Rethinking Token Reduction for State Space Models

Zheng Zhan[1*], **Yushu Wu**[1*], **Zhenglun Kong**[12*], **Changdi Yang**[1], **Yifan Gong**[1], **Xuan Shen**[1], **Xue Lin**[1], **Pu Zhao**[1], **Yanzhi Wang**[1]
[1]Northeastern University, [2]Harvard University
{zhan.zhe, wu.yushu, p.zhao, yanz.wang}@northeastern.edu

Abstract

Recent advancements in State Space Models (SSMs) have attracted significant interest, particularly in models optimized for parallel training and handling long-range dependencies. Architectures like Mamba have scaled to billions of parameters with selective SSM. To facilitate broader applications using Mamba, exploring its efficiency is crucial. While token reduction techniques offer a straightforward post-training strategy, we find that applying existing methods directly to SSMs leads to substantial performance drops. Through insightful analysis, we identify the reasons for this failure and the limitations of current techniques. In response, we propose a tailored, unified post-training token reduction method for SSMs. Our approach integrates token importance and similarity, thus taking advantage of both pruning and merging, to devise a fine-grained intra-layer token reduction strategy. Extensive experiments show that our method improves the average accuracy by 5.7% to 13.1% on six benchmarks with Mamba-2 compared to existing methods, while significantly reducing computational demands and memory requirements.[1]

1 Introduction

There are growing research interests and efforts in SSMs in recent years. Building on the foundation laid by the Kalman filter model (Kalman, 1960), SSMs have evolved to address long-range dependencies and are optimized for parallel training. Several works (Gu et al., 2021a,b, 2022; Gupta et al., 2022; Dao and Gu, 2024) have proposed SSM-based models capable of processing sequence data across a variety of tasks and modalities.

A notable recent contribution, Mamba (Gu and Dao, 2023a), integrates time-varying parameters into SSMs, allowing the model to selectively propagate or forget information. Additionally, Mamba introduces a hardware-aware parallel algorithm that accelerates both training and inference. Unlike quadratic attention mechanisms, which become prohibitively expensive with longer sequence lengths, Mamba's subquadratic-time architecture is more efficient and better suited for handling long sequences. The exceptional scaling performance of Mamba underscores its potential as an effective alternative to the Transformer model (Vaswani et al., 2017) for generative language modeling tasks.

In line with existing research efforts aimed at enhancing the efficiency of Transformer models (Shen et al., 2024b,c; Zhan et al., 2021), exploring the efficiency of SSMs is crucial for facilitating real-time applications. While weight pruning and quantization are prevalent techniques for optimizing Transformer models (Vaswani et al., 2017; Yang et al., 2023; Zhang et al., 2022), token reduction (Rao et al., 2021; Pan et al., 2021; Yuan et al., 2021; Renggli et al., 2022) has proven effective in improving Transformer efficiency due to the token length dimension or number of token is independent of the model architecture.

Given that SSM blocks also process input tokens similarly to Transformer models, applying existing state-of-the-art (SOTA) token reduction techniques (Liang et al., 2022; Cao et al., 2023; Bolya et al., 2023) to SSMs appears to be a straightforward post-training approach to enhance their efficiency, especially when scaling to billions of model parameters. This can achieve faster serving and lower peak memory usage, facilitating the wider deployment of large-scale SSMs like Mamba. However, as illustrated in Figure 1, this application of token reduction to SSMs, while offering some benefits of faster inference with fewer tokens, results in significant performance drops.

In this paper, after applying existing Transformer token reduction techniques to SSMs and observing

[*]Equal contribution.
[1]Code available at https://github.com/wuyushuwys/ToR_SSM

Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing, pages 1686–1697
November 12-16, 2024

their failures, we conduct an insightful analysis to understand the patterns and reasons for their failures on SSMs. Based on our analysis, we propose a unified post-training token reduction method for SSMs to preserve performance and improve efficiency. We first employ a decoupling strategy that computes the importance of each token and classifies them into two sets: less important tokens and more important tokens. Following this, we devise a fine-grained intra-layer token reduction strategy for the hidden states and residual connections of Mamba. Our approach uses a hybrid token reduction strategy (combining and taking advantages of pruning and merging) on hidden state tokens, meticulously designed to balance preserving essential information and eliminating redundancy. Our unified strategy can be generalized to other model architectures like Transformers. In summary, the main contributions of our work are as follows:

- We observe the failure of directly applying token reduction techniques from Transformers to SSMs, and we conduct an insightful analysis to investigate the patterns of token reduction strategies and the possible reasons for their failures.
- We are the first to propose a unified post-training token reduction method designed for SSMs. This strategy leverages insights from both token pruning and token merging, and incorporates the token importance and similarity evaluation.
- Zero-shot evaluations on various SSMs demonstrate the effectiveness of our method, improving average accuracy by 5.7% to 13.1% on six benchmarks with Mamba-2, and by 6.5% to 15.1% with Mamba compared to baseline methods. Meanwhile, our method significantly reduces computational demands and memory requirements.

2 Related Work

State Space Models. SSMs (Gu and Dao, 2023b; Mehta et al., 2022; Wang et al., 2023) are emerging architecture designs for sequence-to-sequence transformation. The design has the strength to model complex systems by focusing on how the input, output, and state variables evolve over time. Mamba-2 (Dao and Gu, 2024) propose state space duality to design a new architecture whose core layer is a refinement of selective SSM. S4ND (Nguyen et al., 2022) is the first work that applies the state space mechanism to visual tasks and shows the potential to achieve competitive performance with ViTs (Dosovitskiy et al., 2020). ViM (Zhu et al., 2024) proposes a novel vision backbone with bidirectional selective SSM. The accomplishments demonstrate the potential of SSMs as an emerging foundation model family.

Token Reduction. Token reduction is an effective strategy to enhance computational efficiency by reducing the number of processed tokens or patches (Modarressi et al., 2022; Huang et al., 2022; Nawrot et al., 2022; Wang and Yu, 2023; Kong et al., 2023; Zhan et al., 2024). It enables significant acceleration without requiring additional weights or specialized hardware, aiming to selectively retain the most informative tokens. Several innovative approaches have been developed for Transformers. For example, EViT (Liang et al., 2022) uses the attentiveness of the [CLS] token with respect to other tokens to identify the most important tokens. DynamicViT (Rao et al., 2021) and SPViT (Kong et al., 2022) add layers that employ the Gumbel-Softmax trick to selectively prune less informative tokens. Agile-Quant (Shen et al., 2024a) leverage the activation-aware token pruning technique to reduce the outliers for LLMs. ToMe (Bolya et al., 2023) measures dot product similarity between token keys to determine redundancy and merge accordingly. PuMer (Cao et al., 2023) proposed a token reduction framework for large-scale VLMs with text-informed pruning and modality-aware merging strategies to progressively reduce the tokens of input image and text.

However, the dynamics of information flow between tokens and the learning mechanisms in models like Mamba (Gu and Dao, 2023b) remain largely unexplored. The absence of attention layers in Mamba makes current token reduction methods ineffective. Furthermore, the inclusion of the SSM module prevents the effective use of existing token reduction methods.

3 Preliminary and Motivation

3.1 State Space Models

SSMs are sequential models that map an input sequence $x(t) \in \mathbb{R}^L$ to an output sequence $y(t) \in \mathbb{R}^L$ through a hidden state $h(t) \in \mathbb{R}^N$ as follows,

$$h'(t) = \mathbf{A}h(t) + \mathbf{B}x(t), \quad y(t) = \mathbf{C}h(t), \quad (1)$$

where L denotes the length of the sequence, N denotes the number of representation dimensions, $\mathbf{A} \in \mathbb{R}^{N\times N}$ is the evolution matrix, and $\mathbf{B} \in \mathbb{R}^{N\times L}$, $\mathbf{C} \in \mathbb{R}^{L\times N}$ are the projection matrices.

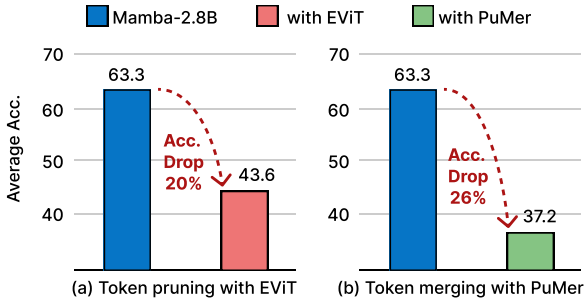

Figure 1: Performance of applying token pruning (EViT) and merging (PuMer) methods on Mamba-2.8B, showcasing significant drop in accuracy.

Mamba (Gu and Dao, 2023b) represents a discrete version of the continuous system for SSMs and incorporates a timescale parameter Δ to facilitate the transformation of continuous parameters with the zero-order hold (ZOH) as $\overline{\mathbf{A}} = \exp(\Delta\mathbf{A})$, and $\overline{\mathbf{B}} = (\Delta\mathbf{A})^{-1}(\exp(\Delta\mathbf{A}) - \mathbf{I}) \cdot \Delta\mathbf{B}$. After obtaining the discretized $\overline{\mathbf{A}}$ and $\overline{\mathbf{B}}$, the discretization of Equation (1) can be rewritten as,

$$\mathbf{h}_t = \overline{\mathbf{A}}\mathbf{h}_{t-1} + \overline{\mathbf{B}}\mathbf{x}_t, \quad \mathbf{y}_t = \mathbf{C}\mathbf{h}_t. \tag{2}$$

Finally, the Mamba model computes the output through a global convolution as follows,

$$\begin{aligned} \overline{\mathbf{K}} &= (\mathbf{C}\overline{\mathbf{B}}, \mathbf{C}\overline{\mathbf{A}\mathbf{B}}, \ldots, \mathbf{C}\overline{\mathbf{A}}^{\mathbf{L}-1}\overline{\mathbf{B}}), \\ \mathbf{y} &= \mathbf{x} * \overline{\mathbf{K}}, \end{aligned} \tag{3}$$

where $\mathbf{y}$ denotes the output sequence, L denotes the length of the input sequence $\mathbf{x}$, and $\overline{\mathbf{K}} \in \mathbb{R}^L$ denotes a structured convolutional kernel.

3.2 Analysis of Reasons Behind the Failure of Token Reduction on SSMs

Due to the SSMs' reliance on a sequential strategy for token computation, the previous token reduction strategies highlighted in Figure 1 do not yield effective results. In this section, we delve into the reasons why directly applying SOTA token pruning or merging method fails on SSMs.

Failure of token pruning on SSMs. Existing SOTA token pruning methods for Transformers, such as Token Filtering (Berchansky et al., 2023), Agile-Quant (Shen et al., 2024a), and EViT (Liang et al., 2022), typically involve sorting all tokens in the current layer based on an importance evaluation criterion, and then removing the less important tokens. As shown in Figure 1(a), after we directly implement post-training token pruning (EViT) to reduce 20% of the overall FLOPS for Mamba-2.8B, there is a dramatic drop in average accuracy on zero-shot evaluation. This performance drop is introduced by pruning certain tokens with *unrecoverable information loss*, although the pruned tokens are less important based on a heuristic importance metric. This *information loss* is *gradually amplified* during the sequence computations process of Equation (2) and (3) in SSMs.

Failure of token merging on SSMs. On the other hand, linguistic contexts often contain *redundant* tokens, which do not add significant contextual depth to the model's understanding. ToMe (Bolya et al., 2023) introduces a bipartite token merging strategy for vision Transformers. Following this, initiatives like PuMer (Cao et al., 2023) extend this strategy to vision-language models, merging redundant tokens in linguistic model components and their vision counterparts at the same time. However, as shown in Figure 1(b), applying this bipartite token merging strategy directly to SSMs proves ineffective. The strategy uniformly partitions the tokens in the current layer into two groups, and merges tokens in one group into the other group, disregarding the inherent value (or token importance) of each token. Thus, certain important tokens may be merged into other tokens. Given the critical role of important tokens in sequence computations using Equation (3) in SSMs, *overlooking* the *inherent significance* of tokens and thus removing important tokens can lead to substantially different $\mathbf{y}$ in Equation (3) and thus severe performance degradation.

3.3 Motivation

From the analysis presented, we conclude that the failure of token pruning in SSMs comes from the loss of crucial information due to token removal. Meanwhile, the failure of token merging in SSMs can be attributed to the neglect of token importance. This oversight can result in a more significant drop in accuracy compared to pruning, underscoring the critical role of token importance in the model's performance. Therefore, our objective is to combine token importance and similarity as guidance for a unified token reduction method (combining pruning and merging). We aim to develop a more fine-grained reduction strategy to handle the computation sensitivity of selective SSMs, ensuring that the reduction process maintains model accuracy and efficiency simultaneously.

4 Methodology

To tackle the problem, we first rethink the token importance metric for SSMs. We then introduce a novel approach for unified token reduction by token importance classification that combines the advantages of both token pruning and token merging to facilitate faster and memory-efficient computation across SSM layers.

4.1 Rethinking Token Importance Metric for State Space Models

To derive the appropriate token importance metric, we look at the layer computations in SSMs such as Mamba. For the l^{th} layer, the input token sequence $\mathbf{T}_{l-1} \in \mathbb{R}^{B\times N\times D}$ is first projected to $\mathbf{x} \in \mathbb{R}^{B\times N\times D'}$, and then goes through SSMs for data-dependent context modeling. It processes $\mathbf{x}$ from the `forward` scan via:

$$\mathbf{y} \leftarrow \texttt{SSM}(\mathbf{A}, \mathbf{B}, \mathbf{C})(\mathbf{x}), \tag{4}$$

where the hidden states $\mathbf{y} \in \mathbb{R}^{B\times N\times D'}$ is the output of SSM (see Equation (3)). The token sequence output of the l^{th} layer can be obtained as $\mathbf{T}_l \leftarrow \texttt{Linear}^T\mathbf{y} + \mathbf{T}_{l-1}$. To evaluate the importance of each token, we first extract the hidden states $\mathbf{y}$ from the SSM layer, denoted as $\mathbf{y} \in \mathbb{R}^{B\times N\times D'}$. The hidden states represent the intermediate representations of the tokens after passing through the SSM layer. To quantify the importance of each token, we compute the sum of the $\mathbf{y}$ across the last dimension, which corresponds to the feature dimension D'. The SSMs architecture, with its high-dimensional channel space, allows for a finer-granularity analysis of attention across numerous channels. Unlike Transformers that produce a single attention matrix per head, SSMs exploit their extensive channel capacity for a more detailed attention distribution, enhancing the model's ability to discern subtle features and interactions among tokens. Thus, we aggregate the clipped values across all channels for each token to evaluate token importance as follows,

$$\mathcal{S} = \frac{\sum_{d=1}^{D'} \texttt{max}(0, [\mathbf{y}]_{::d})}{D'}, \tag{5}$$

where $[\cdot]_{::d}$ denotes the d^{th} feature map in the feature dimension with size D'. We use $\mathcal{S} \in \mathbb{R}^{B\times N\times 1}$ as the token importance metric corresponding to $B \times N$ tokens to guide the reduction process, ensuring that only the most contextually relevant tokens are retained. To make a comprehensive study, we compare the performance with other token importance metrics, including the ℓ_1 norm, ℓ_2 norm, as well as unclipped values without the `max` operation. We find that using clipped values in Equation (5) as the token importance metric can constantly yield better results.

4.2 Unified Token Reduction by Token Importance Classification

To achieve token reduction, it is important to derive a token importance classification strategy that effectively differentiates between less important and more important tokens. However, it is challenging to directly classify thousands of tokens in real-time due to high complexity. To overcome this, we further leverage the token importance evaluation as in Equation (5), and employ a decoupling strategy. The strategy initially computes the importance of each token, followed by classification based on this obtained importance. After that, we perform unified token reduction (UTR) and leverage multiple design choices to enable effective and fine-grained strategies. Figure 2 illustrates our proposed approach. The steps of our method are as follows:

1. **Calculate** token importance with Equation (5).
2. **Classify** the tokens into set M_A and M_B based on their importance. At the end, $N/2$ less important tokens are assigned to set M_A, with the rest $N/2$ more important tokens to set M_B.
3. **Create** a single connection from each token in set M_A to its most similar counterpart in set M_B, as shown below,

$$f_i = \underset{b_j\in M_B}{\arg\max}\ \text{sim}(a_i, b_j), \tag{6}$$

$$g_i = \underset{b_j\in M_B}{\max}\ \text{sim}(a_i, b_j), \tag{7}$$

   where $\text{sim}(a, b)$ is the cosine similarity between token a and b, f_i denotes the most similar token in M_B to $a_i \in M_A$, and g_i is the corresponding largest similarity between a_i and f_i.
4. **Retain** the $p\%$ most similar connections after sorting $\{g_i, \forall i\}$.
5. **Process** the connected tokens with our UTR method.
6. **Reassemble** the two sets of tokens into one set.

Unified token merging and pruning. For the 5^{th} step of our method, we apply two token reduction strategies – merging and pruning. We can apply

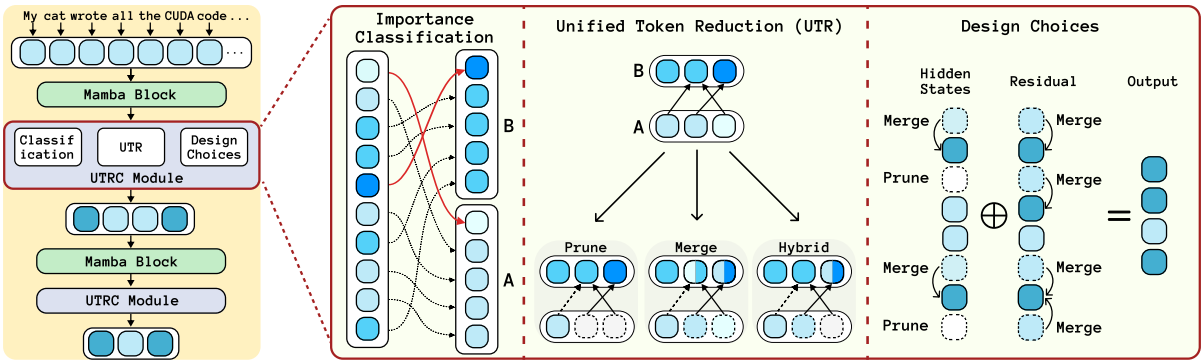

Figure 2: Overview of our proposed Unified Token Reduction by token importance Classification (UTRC) method. It contains three parts: Token Importance Classification, Unified Token Reduction (UTR), and Design Choices. Lighter colors indicate tokens with less importance, and darker colors indicate tokens with greater importance.

token pruning or merging for each of the connections obtained from the 4^{th} step. For token pruning, we do not change the tokens in Set M_B and only update Set M_A by removing the token a_i, i.e., $M_A = M_A \setminus a_i$, where $\setminus$ denotes the operation of element removal from the set. Consequently, f_i represents the remaining token in M_B for a connected pair (a_i, f_i). For merging, the tokens connected by retained pairs are combined by averaging their features. Specifically, we update the most similar token in M_B with $f_i = (a_i + f_i)/2$, and remove a_i from M_A. The modified f_i represents the fused token for the connected pair (a_i, f_i).

Our proposed merging and pruning techniques can be seamlessly integrated as shown in the UTR part in Figure 2. This allows for fine-grained reduction strategies across intra-layer branches, enabling distinct reduction strategies to both hidden states and residuals. The motivation is to address the *removed index misalignment* issue between branches. Such misalignment occurs when a token reduced in the hidden state is not concurrently reduced in the residual branch, and vice versa. This discrepancy, especially when branches recombine at the end of each layer, can significantly lower the overall compression ratio and hinder the effectiveness of fine-grained token reduction strategies. By unifying these techniques, we can optimize the method while meeting the required compression levels.

Hybrid token reduction. With the proposed UTR strategy, we further leverage a fine-grained strategy to balance the information importance and redundancy. For the corresponding tokens of retained $p\%$ most similar connections (the 4^{th} step), we prune $(p \times q)\%$ tokens and merge the remaining $[p \times (1-q)]\%$ tokens. We find that $q = 0.5$ leads to best performance compared with other q values. We provide a detailed evaluation in Table 5.

4.3 Design Choices

Intra-layer token reduction design. We delve deeper into our intra-layer token reduction design tailored for SSMs, targeting the hidden states and residual connections. Our approach employs the *hybrid token reduction* strategy on *hidden state* tokens, meticulously designed to strike a balance between preserving essential information and eliminating redundancy. By discerning the contextual significance of each token, this strategy focuses on removing tokens with minimal contextual relevance, thus enhancing the overall informational flow of the SSM module. This design choice not only preserves but also amplifies the high-contextual tokens. *Residual connections* are crucial for maintaining the integrity of information from the last layer. Therefore, we aim to preserve as much residual information as possible through our *token merging* method. The final design is shown in the design choices part in Figure 2. Empirical results support our fine-grained design, demonstrating that reducing tokens with our method in the hidden state and residual connection areas effectively preserves the performance of SSMs.

Hierarchical token reduction procedure. We apply a hierarchical method to reduce tokens across multiple layers. Tokens reduced in one layer are further reduced in subsequent layers, balancing overall efficiency and performance. Reducing tokens in each layer can cause high overhead, as token importance between adjacent layers is often similar. Thus, it is unnecessary to reduce tokens at every layer. Furthermore, reducing tokens in earlier layers yields greater computational savings, but these layers cannot fully capture token importance. In

Method	FLOPS Reduction	LAMBADA		HellaSwag	PIQA	Arc-E	Arc-C	WinoGrade	Avg.
		PPL ↓	Acc↑(%)	Acc↑(%)	Acc↑(%)	Acc↑(%)	Acc↑(%)	Acc↑(%)	Acc↑(%)
Mamba-2-1.3B	0%	5.02	65.7	59.9	73.2	64.3	33.3	60.9	59.5
+ PuMer	10%	532.52	33.3	27.5	61.3	57.8	30.6	59.8	45.1
+ EViT	10%	27.10	52.2	32.9	68.9	63.2	33.2	61.0	51.9
+ **Ours**	10%	11.16	55.9	59.2	71.0	64.3	34.1	61.0	57.6
+ PuMer	20%	49017.23	14.9	25.5	54.1	45.5	28.2	54.4	37.1
+ EViT	20%	1655.76	32.4	26.5	59.4	56.9	30.6	59.4	44.2
+ **Ours**	20%	25.94	46.1	58.0	64.3	64.0	34.4	60.7	54.6
Mamba-2-2.7B	0%	4.10	69.7	66.6	76.4	69.6	36.4	64.0	63.8
+ PuMer	10%	712.73	36.4	27.2	63.4	63.8	30.9	63.5	47.5
+ EViT	10%	11.43	55.8	35.7	72.0	69.1	35.4	64.1	55.4
+ **Ours**	10%	8.55	59.0	66.1	73.2	69.4	36.5	64.0	61.4
+ PuMer	20%	7820.51	20.7	25.9	56.0	50.5	28.8	56.0	39.7
+ EViT	20%	196.42	44.5	28.8	65.1	62.3	32.6	63.9	49.6
+ **Ours**	20%	17.96	49.1	64.7	68.2	69.4	37.5	63.1	58.7
+ PuMer	30%	49301.49	10.6	26.9	53.9	44.4	29.2	53.5	36.4
+ EViT	30%	3412.13	27.9	25.9	57.7	51.8	27.3	59.1	41.6
+ **Ours**	30%	42.61	38.3	59.4	61.2	68.4	37.3	63.9	54.7

Table 1: Main results of post-training performance on Mamba-2-1.3B and Mamba-2-2.7B. We compare with baseline methods and evaluate them on six benchmarks under 10%, 20%, and 30% FLOPS reduction.

our experiments, we apply token reduction after at least the 10^{th} layer and every 5 layers with a fixed compression ratio.

5 Experiment Results

5.1 Implementation Details

We implement our method based on PyTorch (Paszke et al., 2019) for scientific computations and HuggingFace (Wolf et al., 2019) for managing models. We use Mamba models to test the effectiveness of our method. Our approach covers a variety of Mamba models, with Mamba-2-2.7B, Mamba-2-1.3B, Mamba-2.8B and Mamba-1.4B. We evaluate the task performance on multiple common sense reasoning datasets including LAMBADA (Paperno et al., 2016), HellaSwag (Zellers et al., 2019), PIQA (Bisk et al., 2020), Arc-easy (Clark et al., 2018), Arc-challenge (Clark et al., 2018), and WinoGrade (Sakaguchi et al., 2021). Perplexity on LAMBADA dataset and average accuracy on all mentioned datasets are provided. All experiments are conducted on a NVIDIA A100 80GB GPU.

Reduction locations. We adopt the hierarchical token reduction procedure. For Mamba2-2.7B and Mamba-2.8B, we perform all methods in the [12, 17, 22, 27, 32, 37, 42] layers; for Mamba2-1.3B and Mamba-1.4B, we perform all methods in the [10, 15, 20, 25, 30, 35] layers. We use a fixed compression ratio for each prune layer.

Evaluation Details. The evaluation of perplexity (PPL) and average accuracy are adjusted to account for the reduction in the number of output due to token reduction. The target label logits are adjusted accordingly. For example, when the output token reduction rate is $m\%$, the label logits are also reduced to their first $1 - m\%$ logits to calculate the PPL and average accuracy properly.

Baselines. We compare our method with PuMer (Cao et al., 2023) and EViT (Liang et al., 2022). PuMer, which includes a dedicated text token reduction module, can be directly adopted in our study. For EViT, originally designed for vision Transformers, we configure it to ensure a fair comparison in our evaluation.

5.2 Quantitative Evaluation

Evaluation on Mamba-2. As shown in Table 1, for Mamba-2 models (1.3B and 2.7B), our method consistently achieves better performance than all baselines (PuMer and EViT) with non-marginal improvements under the same FLOPS reduction ratios. For Mamba-2-1.3B, our method achieves significantly lower PPL and higher accuracy on almost all downstream datasets, with an average accuracy 10% (54.6% v.s. 44.2% from EViT) higher than the best baseline under 20% FLOPS reduction. For Mamba-2-2.7B, our method outperforms baselines on various benchmarks with wide margins, achieving an average accuracy 13.1% higher than the best baseline under 30% FLOPS reduction.

Method	FLOPS Reduction	LAMBADA PPL ↓	LAMBADA Acc↑(%)	HellaSwag Acc↑(%)	PIQA Acc↑(%)	Arc-E Acc↑(%)	Arc-C Acc↑(%)	WinoGrade Acc↑(%)	Avg. Acc↑(%)
Mamba-1.4B	0%	5.04	64.9	59.1	74.2	65.5	32.8	61.5	59.7
+ PuMer	10%	534.91	34.6	25.8	59.7	55.6	29.5	59.5	44.1
+ EViT	10%	43.69	47.6	33.0	69.2	64.3	32.1	61.4	51.3
+ **Ours**	10%	11.46	56.5	58.9	71.3	65.1	33.9	61.4	57.8
+ PuMer	20%	11733.02	13.1	25.6	52.5	41.8	27.2	48.8	34.8
+ EViT	20%	5687.80	21.8	26.3	58.4	54.0	28.2	58.2	41.1
+ **Ours**	20%	31.32	44.9	57.7	62.8	62.8	33.2	59.0	53.4
Mamba-2.8B	0%	4.23	69.2	66.1	75.2	69.7	36.3	63.5	63.3
+ PuMer	10%	487.09	36.6	26.3	62.4	63.6	30.7	63.1	47.1
+ EViT	10%	174.92	51.8	35.7	71.0	68.9	35.7	63.2	54.4
+ **Ours**	10%	9.53	59.9	66.0	72.0	69.8	36.7	63.5	61.3
+ PuMer	20%	10746.15	17.9	25.3	52.5	47.0	28.7	52.0	37.2
+ EViT	20%	9784.73	26.9	24.8	59.9	57.2	29.9	63.1	43.6
+ **Ours**	20%	23.97	49.0	63.8	62.3	68.5	38.1	64.0	57.6
+ PuMer	30%	140763.76	6.0	26.0	54.6	41.5	26.6	51.7	34.4
+ EViT	30%	63230.76	12.3	25.0	52.5	41.9	23.6	51.9	34.5
+ **Ours**	30%	81.16	36.1	39.4	58.1	66.2	37.1	60.8	49.6

Table 2: Main results of post-training performance on Mamba-1.4B and Mamba-2.8B. We compare with baseline methods and evaluate them on six benchmarks under 10%, 20%, and 30% FLOPS reduction.

Evaluation on Mamba. As demonstrated in Table 2, for Mamba models (1.4B and 2.8B), we can make similar observations that our method outperforms all baselines with non-marginal improvements in terms of PPL and accuracy on multiple benchmarks. Our method maintains a low PPL while baselines can hardly keep a reasonable PPL (such as our 23.97 PPL v.s. 9785 from EViT under 20% FLOPS reduction for Mamba-2.8B). Our average accuracy is significantly higher than baselines, such as our 53.4% over 41.1% from EViT for Mamba-1.4B under 20% FLOPS reduction.

Summary. For SSMs such as Mamba, our proposed method consistently demonstrates better performance in terms of PPL and average accuracy across various levels of FLOPS reduction compared with baselines. PuMer and EViT fail to maintain high performance due to the reasons discussed in Section 3.2. After an insightful investigation of the reasons for failure and a comprehensive design to combine the advantages of pruning and merging, our unified method can effectively and efficiently prune tokens in SSMs without significant performance degradation.

5.3 Ablation Study & Analysis

Different Importance Metric. We study the token importance metric for our token reduction strategy. As shown in Table 3, for Mamba-2-2.7B and Mamba-2.8B, we provide a comparative analysis of different metrics: ℓ_1-norm, ℓ_2-norm, without `Clip` (the max function in Equation (5)), and with `Clip`, along with their impacts on LAMBADA PPL and average accuracy across six tasks (as in Table 2). The results show that `Clip` achieves the lowest PPL of 17.96 and the highest average accuracy of 58.7% for Mamba-2-2.7B, outperforming other metrics. For Mamba-2.8B, though `Clip` has a slightly higher PPL, its average accuracy is the highest 57.6%. This analysis underscores the importance of the proposed token importance metric in enhancing model accuracy and efficiency.

Model	Metric	LAMBADA PPL ↓	Avg. Acc. ↑(%)
Mamba-2-2.7B	ℓ_1-norm	17.96	58.6
	ℓ_2-norm	19.86	58.6
	w/o Clip	18.17	58.5
	Clip (ours)	**17.96**	**58.7**
Mamba-2.8B	ℓ_1-norm	**23.93**	56.8
	ℓ_2-norm	**23.93**	57.5
	w/o Clip	1365.69	40.7
	Clip (ours)	23.97	**57.6**

Table 3: Ablation study of token importance metric with our unified token merging and pruning design.

Reduction location analysis. The choice of token reduction location impacts model performance. Table 4 presents the ablation study of reduction location on Mamba-2-2.7B under a 20% FLOPS reduction. Notably, the configuration with reduction layers at [12, 17, 22, 27, 32, 37, 42] achieves the lowest PPL 17.96 and the highest 58.7% aver-

Location (every 5 layers)	LAMBADA PPL ↓	Avg. Acc. ↑(%)
[20, 25, 30, 35, 40, 45, 50]	18.88	57.8
[18, 23, 28, 33, 38, 43, 48]	18.32	58.3
[16, 21, 26, 31, 36, 41, 46]	18.79	58.1
[14, 19, 24, 29, 34, 39, 44]	18.74	58.3
[10, 15, 20, 25, 30, 35, 40]	18.76	58.2
[12, 17, 22, 27, 32, 37, 42]	**17.96**	**58.7**

Table 4: Ablation study of reduction location on Mamba-2-2.7B under 20% overall reduction of FLOPS.

age accuracy, demonstrating the effectiveness of this specific reduction strategy. In contrast, deeper reduction layers, such as [20, 25, 30, 35, 40, 45, 50], result in higher PPL and lower average accuracy, indicating that deeper layers do not always yield better results. Token reduction at earlier layers can lead to higher computation efficiency without sacrificing accuracy significantly.

Different design choices. For hidden states and residual connections, we can apply pruning, merging, or our hybrid token reduction with different combinations of pruning and merging (denoted by q). We conduct ablation studies to find the optimal q configuration for both hidden states and residual connections. Table 5 presents experiments on the Mamba-2-2.7B model under a 30% FLOPS reduction. The results indicate that the combination of $q = 0.5$ for hidden states and merging only for residual connections achieves the lowest 40.61 PPL and the highest 54.7% average accuracy, highlighting its effectiveness in this context. Furthermore, combining pruning and merging with $q = 0.5$ for hidden states consistently outperforms pruning-only or merging-only strategies. Notably, even our basic method using importance classification (M-only & M-only Acc. 54.0%) outperforms existing methods (PuMer Acc. 36.4% and EViT Acc. 41.6%) by a large margin.

5.4 Efficiency Results

We evaluate the GPU peak memory usage of Mamba-2.8B and Mamba-2-2.7B when generating 2048 tokens with a batch size 96 under various FLOPS reduction ratios. As illustrated in Figure 3, the GPU peak memory reduction for Mamba-2.8B can reach up-to 14.4%, 27.7%, and 40.0%, under 10%, 20%, and 30% FLOPS reduction, respectively. For Mamba-2-2.7B, it can reduce the peak memory by 11.4%, 20.3%, 30.6% when reducing 10%, 20%, and 30% FLOPS, respectively.

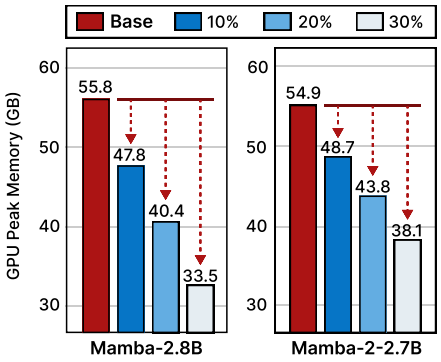

Figure 3: Comparison of GPU peak memory reduction between different FLOPS reduction ratios for Mamba-2.8B and Mamba-2-2.7B.

Hidden States	Residual Connections	LAMBADA PPL ↓	Avg. Acc. ↑(%)
M-only	M-only	42.61	54.0
P-only	P-only	42.65	53.9
$q = 0.8$	$q = 0.2$	42.65	54.3
$q = 0.2$	$q = 0.8$	42.67	54.1
$q = 0.5$	$q = 0.5$	42.35	53.7
$q = 0.5$	P-only	42.67	54.1
$q = 0.5$	M-only	**40.61**	**54.7**

Table 5: Ablation study of different design choices on Mamba-2-2.7B under 30% overall reduction of FLOPS.

Further, our proposed method can lead to practical inference acceleration with higher model throughput, as shown in Figure 4. The throughput can be improved by 1.07×, 1.17×, and 1.29× for Mamba-2.8B, and 1.10×, 1.22×, and 1.37× for Mamba-2-2.7B, when reducing 10%, 20%, and 30% FLOPS, respectively. The throughput measurements are collected with a batch size 16 by generating 100 tokens with a prompt length of 2048. More details and efficiency results of other models can be found in Appendix A.

6 Conclusion

In this paper, we introduced a unified post-training token reduction method for SSM architectures like Mamba. We addressed the limitations of existing token reduction techniques by combining token importance and similarity to create a fine-grained reduction strategy. Our method includes multiple design choices for effective intra-layer optimizations. Experiments show significant reductions in computational demands and peak memory usage, while maintaining competitive accuracy, outperforming baseline methods on benchmarks.

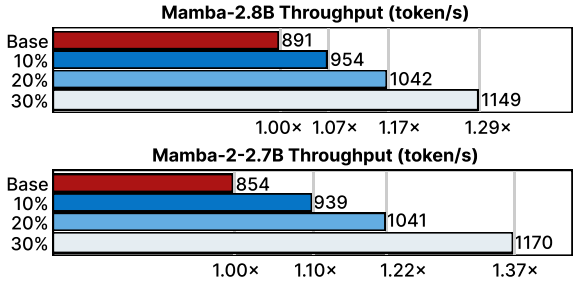

Figure 4: Comparison of the generation throughput between different FLOPS reduction ratios for Mamba-2.8B and Mamba-2-2.7B.

Limitations

Our experiments do not involve results after fine-tuning, which we believe could further improve the performance of our method. While our approach is applicable to Transformer-based LLMs, we have not tested it on other Transformer-based LLMs. We intend to address these extensions in future work.

Acknowledgement

This work is supported by National Science Foundation CNS-2312158. We would like to express our sincere gratitude to the reviewers for their invaluable feedback and constructive comments to improve the paper.

References

Moshe Berchansky, Peter Izsak, Avi Caciularu, Ido Dagan, and Moshe Wasserblat. 2023. Optimizing retrieval-augmented reader models via token elimination. *arXiv preprint arXiv:2310.13682*.

Yonatan Bisk, Rowan Zellers, Jianfeng Gao, Yejin Choi, et al. 2020. Piqa: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 7432–7439.

Daniel Bolya, Cheng-Yang Fu, Xiaoliang Dai, Peizhao Zhang, Christoph Feichtenhofer, and Judy Hoffman. 2023. Token Merging: Your ViT but Faster. In *International Conference on Learning Representations*.

Maxim Bonnaerens and Joni Dambre. 2023. Learned Thresholds Token Merging and Pruning for Vision Transformers. *Transactions on Machine Learning Research*.

Qingqing Cao, Bhargavi Paranjape, and Hannaneh Hajishirzi. 2023. Pumer: Pruning and merging tokens for efficient vision language models. *arXiv preprint arXiv:2305.17530*.

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 preprint arXiv:1803.05457*.

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

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. 2020. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*.

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

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

Albert Gu, Karan Goel, Ankit Gupta, and Christopher Ré. 2022. On the parameterization and initialization of diagonal state space models. *Advances in Neural Information Processing Systems*, 35:35971–35983.

Albert Gu, Karan Goel, and Christopher Ré. 2021a. Efficiently modeling long sequences with structured state spaces. *arXiv preprint arXiv:2111.00396*.

Albert Gu, Isys Johnson, Karan Goel, Khaled Saab, Tri Dao, Atri Rudra, and Christopher Ré. 2021b. Combining recurrent, convolutional, and continuous-time models with linear state space layers. *Advances in neural information processing systems*, 34:572–585.

Ankit Gupta, Albert Gu, and Jonathan Berant. 2022. Diagonal state spaces are as effective as structured state spaces. *Advances in Neural Information Processing Systems*, 35:22982–22994.

Xin Huang, Ashish Khetan, Rene Bidart, and Zohar Karnin. 2022. Pyramid-bert: Reducing complexity via successive core-set based token selection. *arXiv preprint arXiv:2203.14380*.

Rudolph Emil Kalman. 1960. A new approach to linear filtering and prediction problems.

Zhenglun Kong, Peiyan Dong, Xiaolong Ma, Xin Meng, Wei Niu, Mengshu Sun, Bin Ren, Minghai Qin, Hao Tang, and Yanzhi Wang. 2022. Spvit: Enabling faster vision transformers via soft token pruning. *ECCV*.

Zhenglun Kong, Haoyu Ma, Geng Yuan, Mengshu Sun, Yanyue Xie, Peiyan Dong, Xin Meng, Xuan Shen, Hao Tang, Minghai Qin, et al. 2023. Peeling the onion: Hierarchical reduction of data redundancy for efficient vision transformer training. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 8360–8368.

Youwei Liang, Chongjian GE, Zhan Tong, Yibing Song, Jue Wang, and Pengtao Xie. 2022. EVit: Expediting vision transformers via token reorganizations. In *International Conference on Learning Representations*.

Harsh Mehta, Ankit Gupta, Ashok Cutkosky, and Behnam Neyshabur. 2022. Long range language modeling via gated state spaces. *arXiv preprint arXiv:2206.13947*.

Ali Modarressi, Hosein Mohebbi, and Mohammad Taher Pilehvar. 2022. Adapler: Speeding up inference by adaptive length reduction. *arXiv preprint arXiv:2203.08991*.

Piotr Nawrot, Jan Chorowski, Adrian Łańcucki, and Edoardo M Ponti. 2022. Efficient transformers with dynamic token pooling. *arXiv preprint arXiv:2211.09761*.

Eric Nguyen, Karan Goel, Albert Gu, Gordon Downs, Preey Shah, Tri Dao, Stephen Baccus, and Christopher Ré. 2022. S4nd: Modeling images and videos as multidimensional signals with state spaces. *Advances in neural information processing systems*, 35:2846–2861.

Bowen Pan, Rameswar Panda, Yifan Jiang, Zhangyang Wang, Rogerio Feris, and Aude Oliva. 2021. Ia-red^2: Interpretability-aware redundancy reduction for vision transformers. *Advances in Neural Information Processing Systems*, 34:24898–24911.

Denis Paperno, Germán Kruszewski, Angeliki Lazaridou, Quan Ngoc Pham, Raffaella Bernardi, Sandro Pezzelle, Marco Baroni, Gemma Boleda, and Raquel Fernández. 2016. The lambada dataset: Word prediction requiring a broad discourse context. *arXiv preprint arXiv:1606.06031*.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. *PyTorch: an imperative style, high-performance deep learning library*. Curran Associates Inc., Red Hook, NY, USA.

Yongming Rao, Wenliang Zhao, Benlin Liu, Jiwen Lu, Jie Zhou, and Cho-Jui Hsieh. 2021. Dynamicvit: Efficient vision transformers with dynamic token sparsification. *Advances in neural information processing systems*, 34:13937–13949.

Cedric Renggli, André Susano Pinto, Neil Houlsby, Basil Mustafa, Joan Puigcerver, and Carlos Riquelme. 2022. Learning to merge tokens in vision transformers. *arXiv preprint arXiv:2202.12015*.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2021. Winogrande: An adversarial winograd schema challenge at scale. *Communications of the ACM*, 64(9):99–106.

Xuan Shen, Peiyan Dong, Lei Lu, Zhenglun Kong, Zhengang Li, Ming Lin, Chao Wu, and Yanzhi Wang. 2024a. Agile-quant: Activation-guided quantization for faster inference of llms on the edge. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 18944–18951.

Xuan Shen, Zhenglun Kong, Changdi Yang, Zhaoyang Han, Lei Lu, Peiyan Dong, Cheng Lyu, Chih hsiang Li, Xuehang Guo, Zhihao Shu, Wei Niu, Miriam Leeser, Pu Zhao, and Yanzhi Wang. 2024b. EdgeQAT: Entropy and Distribution Guided Quantization-Aware Training for the Acceleration of Lightweight LLMs on the Edge. *arXiv preprint arXiv:2402.10787*.

Xuan Shen, Pu Zhao, Yifan Gong, Zhenglun Kong, Zheng Zhan, Yushu Wu, Ming Lin, Chao Wu, Xue Lin, and Yanzhi Wang. 2024c. Search for Efficient Large Language Models. *arXiv preprint arXiv:2402.10787*.

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.

Hongwei Wang and Dong Yu. 2023. Going beyond sentence embeddings: A token-level matching algorithm for calculating semantic textual similarity. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 563–570.

Jue Wang, Wentao Zhu, Pichao Wang, Xiang Yu, Linda Liu, Mohamed Omar, and Raffay Hamid. 2023. Selective structured state-spaces for long-form video understanding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6387–6397.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. 2019. Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*.

Changdi Yang, Pu Zhao, Yanyu Li, Wei Niu, Jiexiong Guan, Hao Tang, Minghai Qin, Bin Ren, Xue Lin, and Yanzhi Wang. 2023. Pruning parameterization with bi-level optimization for efficient semantic segmentation on the edge. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15402–15412.

Li Yuan, Yunpeng Chen, Tao Wang, Weihao Yu, Yujun Shi, Zi-Hang Jiang, Francis EH Tay, Jiashi Feng, and Shuicheng Yan. 2021. Tokens-to-token vit: Training vision transformers from scratch on imagenet. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 558–567.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. Hellaswag: Can a

machine really finish your sentence? *arXiv preprint arXiv:1905.07830*.

Zheng Zhan, Yifan Gong, Pu Zhao, Geng Yuan, Wei Niu, Yushu Wu, Tianyun Zhang, Malith Jayaweera, David Kaeli, Bin Ren, Xue Lin, and Yanzhi Wang. 2021. Achieving On-Mobile Real-Time Super-Resolution With Neural Architecture and Pruning Search. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 4821–4831.

Zheng Zhan, Zhenglun Kong, Yifan Gong, Yushu Wu, Zichong Meng, Hangyu Zheng, Xuan Shen, Stratis Ioannidis, Wei Niu, Pu Zhao, and Yanzhi Wang. 2024. Exploring Token Pruning in Vision State Space Models. *Preprint*, arXiv:2409.18962.

Yihua Zhang, Yuguang Yao, Parikshit Ram, Pu Zhao, Tianlong Chen, Mingyi Hong, Yanzhi Wang, and Sijia Liu. 2022. Advancing model pruning via bi-level optimization. *Advances in Neural Information Processing Systems*, 35:18309–18326.

Lianghui Zhu, Bencheng Liao, Qian Zhang, Xinlong Wang, Wenyu Liu, and Xinggang Wang. 2024. Vision mamba: Efficient visual representation learning with bidirectional state space model. *arXiv preprint arXiv:2401.09417*.

A Appendix

A.1 More Details

Peak memory refers to the maximum memory required during a program's execution. If the peak memory exceeds the available VRAM on a GPU, it will result in an **"Out of Memory"** error, preventing the program from running.

A.2 More Efficiency Results

The GPU peak memory usage of Mamba-1.4B and Mamba-2-1.3B are shown in Figure 5 following the same configuration as Section 5.4. We follow the PyTorch instruction[2] to capture the GPU peak memory snapshot.

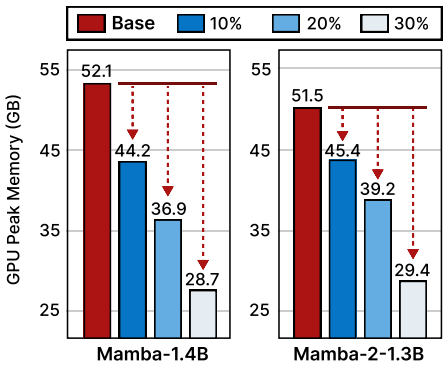

Figure 5: Comparison of GPU peak memory reduction between different FLOPS reduction ratios for Mamba-1.4B and Mamba-2-1.3B.

When reducing 10%, 20%, and 30% FLOPS compared to the baseline, Mamba-1.4B can obtain up to 15.2%, 29.1%, and 44.7% peak memory reduction, while the peak memory reduction for Mamba-2-1.3B can reach up-to 11.9%, 23.9%, and 42.9%.

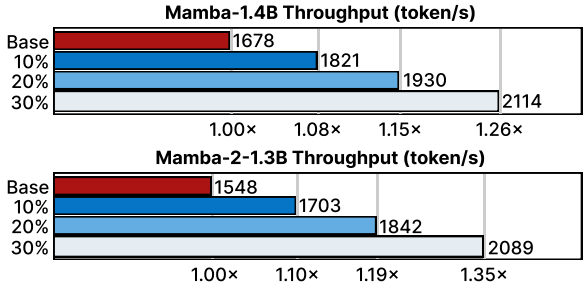

Figure 6: Comparison of the generation throughput between different FLOPS reduction ratios for Mamba-1.4B and Mamba-2-1.3B.

[2] https://pytorch.org/docs/stable/torch_cuda_memory.html

Method	FLOPS Reduction	LAMBADA		HellaSwag	PIQA	Arc-E	Arc-C	WinoGrade	Avg.
		PPL ↓	Acc↑(%)	Acc↑(%)	Acc↑(%)	Acc↑(%)	Acc↑(%)	Acc↑(%)	Acc↑(%)
Mamba-2-2.7B	0%	4.10	69.7	66.6	76.4	69.6	36.4	64.0	63.8
+ LTMP	10%	55.00	52.0	34.1	72.4	69.2	35.7	62.2	57.2
+ **Ours**		8.55	59.0	66.1	73.2	69.4	36.5	64.0	61.4
+ LTMP	20%	466.40	38.4	27.7	63.5	64.7	33.1	63.8	48.5
+ **Ours**		17.96	49.1	64.7	68.2	69.4	37.5	63.1	58.7
+ LTMP	30%	4670.71	22.3	24.9	58.9	54.0	28.3	59.2	41.3
+ **Ours**		42.61	38.3	59.4	61.2	68.4	37.3	63.9	54.7

Table 6: Additional results of post-training performance on Mamba-2-2.7B. We compare with LTMP and evaluate them on six benchmarks under 10%, 20%, and 30% FLOPS reduction.

The throughput of token generation for Mamba-1.4B and Mamba-2-1.3B using the proposed method are also collected under the same configuration in Section 5.4, as illustrated in Figure 6. With our optimization, the throughput can be improved by $1.08\times$, $1.15\times$, and $1.26\times$ for Mamba-1.4B, and $1.10\times$, $1.19\times$, and $1.35\times$ for Mamba-2-1.3B, when reducing 10%, 20%, and 30% FLOPS, respectively.

A.3 More Results

We compared our method with LTMP (Bonnaerens and Dambre, 2023), a simple token pruning and merging method designed for Vision Transformer. Our method outperforms LTMP in six benchmarks under same FLOPS reduction by a large margin, as shown in Table 6. The results emphasizing that the simple combination of token pruning and merging from Transformer is inadequate for SSMs.