Exploring Token Pruning in Vision State Space Models

Zheng Zhan[1*], **Zhenglun Kong**[12*], **Yifan Gong**[1], **Yushu Wu**[1], **Zichong Meng**[1]
Hangyu Zheng[3], **Xuan Shen**[1], **Stratis Ioannidis**[1], **Wei Niu**[3], **Pu Zhao**[1], **Yanzhi Wang**[1]
[1]Northeastern University, [2]Harvard University, [3]University of Georgia
{zhan.zhe, kong.zhe, yanz.wang}@northeastern.edu

Abstract

State Space Models (SSMs) have the advantage of keeping linear computational complexity compared to attention modules in transformers, and have been applied to vision tasks as a new type of powerful vision foundation model. Inspired by the observations that the final prediction in vision transformers (ViTs) is only based on a subset of most informative tokens, we take the novel step of enhancing the efficiency of SSM-based vision models through token-based pruning. However, direct applications of existing token pruning techniques designed for ViTs fail to deliver good performance, even with extensive fine-tuning. To address this issue, we revisit the unique computational characteristics of SSMs and discover that naive application disrupts the sequential token positions. This insight motivates us to design a novel and general token pruning method specifically for SSM-based vision models. We first introduce a pruning-aware hidden state alignment method to stabilize the neighborhood of remaining tokens for performance enhancement. Besides, based on our detailed analysis, we propose a token importance evaluation method adapted for SSM models, to guide the token pruning. With efficient implementation and practical acceleration methods, our method brings actual speedup. Extensive experiments demonstrate that our approach can achieve significant computation reduction with minimal impact on performance across different tasks. Notably, we achieve 81.7% accuracy on ImageNet with a 41.6% reduction in the FLOPs for pruned PlainMamba-L3. Furthermore, our work provides deeper insights into understanding the behavior of SSM-based vision models for future research[2].

1 Introduction

Recent years have witnessed the rapid evolvement of the computer vision field in the era of deep learning. Significant research efforts have been devoted to designing effective and efficient architectures of deep neural networks (DNNs) for visual tasks. Convolution Neural Networks (CNNs) [29, 12, 23, 30] and Vision Transformers (ViTs) [6, 22, 31, 43] are two representative categories of backbone networks. Though ViTs exhibit superior modeling capabilities with the incorporation of the self-attention mechanism [6, 32], the complexity of self-attention grows quadratically as the input size increases. Inspired by the great potential of State Space Models (SSMs) for long sequence modeling with linear complexity in natural language processing (NLP) tasks [9, 24, 33, 44], the latest backbone network designs for visual tasks [10, 21] leverage SSM-based blocks. Particularly, VMamba [21] reduces the complexity of attention computation with the selective scan mechanism presented in the S6 model [9] and matches the performance with existing foundation models.

Like the existing research efforts promoting the efficiency of CNNs and ViTs, the exploration of the SSM efficiency is desirable to facilitate real-time applications. While weight pruning is the prevalent

[*]Equal contributions
[2]Code available at https://github.com/ZLKong/ToP-ViM

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

technique for CNNs [34, 13, 17, 14, 38, 7, 8, 35, 40], token pruning [27, 26, 39, 28, 16, 41, 5] proves to be an effective way to enhance the efficiency of ViTs due to the independent patch processing design. Given that the SSM-based blocks also process input by dividing it into patches like ViTs, the existing token pruning techniques [18] for ViTs can be applied as a straightforward approach to boost the SSM efficiency. However, as shown in Figure 2, although enjoying certain benefits of faster inference with less tokens, this naive token pruning application for SSMs suffers from significant accuracy drops. Even after extensive fine-tuning efforts, its accuracy is still not able to recover from the token pruning with non-marginal gaps compared with the original accuracy. This indicates that the direct application of token pruning designed for ViTs permanently harms the performance of SSM-based vision models.

Given this observation, we conduct a thorough analysis of the computation patterns in SSM-based blocks, aiming to find the root cause and provide a foundation for efficient token pruning design in SSMs. Unlike ViTs whose attention mechanism computes the correlation between each pair of patches, SSM-based blocks follow traversal paths and thus the paths are sensitive to their adjacent patches. The direct application of token pruning techniques from ViT disrupts the patch locations/neighborhood in SSM-based blocks, thus incurring massive accuracy drops.

Based on our analysis, the question naturally arises whether we can keep the sequential property of tokens/batches in SSM-based vision models while pruning tokens to accelerate the forward computation. A successful solution not only improves the computational efficiency, but also provides more insights into the interpretability of SSM scan/token for future research. We take the first novel step towards this direction by proposing a general token pruning method for SSM-based vision models. Specifically, we propose a token importance evaluation method adapted for SSM models to guide the token pruning process based on a comprehensive analysis of SSM-based models. More importantly, to address the root cause of the above significant accuracy drop, we introduce a pruning-aware hidden state alignment method to reform the scan mechanism in SSMs for pruned and remaining tokens, thus stabilizing the neighborhood of remaining tokens and enhancing performance. Following the token pruning designs, we explore the efficient implementation and practical acceleration methods. With our tailored design, the computations can be significantly reduced with high accuracy performance. Notably, we achieve 81.7% accuracy on ImageNet for token pruned PlainMamba-L3, with 41.4% FLOPs reduction. We summarize our contributions as follows:

- After observing the incapability of directly applying token-based pruning techniques from ViTs for vision SSMs, we conduct a comprehensive analysis of SSM-based blocks to identify the failure reason, as well as provide more insights for the SSM scan mechanism in vision tasks, shedding lights on future research on SSM-based vision models.
- Based on our analysis, we propose a general token pruning method for SSM-based vision models, incorporating an adapted token importance evaluation to determine the pruned tokens, a pruning-aware hidden state alignment method to reform the SSM scan mechanism for pruned and remaining tokens, and practical implementation for efficient inference.
- We take the first step towards accelerating vision SSM models with token-based pruning. Our extensive and comprehensive experiments for image classification and object detection demonstrate the effectiveness of our proposed method for vision SSMs.

2 Related Work

State Space Models. SSMs [9, 24, 33] were first proposed to tackle long sequence modeling in the NLP community. The design has the strength to model complex systems by focusing on how the input, output, and state variables evolve over time. Recent progress has demonstrated that the variants of SSMs can be applied to visual tasks as an alternative to CNNs and ViTs with promising results. S4ND [25] is the first work that applies the state space mechanism to visual tasks and shows the potential to achieve competitive performance with ViTs [6]. The design expands the S4 model [10] and normalizes the parameters into a diagonal structure. But it fails to efficiently capture image information in an input-dependent manner. ViM [46] proposes a novel vision backbone with bidirectional Mamba. Based on that, PlainMamba [37] invents a continuous 2D scanning to enhance spatial continuity by ensuring adjacency of tokens in the scanning sequence. VMamba [21] introduce Cross-Scan Module (CSM) to enable 1D selective scan, matching the performance with existing foundation models including ResNet [12], ViT [6], Swin [22], and ConvNext [23]. The

great accomplishments demonstrate the potential of vision SSMs as an emerging fantastic foundation model family.

Token Pruning. Token pruning is an effective strategy to enhance computational efficiency by reducing the number of processed tokens or patches. It enables significant acceleration without requiring additional weights or specialized hardware, aiming to selectively retain the most informative tokens and sparsify the sequence. It is also vital for dense prediction tasks where sequence sizes are extensive. Several innovative approaches have been developed for vision transformers. For example, EViT [18] uses the attentiveness of the [CLS] token with respect to other tokens to identify the most important tokens. DynamicViT [27] and SPViT [15] add layers that employ the Gumbel-Softmax trick to selectively prune less informative tokens. IA-RED2 [26] drops redundant tokens with a multi-head interpreter. PS-ViT (T2T) [39] discard useless patches in a top-down paradigm. PATCHMERGER [28] uses spatial attention to generate a small set of tokens adaptive to the input. ToMe [2] measures dot product similarity between token keys to determine redundancy and prune accordingly. However, the dynamics of information flow between tokens and the learning mechanisms in models like Mamba [9] remain largely unexplored. Unlike ViTs that reply on attention features, the absence of attention layers in Mamba makes current pruning methods ineffective. Furthermore, the inclusion of the SSM module prevents the effective use of existing token pruning methods [42].

3 Preliminary and Motivation

3.1 State Space Models

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,

$$\begin{aligned} h'(t) &= \mathbf{A}h(t) + \mathbf{B}x(t), \\ y(t) &= \mathbf{C}h(t), \end{aligned} \tag{1}$$

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, $\mathbf{B} \in \mathbb{R}^{N\times L}$, and $\mathbf{C} \in \mathbb{R}^{L\times N}$ are the projection matrices.

The Mamba model [9] 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 follows,

$$\begin{aligned} \overline{\mathbf{A}} &= \exp(\Delta\mathbf{A}), \\ \overline{\mathbf{B}} &= (\Delta\mathbf{A})^{-1}(\exp(\Delta\mathbf{A}) - \mathbf{I}) \cdot \Delta\mathbf{B}. \end{aligned} \tag{2}$$

After obtaining the discretized $\overline{\mathbf{A}}$ and $\overline{\mathbf{B}}$, the discretization of Equation (1) can be rewritten as follows,

$$\begin{aligned} h_t &= \overline{\mathbf{A}}h_{t-1} + \overline{\mathbf{B}}x_t, \\ y_t &= \mathbf{C}h_t. \end{aligned} \tag{3}$$

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

$$\begin{aligned} \overline{\mathbf{K}} &= (\mathbf{C}\overline{\mathbf{B}}, \mathbf{C}\overline{\mathbf{A}\mathbf{B}}, \ldots, \mathbf{C}\overline{\mathbf{A}}^{\mathbf{L-1}}\overline{\mathbf{B}}), \\ \mathbf{y} &= \mathbf{x} * \overline{\mathbf{K}}, \end{aligned} \tag{4}$$

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 Failure of Applying ViT Token Pruning for ViMs

(Observation) After applying token pruning method to an SSM-based vision model, the **Zero-shot** performance will **drop significantly**. Moreover, this process will **permanently harm** the model's performance, even after **extensive fine-tuning**.

Epic failure of traditional token pruning for vision SSMs. To explore the token sparsity in vision SSMs, we first prune tokens in SSM-based models with the 'must-try' baseline, which directly applies the token pruning techniques designed for ViTs. Specifically, we prune the tokens using

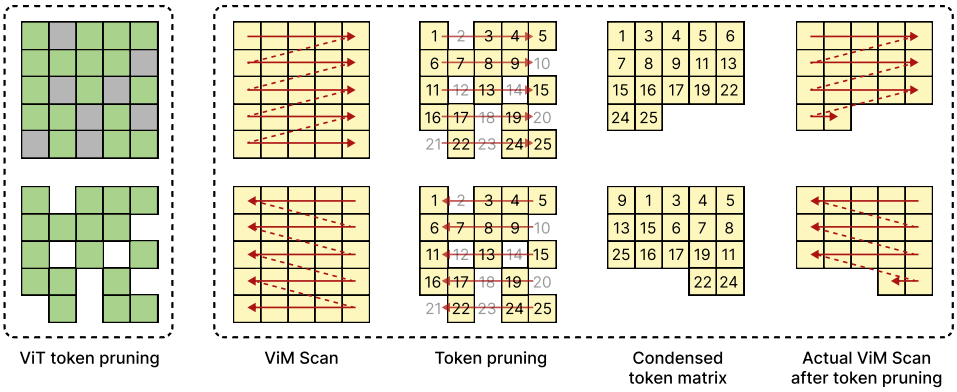

Figure 1: Illustration of the cross-scan in ViM models before and after token pruning.

selector metrics of EViT [18] on both transformer-based ViT-S [6] and SSM-based ViM-S models [46]. Given N input tokens for one layer, with token pruning, K tokens remain while the other $N - K$ tokens are pruned. The remaining tokens are relabeled as $\{x_j\}_{j=0}^{K-1}$ and their hidden states are obtained following Equation (3). In this way, the number of the tokens and the corresponding hidden states are reduced, condensing token matrix to save computation costs, as shown in Figure 1. After pruning tokens based on the token selector, we evaluate the performance in terms of zero-shot and fine-tuning accuracy. The results are shown in Figure 2. As observed, the direct application of token pruning from ViTs is not capable of delivering satisfying performance on ViMs. Specifically, direct token pruning suffers from substantial zero-shot accuracy degradation on ViM-S (with over 68% accuracy drop compared with the original accuracy), despite its success for ViT-S with merely 1.4% accuracy drop. Furthermore, even after extensive fine-tuning for the pruned model, its accuracy is not restored still with a 5.7% accuracy gap compared with the original ViM-S model, while it can boost the accuracy to a competitive level on ViT-S after fine-tuning. The significant performance degradation in ViM-S demonstrates that the direct application of token pruning hurts the underlying computations in SSM-based blocks, with permanent negative effects which can hardly be restored after fine-tuning.

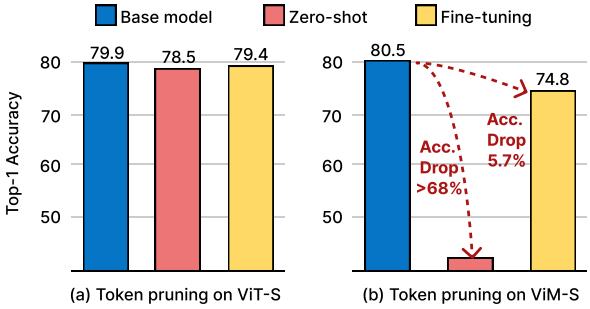

Figure 2: Accuracy comparison for token pruning on transformer-based ViT-S and SSM-based ViM-S.

Computation patterns in vision SSM. Observing the great success for ViT-S and epic failure for ViM-S with the same method, we are motivated to revisit the unique computation characteristics of SSMs and rethink the token pruning strategy in ViMs. To figure out the reason of failure, we look into the token computation patterns in SSM-based blocks. Given the input data, SSM-based blocks first unfold image patches/tokens into sequences along traversal paths (i.e., cross-scan, as shown in Figure 1 with ViM scan), process each token sequence using a separate computation block in parallel, and subsequently reshape and merge the resultant sequences to form the output map (i.e., cross-merge). The traversal paths facilitate the integration of information from all image pixels in various directions with linear complexity, enhancing the model's understanding.

Reason for the failure. However, the unique traversing along the sequence paths in ViM makes each token sensitive to its neighboring tokens. This is not a problem for ViT as the quadratic design of the attention mechanism calculates the correlation between the target token and all other tokens in the image, eliminating the sensitivity to adjacent tokens. As shown in Figure 1, introducing a token pruning strategy within an SSM-based block disrupts the original token positions in the SSM scan. Consequently, tokens that were not previously adjacent become neighbors during the scan in different directions or paths, leading to a distorted scan functionality and a significant accuracy degradation. Especially considering that the tokens are actually image patches in visual tasks with semantic information, disrupting their positions during the scan brings great difficulties to understand their relationship and the overall semantics.

In response to the limitations of directly applying existing token pruning methods designed for ViTs, we aim to address the following question:

(Question) *Can we prune tokens in SSM-based vision models to accelerate their forward computation without disrupting the original sequential token positions in different directions during the scan?*

4 Methodology

To address the above **Question**, we propose a general token pruning method tailored for SSM-based vision models. Specifically, we propose a pruning-aware hidden state alignment method to stabilize the neighborhood of remaining tokens during the scan, addressing the distorted scan functionality in traditional token pruning and thus enhancing accuracy performance. Furthermore, based on our detailed analysis of SSM-based vision models, we propose a token importance evaluation method adapted for SSM models, to guide the token pruning. Moreover, we discuss the efficient implementation and practical acceleration methods for token-pruned SSM-based vision models.

4.1 Pruning-Aware Hidden State Alignment

To maintain the sequential property of SSM tokens during the scan and tackle the **Question**, we propose the following novel pruning-aware hidden state alignment technique to align the sequential positions or neighbourhood of tokens before and after token pruning during the scan, thus maintaining the model performance under token pruning. For SSM-based vision models, the input token sequence for the l^{th} layer is denoted as $T_{l-1} \in \mathbb{R}^{B\times N\times D}$, where B, N, and D are the batch size, token number, and hidden state dimension, respectively. The tokens in one batch of the sequence can be unfolded as $\{x_j\}_{j=0}^{N-1}$ with N tokens in total. After applying token pruning (Section 4.2), K tokens are kept while the other $N-K$ tokens are removed from the input token sequence. We adopt different strategies to align the hidden states of remained tokens and pruned tokens during the scan as detailed below.

Alignment of hidden states for remaining tokens. We denote the set of the remaining token indices as $\{q_j\}_{j=0}^{K-1}$ with K elements and $q_s < q_t$ if $s < t$. Formally, the pruning-aware hidden states during the scan corresponding to the remained tokens can be represented as

$$\begin{aligned} h'_{q_0} &= \overline{\mathbf{B}}x_{q_0}, \\ h'_{q_1} &= \overline{\mathbf{A}}^{q_1-q_0}\overline{\mathbf{B}}x_{q_0} + \overline{\mathbf{B}}x_{q_1}, \\ &... \\ h'_{q_{(K-1)}} &= \underbrace{\overline{\mathbf{A}}^{q_{(K-1)}-q_0}\overline{\mathbf{B}}x_{q_0} + \overline{\mathbf{A}}^{q_{(K-1)}-q_1}\overline{\mathbf{B}}x_{q_1} + \overline{\mathbf{A}}^{q_{(K-1)}-q_2}\overline{\mathbf{B}}x_{q_2} + ... + \overline{\mathbf{B}}x_{q_{(K-1)}}}_{K \text{ terms/tokens}}. \end{aligned} \tag{5}$$

As shown in Equation (5), the hidden states of remained tokens depend on its current token and all previous remaining tokens. The pruned tokens are not effective in the hidden states.

Alignment of hidden states for pruned tokens. As observed in Figure 1 and 2, if one token is pruned, removing its position during the scan disrupts the neighbourhood of its adjacent tokens, leading to significant zero-shot accuracy drop which can hardly be compensated even after extensive fine-tuning. To mitigate this problem, our pruning-aware hidden state alignment maintains the position gap from pruned tokens during the scan to stabilize the neighbourhood of all remaining tokens. Specifically, to make the problem tractable, for two adjacent remaining tokens x_{q_i} and $x_{q_{i+1}}$, if $q_{i+1} - q_i > 1$, meaning there are tokens pruned between x_{q_i} and $x_{q_{i+1}}$, we denote the number of pruned tokens between x_{q_i} and $x_{q_{i+1}}$ as K_i ($K_i \geq 1$) and their indices can be represented as $\{q_i + j\}_{j=1}^{K_i}$. We have $q_i < (q_i) + 1 < ... < (q_i) + K_i < q_{(i+1)}$. To highlight the difference between q_{i+1} and $q_i + 1$, we use round brackets in the expression (e.g., $q_{(i+1)}$ and $(q_i) + 1$) without changing their meanings. Thus, the hidden states for the pruned tokens between two remaining adjacent tokens can be represented as follows,

$$
\begin{aligned}
h'_{q_i} &= \overline{\mathbf{A}}^{q_i-q_0}\overline{\mathbf{B}}x_{q_0} + \overline{\mathbf{A}}^{q_i-q_1}\overline{\mathbf{B}}x_{q_1} + ... + \overline{\mathbf{B}}x_{q_i}, \\
h'_{(q_i)+1} &= \overline{\mathbf{A}}^{(q_i)+1-(q_0)}\overline{\mathbf{B}}x_{q_0} + \overline{\mathbf{A}}^{(q_i)+1-(q_1)}\overline{\mathbf{B}}x_{q_1} + ... + \overline{\mathbf{A}\mathbf{B}}x_{q_i}, \\
&... \\
h'_{(q_i)+K_i} &= \overline{\mathbf{A}}^{(q_i)+K_i-(q_0)}\overline{\mathbf{B}}x_{q_0} + \overline{\mathbf{A}}^{(q_i)+K_i-(q_1)}\overline{\mathbf{B}}x_{q_1} + ... + \overline{\mathbf{A}}^{K_i}\overline{\mathbf{B}}x_{q_i}, \\
h'_{q_{(i+1)}} &= \overline{\mathbf{A}}^{q_{(i+1)}-q_0}\overline{\mathbf{B}}x_{q_0} + \overline{\mathbf{A}}^{q_{(i+1)}-q_1}\overline{\mathbf{B}}x_{q_1} + ... + \overline{\mathbf{A}}^{q_{(i+1)}-q_i}\overline{\mathbf{B}}x_{q_i} + \overline{\mathbf{B}}x_{q_{(i+1)}}.
\end{aligned} \tag{6}
$$

For pruned tokens with indices smaller than q_0, their hidden states are set to zero. For pruned tokens with indices larger than q_{K-1}, their hidden states can still be obtained following Equation (6). As shown in Equation (6), if a token is pruned, we do not simply remove its corresponding hidden state during the scan as it leads to substantial accuracy degradation shown in Figure 1 and 2. Instead, its hidden state in the scan can be obtained by using the previous state with one step forward, i.e., $h'_{(q_i)+1} = \overline{\mathbf{A}}h'_{(q_i)} + \overline{\mathbf{B}}x_{(q_i)+1} = \overline{\mathbf{A}}h'_{(q_i)}$ where the token $x_{(q_i)+1}$ is pruned. In this way, the hidden states corresponding to pruned tokens are aligned with that of the original unpruned tokens to maintain the sequential positions of the original tokens without disrupting their neighbours.

Comparison with traditional token pruning. As discussed in Section 3.2, in traditional ViT token pruning, the remaining tokens are relabeled as $\{x_j\}_{j=0}^{K-1}$ (disrupting their neighbours due to removal of pruned indices) and their hidden states are obtained following Equation (3). Different from ViT token pruning, we still keep the original indices of all tokens (including remaining and pruned tokens) to record their original sequential positions and neighbourhood. During the scan, the hidden states of pruning tokens becomes completely zero in ViT token pruning, which is different from our adapted scan mechanism in Equation (6) to keep a copy from its previous unpruned neighbour.

4.2 Token Pruning based on Importance Evaluation

In SSM-based vision models such as ViM, for the l^{th} layer, the input token sequence $T_{l-1} \in \mathbb{R}^{B\times N\times D}$ is first projected to $X' \in \mathbb{R}^{B\times N\times D'}$, and then goes through bidirectional SSMs for data-dependent global visual context modeling. It processes X' from the `forward` and `backward` scan via:

$$
y_m \leftarrow \texttt{SSM}(A_m, B_m, C_m)(X'_m), \quad \text{for } m \in \{\texttt{forward}, \texttt{backward}\}, \tag{7}
$$

where $y_m \in \mathbb{R}^{B\times N\times D'}$ is the output of SSM. Then y_m is gated to obtain $y'_{\texttt{forward}}$ and $y'_{\texttt{backward}}$. The token sequence output of the l^{th} layer can be obtained as follows:

$$
T_l \leftarrow \texttt{Linear}^T(y'_{\texttt{forward}} + y'_{\texttt{backward}}) + T_{l-1}. \tag{8}
$$

Therefore, the output of SSM can directly reflect the token importance. The Mamba 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, Mamba models 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,

$$
\mathcal{S} = \frac{\sum_{d=1}^{D'} \texttt{max}(0, [\mathbf{y}]_{::d})}{D'}, \tag{9}
$$

where $[\cdot]_{::d}$ denotes the d^{th} feature map in the feature dimension with size D'. We use $\mathcal{S}$ as the token importance metric to guide the pruning process, ensuring that only the most contextually relevant tokens are retained, thereby optimizing computational resources. Given the sparsity requirement for the token pruning, we sort $\mathcal{S}$ and prune the corresponding less important tokens. 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 (9) as the token importance metric can constantly yield better results.

4.3 Efficient Implementation and Practical Acceleration

Efficient implementation for the SSM scan. Based on the pruning-aware hidden state alignment technique discussed in Section 4.1, we propose the `pruning-aware hidden state alignment` kernel for practical acceleration. It utilizes a position map to guide the `SSM` operator, ensuring the correctness of computations. The position map is the token pruning indicator based on $\{q_j\}_{j=0}^{K-1}$ in Section 4.1, which inherits from token importance evaluation and records the location of remained tokens and pruned tokens. The `pruning-aware hidden state alignment` kernel takes the pruned dense sequences and the position map as its inputs. During the scan, it switches between the token remaining pattern and the token pruned pattern based on the remaining/pruning state of the token indexed by the position map. The token remaining pattern in the kernel follows the computations in Equation (5). Similarly, following Equation (6), the token pruned pattern still updates the hidden states but ignores computations related to the current token. The kernel switches to another pattern if it detects a corresponding change in token pruning state. A pseudo-code for our `pruning-aware hidden state alignment` is demonstrated in Appendix A. Thus, the `pruning-aware hidden state alignment` kernel can effectively accelerate the SSM scan under token pruning.

Practical acceleration for the whole model. Note that the SSM scan only takes up around $10\sim20\%$ computations in the whole model. With less tokens, other parts in the model can be accelerated directly due to less computations from pruned tokens, leading to significant inference speedup performance as demonstrated in our experiments.

5 Experiment Results

We conduct comprehensive experiments on ImageNet-1K[4], COCO 2017 [20] and ADE20K [45] datasets. All experiments are conducted on 4 NVIDIA V100s. "-EViT" means apply EViT token pruning method. "-prune" means apply our token pruning method. We report average results of multiple runs for all experimental sections, and different runs do not vary much. For ViM-T, we prune after the 10th and 20th layers. For ViM-S, we prune after the 5th, 10th, 15th, and 20th layers. For PlainMamba-L1, we prune after the 5th and 10th layers. For PlainMamba-L2, we prune after the 5th, 10th and 15th layers. For PlainMamba-L3, we prune after the 5th, 11th, 17th, and 23th layers.

5.1 Image Classification on ImageNet-1K

Settings. We finetune both the ViM and PlainMamba for 30 epochs on the ImageNet-1K dataset. The top-1 accuracy on the validation set is reported. For ViM, we set a patch extraction stride of 8 while keeping the patch size unchanged, a constant learning rate of 10^{-5}, and a weight decay of 10^{-8}. For PlainMamba, we use a warm-up period of 5 epochs. The weight decay is set to 1e-8, the base learning rate to 2e-5, the warm-up learning rate to 2e-8, and the minimum learning rate to 2e-7.

Results. The comparison results of our token pruning models against benchmark backbone models on ImageNet-1K are summarized in Table 1. One advantage of our method is that it is general and can be applied to a wide range of SSM-based vision model architectures to reduce computation complexity with a minor loss of performance. We evaluate our method on five base models including ViM-T, ViM-S, PlainMamba-L1, PlainMamba-L2, and PlainMamba-L3. We report the top-1 accuracy and FLOPs. Compared to directly applying the EViT method on vision state space models, using our pruning-aware hidden state alignment and token importance metric constantly outperforms EViT across various models of different scales. Specifically, On ViM, our method surpasses EViT by 3.8% on ViM-T and 4.0% on ViM-S. On PlainMamba, our method exceed EViT by 2.4% on PlainMamba-L1, 2.7% on PlainMamba-L2, and 2.8% on PlainMamba-L3.

5.2 Object Detection and Instance Segmentation

Settings. Following previous works, we conduct experiments for object detection and instance segmentation on the COCO 2017 dataset. The COCO 2017 dataset contains 118K images for training, 5K images for validating, and 20K images for testing. We use both the two-stage Mask R-CNN [11] and the single-stage RetinaNet [19]. For both models, we report the results of both $1\times$ schedule. Following [37], we use ViTAdapter [3] to compute multi-scale features to fit the FPN network structure.

Table 1: Classification results of different models on ImageNet-1K. We compare the proposed token pruning method with existing methods under comparable GFLOPs.

Method	Img. Size	Params (M)	FLOPs(G)	Top-1 Acc. (%)
ViT-Base	384^2	86	55.40	77.9
ViT-Large	384^2	307	190.70	76.5
DeiT-Tiny	224^2	6	1.30	72.2
DeiT-Small	224^2	22	4.60	79.8
DeiT-Base	224^2	86	17.50	81.8
ViM-T	224^2	7	1.50	76.1
ViM-S	224^2	26	5.10	80.5
ViM-T-EViT	224^2	7	1.28 (-14.3%)	71.3
ViM-S-EViT	224^2	26	3.57 (-30.0%)	74.8
ViM-T-ToP	224^2	7	1.29 (-14.0%)	75.1
ViM-S-ToP	224^2	26	3.60 (-29.4%)	78.8
PlainMamba-L1	224^2	7	3.0	77.9
PlainMamba-L2	224^2	25	8.1	81.6
PlainMamba-L3	224^2	50	14.4	82.3
PlainMamba-L1-EViT	224^2	7	2.44 (-18.7%)	75.0
PlainMamba-L2-EViT	224^2	25	6.22 (-23.2%)	78.3
PlainMamba-L3-EViT	224^2	50	8.35 (-42.0%)	78.9
PlainMamba-L1-ToP	224^2	7	2.46 (-18.0%)	77.4
PlainMamba-L2-ToP	224^2	25	6.27 (-22.6%)	81.0
PlainMamba-L3-ToP	224^2	50	8.44 (-41.4%)	81.7

Table 2: Results on COCO object detection and instance segmentation.

Backbone	AP^b	AP^b_{50}	AP^b_{75}	AP^m	AP^m_{50}	AP^m_{75}
PVT-Small	40.4	62.9	43.8	37.8	60.1	40.3
PVT-Medium	42.0	64.4	45.6	39.0	61.6	42.1
PVT-Large	42.9	65.0	46.6	39.5	61.9	42.5
Swin-Tiny	42.7	65.2	46.8	39.3	62.2	42.2
Swin-Small	44.8	66.6	48.9	40.9	63.2	44.2
PlainMamba-L1	44.1	64.8	47.9	39.1	61.6	41.9
PlainMamba-L2	46.0	66.9	50.1	40.6	63.8	43.6
PlainMamba-L3	46.8	68.0	51.1	41.2	64.7	43.9
PlainMamba-L1-EViT	41.9	62.8	45.7	37.2	60.1	40.2
PlainMamba-L2-EViT	43.7	64.2	47.6	38.3	62.2	41.9
PlainMamba-L3-EViT	44.2	66.4	49.7	39.5	62.8	42.7
PlainMamba-L1-ToP	43.7	64.6	47.4	38.9	61.3	41.5
PlainMamba-L2-ToP	45.5	66.2	49.9	40.1	63.3	42.7
PlainMamba-L3-ToP	46.5	67.7	50.8	40.6	64.1	43.4

Results. We used our pruned PlainMamba models as the backbone and compared them with existing token pruning methods and dense backbones. As shown in Table 2, our token pruning method maintains similar performance to dense models (less than 0.5%). When compared to existing token pruning methods, specifically for PlainMamba-L1, our pruning method outperforms EViT-based pruning by an average of 1.59% across all six precision metrics. For PlainMamba-L2, our method surpasses EViT by an average of 1.63% across all six precision metrics. Additionally, for PlainMamba-L3, our method exceeds EViT by an average of 1.30% across all six precision metrics.

5.3 Semantic Segmentation on ADE20K

Settings. We conduct experiments for semantic segmentation on the ADE20K dataset [45]. ADE20K contains 150 fine-grained semantic categories, with 20K, 2K, and 3K images for training, validation,

and testing, respectively. We choose UperNet [36] as our base framework. We train all models for 160 iterations with batch size 16 and set the default training image size to 512×512.

Results. We show the results of semantic segmentation on ADE20K in Table 3. The results indicate that our method also works well for the semantic segmentation task by greatly reducing the computation costs while maintaining satisfying performance. For instance, our token pruned PlainMamba-L1 reaches a mIoU of 44.1%, which is the same as the unpruned PlainMamba-L1. Our PlainMamba-L3-prune has a mIoU of 48.6%, which is better than current state-of-the-art model architectures including LocalVim-S and VMamba-T.

Table 3: Semantic Segmentation.

Method	mIoU/%
ViM-T	41.0
ViM-S	44.9
LocalVim-T	43.4
LocalVim-S	46.4
PlainMamba-L1	44.1
PlainMamba-L2	46.8
PlainMamba-L3	49.1
PlainMamba-L1-EViT	42.2
PlainMamba-L2-EViT	44.1
PlainMamba-L3-EViT	46.3
PlainMamba-L1-ToP	44.1
PlainMamba-L2-ToP	46.5
PlainMamba-L3-ToP	48.6

5.4 Ablation & Analysis

5.4.1 Token Importance Metric Analysis

In Table 4, we study on the impact of different token importance metrics, focusing on pruning-aware hidden state alignment. We test on two models: ViM-S and PlainMamba-L3. For the ViM-S model, both ℓ_1-norm and ℓ_2-norm methods achieve an accuracy of 78.6%, while the method without clipping (w/o Clip) results in a lower accuracy of 77.4%. The proposed clipping method (Clip) achieves the highest accuracy of 78.8%. For the L3 model, similar trends are observed: the ℓ_1-norm and ℓ_2-norm methods yield accuracies of 81.6% and 81.5%, respectively. The non-clipping approach results in a decrease in accuracy to 80.5%, whereas the clipping method provides a better enhancement, achieving 81.7%. These results suggest that the clipping mechanism in token importance metrics offers a consistent improvement in model accuracy, particularly in the context of pruning-aware hidden state alignment. It can potentially mitigate the adverse effects of extreme token importance values.

5.4.2 Quantitative Evaluation of pruning-aware hidden state alignment.

Table 5: Comparison of w/o and w/ our alignment (both using Eq. (9) as token importance metric).

Model	Method	FLOPs	Top-1 Acc. (%)	Throughput
ViM-S	Dense	5.10G	80.5	1×
	Prune w/o our alignment	3.57G	75.4	1.30×
	Prune w/ our alignment	**3.60G**	**78.8**	1.27×
PlainMamba-L3	Dense	14.40G	82.3	1×
	Prune w/o our alignment	8.35G	79.3	1.47×
	Prune w/ our alignment	**8.44G**	**81.7**	1.43×

In Table 5, we compare the performance of different pruning methods across two models: ViM-S and PlainMamba-L3. Our pruning method without the alignment process reduces the FLOPs to 3.57G but also lowers the accuracy to 75.4%, resulting in an improved throughput of 1.30×. In contrast, adding the align matrix achieves a much higher accuracy of 78.8%, with a similar throughput of 1.27×. For the PlainMamba-L3 model, our pruning method without the alignment reduces FLOPs to 8.35G but decreases accuracy to 79.3%, while increasing throughput to 1.47×. Equipping the alignment process improves accuracy to 81.7% and achieves a throughput of 1.43×. These results demonstrate that the proposed pruning method with the alignment process can effectively balance computational efficiency and model accuracy, outperforming the baseline pruning approach.

Table 4: Ablation study of token importance metric with pruning-aware hidden state alignment .

Model	Method	Accuracy (%)
ViM-S	ℓ_1-norm	78.6
	ℓ_2-norm	78.6
	w/o Clip	77.4
	Clip (ours)	78.8
L3	ℓ_1-norm	81.6
	ℓ_2-norm	81.5
	w/o Clip	80.5
	Clip (ours)	81.7

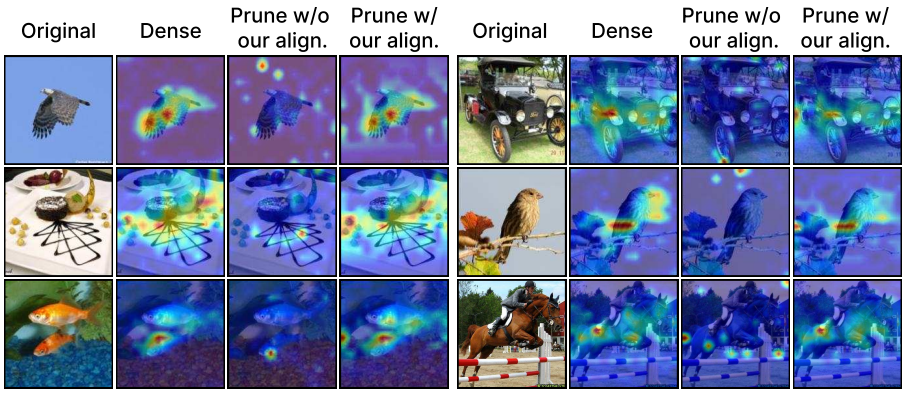

Figure 3: Visual representation on ImageNet-1K. We present the original images, attention visualizations from ViM-S, and **zero-shot results** of w/o and w/ our alignment method after the final layer.

5.5 Visualization and Interpretability

To further interpret token pruning in SSM-based vision models and understand the pruning-aware hidden state alignment behavior of our approach, we present attention visualizations based on zero-shot results in Figure 3. Our pruning-aware hidden state alignment effectively aligns the hidden states of pruned tokens in the SSM scan, maintaining similar visual representations and attention regions as the dense model. In contrast, pruning without our alignment shows significantly different attention regions, which could explain the huge accuracy drop. This demonstrates the effectiveness of our proposed pruning-aware hidden state alignment. The visualization tool is adopted from [1].

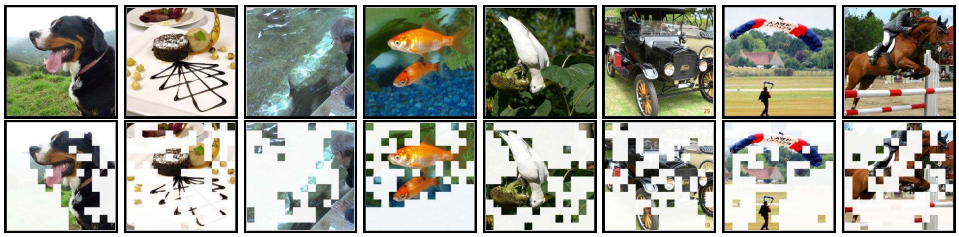

Figure 4: visualizations of locations of pruned token. We use the output after the final layer to visualize this reduction results.

We further visualize the token reduction results of our method within Fig. 4. We show the input images along with their sparsification results. The masked regions represent the tokens that have been pruned. Our method can gradually drop less informative tokens during forward pass and preserve the tokens that contain representative regions with an adaptive pruned region for each image.

6 Conclusion and Limitation

In this paper, we take the first step toward accelerating vision SSM models with token-based pruning. We analyze SSM-based blocks to understand the failure of direct token pruning and propose a general token pruning method for SSM-based vision models. This method includes an adapted token importance evaluation, a pruning-aware hidden state alignment, and practical implementations for efficient inference. Our extensive experiments confirm the effectiveness of our method and provide deeper insights into the SSM scan mechanism, guiding future research on SSM-based vision models. Though our method is general, the efficiency is limited by baseline model architecture design.

Acknowledgement

This work is supported by National Science Foundation CNS-2312158, and also CCF-2428108, OAC-2403090. We would like to express our sincere gratitude to the reviewers for their invaluable feedback and constructive comments to improve the paper.

References

[1] Ameen Ali, Itamar Zimerman, and Lior Wolf. The hidden attention of mamba models, 2024.

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

[3] Zhe Chen, Yuchen Duan, Wenhai Wang, Junjun He, Tong Lu, Jifeng Dai, and Yu Qiao. Vision transformer adapter for dense predictions. *arXiv preprint arXiv:2205.08534*, 2022.

[4] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009.

[5] Peiyan Dong, Mengshu Sun, Alec Lu, Yanyue Xie, Kenneth Liu, Zhenglun Kong, Xin Meng, Zhengang Li, Xue Lin, Zhenman Fang, et al. Heatvit: Hardware-efficient adaptive token pruning for vision transformers. In *2023 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, pages 442–455. IEEE, 2023.

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

[7] Yifan Gong, Geng Yuan, Zheng Zhan, Wei Niu, Zhengang Li, Pu Zhao, Yuxuan Cai, Sijia Liu, Bin Ren, Xue Lin, et al. Automatic mapping of the best-suited dnn pruning schemes for real-time mobile acceleration. *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, 27(5):1–26, 2022.

[8] Yifan Gong, Zheng Zhan, Pu Zhao, Yushu Wu, Chao Wu, Caiwen Ding, Weiwen Jiang, Minghai Qin, and Yanzhi Wang. All-in-one: A highly representative dnn pruning framework for edge devices with dynamic power management. In *Proceedings of the 41st IEEE/ACM International Conference on Computer-Aided Design*, pages 1–9, 2022.

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

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

[11] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017.

[12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[13] Yang He, Guoliang Kang, Xuanyi Dong, Yanwei Fu, and Yi Yang. Soft filter pruning for accelerating deep convolutional neural networks. In *International Joint Conference on Artificial Intelligence (IJCAI)*, 2018.

[14] Yang He, Ping Liu, Ziwei Wang, Zhilan Hu, and Yi Yang. Filter pruning via geometric median for deep convolutional neural networks acceleration. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

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

[16] Zhenglun Kong, Haoyu Ma, Geng Yuan, Mengshu Sun, Yanyue Xie, Peiyan Dong, Xin Meng, Xuan Shen, Hao Tang, Minghai Qin, et al. 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, 2023.

[17] Tuanhui Li, Baoyuan Wu, Yujiu Yang, Yanbo Fan, Yong Zhang, and Wei Liu. Compressing convolutional neural networks via factorized convolutional filters. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

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

[19] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988, 2017.

[20] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*, pages 740–755. Springer, 2014.

[21] Yue Liu, Yunjie Tian, Yuzhong Zhao, Hongtian Yu, Lingxi Xie, Yaowei Wang, Qixiang Ye, and Yunfan Liu. Vmamba: Visual state space model. *arXiv preprint arXiv:2401.10166*, 2024.

[22] Ze Liu, Han Hu, Yutong Lin, Zhuliang Yao, Zhenda Xie, Yixuan Wei, Jia Ning, Yue Cao, Zheng Zhang, Li Dong, et al. Swin transformer v2: Scaling up capacity and resolution. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 12009–12019, 2022.

[23] Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A convnet for the 2020s. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11976–11986, 2022.

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

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

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

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

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

[29] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[30] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International conference on machine learning*, pages 6105–6114. PMLR, 2019.

[31] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. In *International conference on machine learning*, pages 10347–10357. PMLR, 2021.

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

[33] Jue Wang, Wentao Zhu, Pichao Wang, Xiang Yu, Linda Liu, Mohamed Omar, and Raffay Hamid. 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, 2023.

[34] Wei Wen, Chunpeng Wu, Yandan Wang, Yiran Chen, and Hai Li. Learning structured sparsity in deep neural networks. In *Advances in neural information processing systems (NeurIPS)*, pages 2074–2082, 2016.

[35] Yushu Wu, Yifan Gong, Pu Zhao, Yanyu Li, Zheng Zhan, Wei Niu, Hao Tang, Minghai Qin, Bin Ren, and Yanzhi Wang. Compiler-aware neural architecture search for on-mobile real-time super-resolution. In *European Conference on Computer Vision*, pages 92–111. Springer, 2022.

[36] Tete Xiao, Yingcheng Liu, Bolei Zhou, Yuning Jiang, and Jian Sun. Unified perceptual parsing for scene understanding. In *Proceedings of the European conference on computer vision (ECCV)*, pages 418–434, 2018.

[37] Chenhongyi Yang, Zehui Chen, Miguel Espinosa, Linus Ericsson, Zhenyu Wang, Jiaming Liu, and Elliot J Crowley. Plainmamba: Improving non-hierarchical mamba in visual recognition. *arXiv preprint arXiv:2403.17695*, 2024.

[38] Ruichi Yu, Ang Li, Chun-Fu Chen, Jui-Hsin Lai, Vlad I Morariu, Xintong Han, Mingfei Gao, Ching-Yung Lin, and Larry S Davis. Nisp: Pruning networks using neuron importance score propagation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[39] Li Yuan, Yunpeng Chen, Tao Wang, Weihao Yu, Yujun Shi, Zi-Hang Jiang, Francis EH Tay, Jiashi Feng, and Shuicheng Yan. 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, 2021.

[40] Zheng Zhan, Yifan Gong, Pu Zhao, Geng Yuan, Wei Niu, Yushu Wu, Tianyun Zhang, Malith Jayaweera, David Kaeli, Bin Ren, et al. Achieving on-mobile real-time super-resolution with neural architecture and pruning search. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 4821–4831, 2021.

[41] Zheng Zhan, Zhenglun Kong, Yifan Gong, Yushu Wu, Zichong Meng, Hangyu Zheng, Xuan Shen, Stratis Ioannidis, Wei Niu, Pu Zhao, and Yanzhi Wang. Exploring token pruning in vision state space models. *arXiv preprint arXiv:2409.18962*, 2024.

[42] Zheng Zhan, Yushu Wu, Zhenglun Kong, Changdi Yang, Yifan Gong, Xuan Shen, Xue Lin, Pu Zhao, and Yanzhi Wang. Rethinking token reduction for state space models. *arXiv preprint arXiv:2410.14725*, 2024.

[43] Xiaosong Zhang, Yunjie Tian, Lingxi Xie, Wei Huang, Qi Dai, Qixiang Ye, and Qi Tian. Hivit: A simpler and more efficient design of hierarchical vision transformer. In *The Eleventh International Conference on Learning Representations*, 2022.

[44] Pu Zhao, Fei Sun, Xuan Shen, Pinrui Yu, Zhenglun Kong, Yanzhi Wang, and Xue Lin. Pruning foundation models for high accuracy without retraining. *arXiv preprint arXiv:2410.15567*, 2024.

[45] Bolei Zhou, Hang Zhao, Xavier Puig, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Scene parsing through ade20k dataset. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

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

Appendix

A Pseudo-code Example

Algorithm 1: PRUNING-AWARE HIDDEN STATE ALIGNMENT

```
#example code of pruning aware hidden state alignment
def pruning_aware_hsa(state, position_map, x, dt, A, B, C, y_ptr):
    dA = exp(A * dt);
    if position_map:
        #remained token computation as Eq.5
        dB = B * dt;
        state = state * dA + dB * x;
        y_ptr = &sum(state * C);
    else:
        #pruned token computation as Eq.6
        state = state * dA;
        x.ptr++;
    return state
```

This is a pseudo-code example of our pruning-aware hidden state alignment for demonstration.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: We explain method and summarize the contribution in introduction.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The limitation is included in conclusion section.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [NA]

Justification: Our paper does not include theoretical results.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: This paper fully discloses all the information needed to reproduce the main experimental results.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: The datasets and models we used is open-source, and we have provide our code in the footnote of page one.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification:We have specified all the training and test details necessary to understand the results

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment Statistical Significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: We report average results of multiple runs in our experimental section. Our paper does not report error bars.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)

- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We explain the computation resources in experiment section.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code Of Ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: Research is conducted in the paper conform with NeurIPS Code of Ethics.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [NA]

    Justification: Our paper is not highly related to societal impacts.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: Our paper poses no such risks. Our work does not release a new model.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: CC-BY 4.0, and we referenced the works that we used to implement our code.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.

- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: We provided our code in the footnote of page one.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: This paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: This paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.