporated. This dataset is referred to as the LongOrca dataset. We finetune the baseline Mamaba 2.8b model and our ReMamba model on the same dataset. We also finetune a DeciMamba2.8b (Ben-Kish et al., 2024) and a llama-3b (Geng and Liu, 2023) for reference. DeciMamba aims to address Mamba's context-extension issue without requiring additional training. Although our approach differs slightly in terms of settings and objectives, we still fine-tune DeciMamba2.8b using the same data. Given the 2k maximum positional encoding limit of llama-3b, we conduct fine-tuning experiments using the simple linear positional interpolation technique (Chen et al., 2023) to extend its context length. The data construction process for Llama-3b is identical to that of Mamba. Details can be found in A.1. Notice that Mamba2.8b is the largest model we can obtain.

5.2 Evaluations

We conduct comparative analyses of our model against baseline Mamba2.8b and DeciMamba2.8b (both of finetuned and pretrained) on the widely adopted LongBench benchmark (Bai et al., 2024) and LEval benchmark (An et al., 2023), which encompass a diverse set of challenging real-world long-context tasks. For consistency, the same prompt templates and greedy decoding configurations are employed across all models.

To provide a reference point, the performance of a similarly sized transformer architecture (llama-3b) is also included. Both the pretrained and fine-tuned evaluations of Llama-3b utilize the linear positional interpolation technique.

5.3 Results

Results on LongBench We choose the English branch of LongBench because our training set only contains English. Higher values across all indicators are indicative of better performance. We compare the performance of the models in detailed tasks in Table 1 under the max length 6k corresponding to the training setting. Here the hyperparameters for ReMamba are: $s = 0$, $p = 0.18$

Model	2WikiMQA	GovReport	HotpotQA	LCC	MultiNews	MultiQA	PassCount	PassRetrie.	Qasper	RepoBench	SAMSum	TREC	TriviaQA	Average
llama-3b (Pre)	4.07	10.33	3.93	43.17	6.96	7.74	1.78	7.13	3.49	32.14	9.98	15.0	30.65	13.57
llama-3b (SFT)	15.69	24.55	19.69	56.17	19.37	29.73	0.33	6.67	19.73	44.47	33.37	48.67	58.41	28.99
DeciMamba (Pre)	3.89	9.36	3.85	25.69	11.62	4.91	0.83	1.19	3.21	13.88	6.79	6.67	17.72	8.43
DeciMamba (SFT)	19.6	13.74	15.32	23.65	10.91	17.88	1.37	4.96	5.72	12.54	10.59	30.33	46.06	16.36
Mamba (Pre)	3.73	8.72	4.03	24.03	11.31	4.95	0.80	1.75	3.67	12.83	6.86	9.00	17.40	8.39
Mamba (SFT)	22.10	19.08	15.90	40.20	19.36	30.28	0.00	4.67	19.04	36.02	28.30	39.33	45.97	24.63
ReMamba (SFT)	21.18	19.67	20.56	48.21	18.86	26.39	3.21	6.83	16.76	40.40	33.65	48.67	57.73	27.86

Table 1: Performance on LongBench-E (English branch). “MultiQA” denotes MultiFieldQA , “PassCount” denotes PassageCount, “PassRetrie.” denotes PassageRetrieval. Models are evaluated using a maximum length of 6K tokens, matching their finetuning configurations. Here “(Pre)” means pretrained model. “(SFT)” means finetuned model.

Model	Finetuned	Tokens	CodeU	Coursera	GSM	QuALITY	SFictio	TOEFL	Average
llama-3b (Pre)	✗	6k	0.0	24.71	3.0	27.23	32.81	17.47	17.54
llama-3b (SFT)	✓	6k	1.11	19.62	7.0	24.26	57.03	27.14	22.69
DeciMamba (Pre)	✗	6k	2.22	24.71	0.0	25.25	24.6	16.73	15.59
DeciMamba (SFT)	✓	6k	0.0	23.69	0.0	26.86	53.17	22.68	21.07
Mamba (Pre)	✗	6k	2.22	23.26	0.00	25.74	23.44	17.10	15.29
Mamba (SFT)	✓	6k	4.44	26.16	1.00	27.72	50.78	23.05	22.19
ReMamba (SFT)	✓	6k	2.22	22.97	3.00	25.74	58.59	30.48	23.83

Table 2: Model performance on closed-ended tasks of L-Eval. “Tokens” denotes the max length. “SFT” denotes finetuned models. “Pre“ denotes pretrained models.

and $\rho = 0.009$. We will also show later that our model’s robustness to various of hyperparameter combinations. Table 1 shows that our ReMamba model improves the average scores on LongBench 3.23 compared to the SFT Mamba baseline. Our model approaches the pretrained and finetuned transformer baseline. The results for DeciMamba indicate that it may not be sufficiently effective for tasks in LongBench or it may be sensitive to the choosing of hyperparameters.

Results on LEval We compare the performance on the closed-ended tasks of L-Eval. The higher all indicators are, the better. A snap of detailed task scores for the maximum length of 6k is presented in Table 2. We can witness a 1.64 improvement on average scores compared to the SFT Mamba baseline. Here the hyperparameter setting for ReMamba is: $s = 0$, $p = 0.20$ and $\rho = 0.05$. The results for DeciMamba2.8b also show no significant improvements.

5.4 Analyses and Discussions

5.4.1 Ablation Study

To verify the effectiveness of the modules we introduced, we conduct an ablation study by comparing ReMamba against three alternative methods: 1. Random Selection: which randomly select hidden states as the compressed information according to ρ. 2. Fix Selection: given the ρ we select enough hidden states every k positions. The interval k is calculated based on the compression ratio. 3. Multiplicative Selection: This variant just modifies the selective adaptation process by directly multiplying importance scores with the selected hidden states, aligning with the approach proposed by Raposo et al. (2024). All of those models are trained on the same data as ReMamba.

We report the averaged scores on LongBench across various maximum input lengths. As illustrated in Figure 3, both the fixed and random selection methods achieve performance comparable to the finetuned Mamba baseline. Interestingly, these methods even outperform Mamba at lengths of 5k and 6k. This observation confirms our hypothesis that Mamba models suffer from severe forgetting issues. Even simple methods like dropping some information appear beneficial. The performance of the multiplicative selection method shows some improvements across varying input lengths. However, the substantial performance gap observed with our selective adaptation module demonstrates its critical role in the ReMamba model. The selective

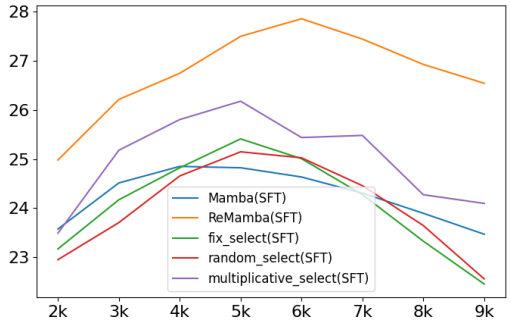

Figure 3: Ablation study about average scores on LongBench varying max length from 2k to 9k. "Mamba(SFT)" is the finetuned Mamba. "fix_select" is the Fix Selection. "random_select" is the Random Selection. "multiplicative_select" is the Multiplicative Selection.

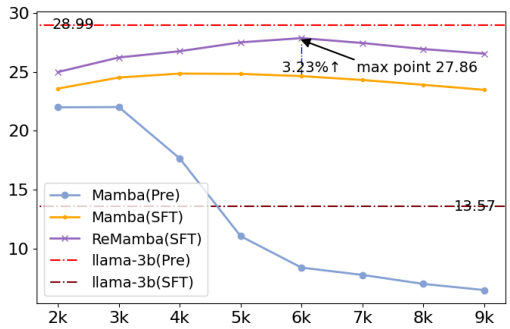

Figure 4: Average scores on LongBench varying max length from 2k to 9k. The "Pre" means pretrained model while "SFT" means finetuned model. The performance of llama-3b (SFT) and llama-3b (Pre) is for reference, using the max length of 6k.

adaptation module not only mitigates the forgetting problem, but also significantly enhances the model's ability to handle longer input sequences effectively.

5.4.2 Varying Length

To complement our main results, which employ a maximum sequence length of 6k tokens to align with training settings, we further evaluate the model performance at varying input lengths ranging from 2k to 9k tokens. This evaluation is conducted using the LongBench and L-Eval benchmarks. As depicted in Figure 4, our ReMamba consistently outperforms the baseline Mamba model across all tested context lengths on LongBench. Notably, the performance gap between our model and the baseline widens as the context length increases. Furthermore, our model extends the efficient context length (the length at which greatest performance is observed) to 6k tokens, compared to 4k tokens for the finetuned Mamba baseline. In Figure 5, we observe performance improvements across all context lengths for our model on L-Eval.

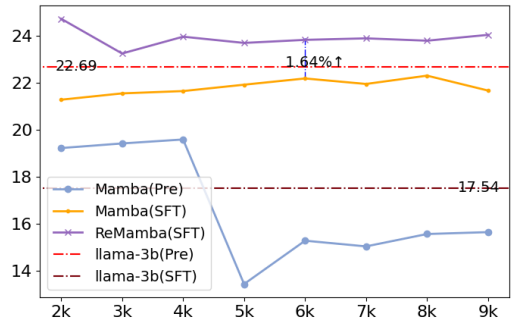

Figure 5: Average scores on L-Eval varying max length from 2k to 9k. The performance of llama-3b (SFT) and llama-3b (Pre) is for reference, using the max length of 6k.

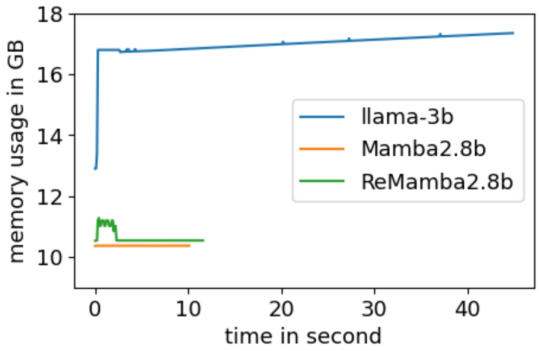

Figure 6: Memory consumption comparisons during inference are conducted with a 6144 input sequence and a 1024 output sequence, using a batch size of 1. The experiments are done on an A100 80GB GPU.

5.4.3 Speed Performance and Memory Expense

Our model introduces a single additional forward pass during inference, resulting in small constant memory consumption. We visualize the memory consumption during inference with a 6k input sequence and a 1k output sequence, using a batch size of 1 in Figure 6. The device we use is an A100 80GB GPU. We observe that the encoding process of Llama consumes a substantial amount of memory and the KV cache increases gradually during the decoding process, whereas ReMamba incurs only a small additional memory cost. After the encoding process, ReMamba's memory con-

Model	2WikiMQA	GovReport	HotpotQA	LCC	MultiNews	MultiQA	PassCount	PassRetrie.	Qasper	RepoBench	SAMSum	TREC	TriviaQA	Average
Mamba2 (Pre)	2.18	4.76	1.54	23.46	7.71	2.88	0.60	1.47	1.17	14.97	2.07	8.33	10.60	6.29
Mamba2 (SFT)	13.73	19.97	15.05	41.78	19.52	25.20	0.67	5.67	13.44	38.79	33.95	49.67	43.54	24.69
ReMamba2 (SFT)	18.90	19.03	18.09	51.15	17.68	25.82	3.33	5.00	14.84	43.99	23.55	44.00	56.90	26.33

Table 3: The performance comparisons of LongBench-E (English Branch) on Mamba2. Mamba2 (Pre) means pretrained Mamba2. Mamba2 (SFT) means finetuned Mamba2. ReMamba2 (SFT) means our model. All use the setting of 6k max length.

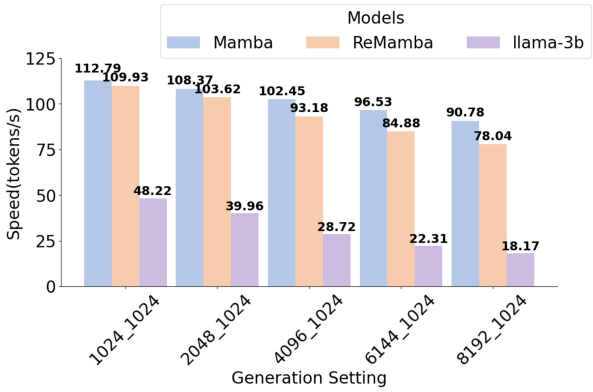

Figure 7: Speed (tokens/second) performance comparisons. Here 1024_1024 means input 1024 tokens and output 1024 tokens.

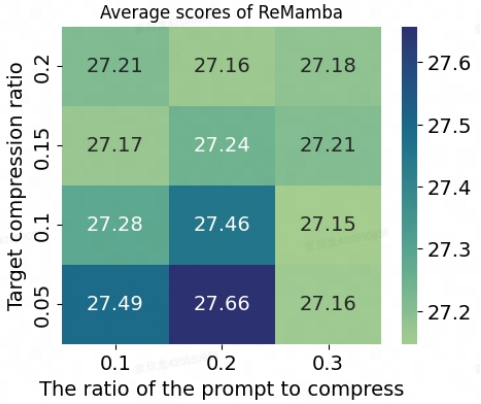

Figure 8: Robustness of the ReMamba model with varying hyperparameters. The row label denotes the relative ratio of the prompt to be compressed, corresponding to parameter p. The column label indicates the compression ratio, corresponding to parameter ρ.

sumption stabilizes at a constant level, which is moderately higher than Mamba's, corresponding to the additional parameters introduced to support the selection mechanism.

To evaluate the speed performance, we varys the input sequence length from 1k to 8k tokens while fixing the output length at 1k tokens. For all the experiments, we use a batch size of 1 and measure the speed on an NVIDIA A100 80GB GPU. We compare the performance of ReMamba, Mamba, and the vanilla transformer model (llama-3b), as illustrated in Figure 7. The speed metric is given in tokens per second. Our experiments indicate that ReMamba operates at speeds comparable to the original baseline, maintaining a significant speed advantage over traditional transformers.

5.4.4 Robustness varying choices of hyperparamters

The aforementioned results were obtained using the hyperparameter settings $s = 0$, $p = 0.18$, and $\rho = 0.009$, which demonstrates relatively superior performance. In Figure 8, we also show the stability of our model by varying the hyperparameters p and ρ. For these experiments, the hyperparameter s is fixed at 0.

5.4.5 Generalizing to Mamba2

While our method is specifically tailored for Mamba, we also conduct experiments to verify its applicability to Mamba2. As is shown in Table 3, we achieves 1.6 improved performance on LongBench. More details can be found in A.2.

6 Conclusions

This study investigates the long-context efficiency challenges posed by Mamba models, hypothesizing that distant information within these models is subject to substantial degradation. In response, we introduce ReMamba, a novel approach that compresses and selectively preserves critical information during an initial forward pass. This compressed information is subsequently integrated into the state space during a second forward pass, capitalizing on Mamba's inherent selective mechanism. Notably, ReMamba incurs minimal computational overhead while substantially enhancing Mamba's long-context performance, thereby offering a promising avenue for advancing the Mamba model family.

Limitations

Although ReMamba improves the long-context performance of Mamba, it is unlikely that Mamba can outperform transformers as the context length increases, due to its fixed-size state space. Additionally, ReMamba primarily relieve the loss of long-context information in Mamba through selective compression. A more promising approach would be to directly modify the state space update mechanism.

References

Chenxin An, Shansan Gong, Ming Zhong, Xingjian Zhao, Mukai Li, Jun Zhang, Lingpeng Kong, and Xipeng Qiu. 2023. L-eval: Instituting standardized evaluation for long context language models. *Preprint*, arXiv:2307.11088.

Yushi Bai, Xin Lv, Jiajie Zhang, Hongchang Lyu, Jiankai Tang, Zhidian Huang, Zhengxiao Du, Xiao Liu, Aohan Zeng, Lei Hou, Yuxiao Dong, Jie Tang, and Juanzi Li. 2024. Longbench: A bilingual, multitask benchmark for long context understanding. *Preprint*, arXiv:2308.14508.

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.

Shouyuan Chen, Sherman Wong, Liangjian Chen, and Yuandong Tian. 2023. Extending context window of large language models via positional interpolation. *Preprint*, arXiv:2306.15595.

Yukang Chen, Shengju Qian, Haotian Tang, Xin Lai, Zhijian Liu, Song Han, and Jiaya Jia. 2024. Longlora: Efficient fine-tuning of long-context large language models. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

Alexis Chevalier, Alexander Wettig, Anirudh Ajith, and Danqi Chen. 2023. Adapting language models to compress contexts. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023*, pages 3829–3846. Association for Computational Linguistics.

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

Yiran Ding, Li Lyna Zhang, Chengruidong Zhang, Yuanyuan Xu, Ning Shang, Jiahang Xu, Fan Yang, and Mao Yang. 2024. Longrope: Extending LLM context window beyond 2 million tokens. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net.

Jun Gao, Ziqiang Cao, and Wenjie Li. 2024. Selfcp: Compressing over-limit prompt via the frozen large language model itself. *Preprint*, arXiv:2405.17052.

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. 2023. A framework for few-shot language model evaluation. https://zenodo.org/records/10256836. Accessed: 2024-May-29.

Tao Ge, Jing Hu, Lei Wang, Xun Wang, Si-Qing Chen, and Furu Wei. 2024. In-context autoencoder for context compression in a large language model. *Preprint*, arXiv:2307.06945.

Xinyang Geng and Hao Liu. 2023. Openllama: An open reproduction of llama. https://github.com/openlm-research/open_llama. Accessed: 2024-May-29.

Albert Gu and Tri Dao. 2024. Mamba: Linear-time sequence modeling with selective state spaces. *Preprint*, arXiv:2312.00752.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. Lora: Low-rank adaptation of large language models. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net.

Dongseong Hwang, Weiran Wang, Zhuoyuan Huo, Khe Chai Sim, and Pedro Moreno Mengibar. 2024. Transformerfam: Feedback attention is working memory. *Preprint*, arXiv:2404.09173.

Patrick S. H. Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Opher Lieber, Barak Lenz, Hofit Bata, Gal Cohen, Jhonathan Osin, Itay Dalmedigos, Erez Safahi, Shaked Meirom, Yonatan Belinkov, Shai Shalev-Shwartz, Omri Abend, Raz Alon, Tomer Asida, Amir Bergman, Roman Glozman, Michael Gokhman, Avashalom Manevich, Nir Ratner, Noam Rozen, Erez Shwartz, Mor Zusman, and Yoav Shoham. 2024. Jamba: A hybrid transformer-mamba language model. *Preprint*, arXiv:2403.19887.

Ziming Mao, Chen Henry Wu, Ansong Ni, Yusen Zhang, Rui Zhang, Tao Yu, Budhaditya Deb, Chenguang Zhu, Ahmed Awadallah, and Dragomir Radev. 2022.

DYLE: Dynamic latent extraction for abstractive long-input summarization. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1687–1698, Dublin, Ireland. Association for Computational Linguistics.

Jesse Mu, Xiang Li, and Noah Goodman. 2023. Learning to compress prompts with gist tokens. In *Advances in Neural Information Processing Systems*, volume 36, pages 19327–19352. Curran Associates, Inc.

Subhabrata Mukherjee, Arindam Mitra, Ganesh Jawahar, Sahaj Agarwal, Hamid Palangi, and Ahmed Awadallah. 2023. Orca: Progressive learning from complex explanation traces of gpt-4. *Preprint*, arXiv:2306.02707.

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. *Preprint*, arXiv:2402.04248.

Zhang Peiyuan. 2024. Longmamba. https://github.com/jzhang38/LongMamba. Accessed: 2024-August-10.

Bowen Peng, Jeffrey Quesnelle, Honglu Fan, and Enrico Shippole. 2024. Yarn: Efficient context window extension of large language models. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

David Raposo, Sam Ritter, Blake Richards, Timothy Lillicrap, Peter Conway Humphreys, and Adam Santoro. 2024. Mixture-of-depths: Dynamically allocating compute in transformer-based language models. *Preprint*, arXiv:2404.02258.

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. *Preprint*, arXiv:2406.07522.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Roger Waleffe, Wonmin Byeon, Duncan Riach, Brandon Norick, Vijay Korthikanti, Tri Dao, Albert Gu, Ali Hatamizadeh, Sudhakar Singh, Deepak Narayanan, Garvit Kulshreshtha, Vartika Singh, Jared Casper, Jan Kautz, Mohammad Shoeybi, and Bryan Catanzaro. 2024. An empirical study of mamba-based language models. *Preprint*, arXiv:2406.07887.

Kaiyue Wen, Xingyu Dang, and Kaifeng Lyu. 2024. Rnns are not transformers (yet): The key bottleneck on in-context retrieval. *arXiv preprint arXiv:2402.18510*.

Dongjie Yang, XiaoDong Han, Yan Gao, Yao Hu, Shilin Zhang, and Hai Zhao. 2024a. Pyramidinfer: Pyramid kv cache compression for high-throughput llm inference. *Preprint*, arXiv:2405.12532.

Kai Yang, Jan Ackermann, Zhenyu He, Guhao Feng, Bohang Zhang, Yunzhen Feng, Qiwei Ye, Di He, and Liwei Wang. 2024b. Do efficient transformers really save computation? *arXiv preprint arXiv:2402.13934*.

A Appendix

A.1 Training Details

During training, the ReMamba hyperparameter s is fixed at 0. The hyperparameter p is randomly sampled from the interval [0.1, 0.3], while ρ is randomly sampled from the interval [0.05, 0.2].

Most of DeciMamba's experiments are conducted on Mamba130m. The only configuration for DeciMamba2.8 provided in the original paper is decimating_layers = 22, decimation_max_p_L_base = 4000, used for language modelling tasks. Here we change the decimation_max_p_L_base = 6000. For all other hyperparameters, we use the default settings: decimation_min_seq_len = 20, decimation_beta = 0.5, and decimation_type = "max_p". The linear positional interpolation configurations for llama-3B are: max_position_embeddings = 6144 and factor = 3. The data construction process for Llama-3b is identical to that of Mamba. We finetune all the models for 1 epoch using DeepSpeed Zero Stage 3 on 8 A100-80GB GPUs. Other training details can be found in the Table 4.

A.2 Mamba2 Details

The hyperparameters here are: $s = 0$, $p = 0.25$ and $\rho = 0.05$. The max length is still 6k. It is noteworthy that Mamba2 exhibits nearly no performance improvement over Mamba on LongBench, suggesting potential limitations within the Mamba model series.

A.3 Why compress from the start

Experimental results indicate that setting $s = 0$ is the best. However, one might wonder about the effectiveness of compressing in the middle of the sequence. We conduct additional analytical studies to explore the impact of compressing the input sequence from different starting positions.

We train a model utilizing s sampled uniformly from the interval [0.1, 0.3] during the training process. Subsequently, we evaluate its performance

Model	dataset	epoch	optimizer	learning scheduler	learning rate	warm up steps	LoRA rank
llama2-3b (SFT)	LongOrca	1	AdamW	linear	2e-5	0	32
DeciMamba (SFT)	LongOrca	1	AdamW	linear	2e-5	0	32
Mamba (SFT)	LongOrca	1	AdamW	linear	2e-5	0	32
ReMamba (SFT)	LongOrca	1	AdamW	cosine	2e-5	0	32
Mamba2 (SFT)	LongOrca	1	AdamW	cosine	2e-5	0	32
ReMamba2 (SFT)	LongOrca	1	AdamW	cosine	2e-5	0	32

Table 4: Model training details

Model	ρ	p	s	model_type	average
ReMamba	0.009	0.18	0.00	ReMamba	27.86
middle0.0	0.009	0.18	0.00	middle	25.96
middle0.1	0.009	0.18	0.10	middle	26.45
middle0.2	0.009	0.18	0.20	middle	26.86
middle0.3	0.009	0.18	0.30	middle	26.56
middle0.4	0.009	0.18	0.40	middle	26.43
special	0.009	1.00	1.00	special	15.76

Table 5: Performance of different model variants on LongBench. Parameters s, p, and ρ represent the relative start position, relative length to compress, and compression ratio, respectively. In this context, the "ReMamba" model type constitutes our optimal model. The "middle" type corresponds to the model variant where s is non-zero. The "special" model variant compresses the entire prompt using $\rho = 0.009$ and subsequently appends the compressed hidden states to the end of the original prompt in the second stage.

on LongBench under conditions identical to those of the ReMamba model, employing a maximum length of 6k tokens, $p = 0.18$, and $\rho = 0.009$. We evaluate the average scores ranging s from 0 to 0.4. Additionally, we train a special model variant that compresses the entire prompt based on $\rho = 0.009$ and appends the compressed hidden states to the end of the original prompt in the second stage.

Table 5 presents the results of these experiments. We observe a performance degradation when the compression is applied in the middle of the sequence. The special model variant performs even worse than the finetuned Mamba baseline.

This degradation can be explained by the disruption caused to the causal language modeling nature of the Mamba model. When compressed information is integrated into the initial position, the subsequent language modeling process can proceed without modification, effectively treating the compressed data as a specialized non-zero initial state. Conversely, inserting those compressed hidden states as tokens within the sequence disrupts the causal language modeling paradigm, which assumes complete sentences as input. This incongruity hinders the model's ability to maintain a coherent state space and can lead to performance degradation. Among the tested models, the special model variant that appends compressed hidden states to the end of the original prompt exhibits the most pronounced negative impact due to the significant disruption of the model's expected input structure.

Despite these challenges, the model that compresses in the middle still outperforms the finetuned Mamba baseline. This demonstrates that our method exhibits apparent effectiveness.