ICCV #8505

Uncorrected author proof. Please cite as Lu, H., A.A. Salah, R. Poppe, "Snakes and Ladders: Two Steps Up for VideoMamba," Int. Conference on Computer Vision (ICCV), Honolulu, 2025. The supplementary material is included in this file at the end.

Snakes and Ladders: Two Steps Up for VideoMamba

Abstract

Video understanding requires the extraction of rich spatio-temporal representations, achieved by transformer models through self-attention. Unfortunately, self-attention poses a computational burden. In NLP, Mamba has surfaced as an efficient alternative for transformers. However, Mamba's successes do not trivially extend to vision tasks, including those in video analysis. In this paper, we theoretically analyze the differences between self-attention and Mamba. We identify two limitations in Mamba's token processing: historical decay and element contradiction. We propose VideoMambaPro (VMP) that addresses these limitations by adding masked backward computation and elemental residual connections to a VideoMamba backbone. VideoMambaPro models surpass VideoMamba by 1.6-3.0% and 1.1-1.9% top-1 on Kinetics-400 and Something-Something V2, respectively. Even without extensive pre-training, our models present an attractive and efficient alternative to current transformer models. Moreover, our two solutions are orthogonal to recent advances in Vision Mamba models, and are likely to provide further improvements in future models.

1. Introduction

Video understanding is challenging and requires models that can extract rich spatio-temporal representations from video inputs. Transformers are powerful neural networks capable of effectively capturing temporal and spatial information from videos [19, 29, 50]. Current state-of-the-art video understanding models are transformers [41, 51]. At the core of transformers is self-attention [49], the self-alignment between tokens in an input obtained by estimating the relative importance of a token with respect to all other tokens. The long-range token dependency accounts for much of the success of transformer models [5, 49].

Calculating self-attention is computationally costly, which eventually limits the application of powerful transformer models in practical settings [17]. Recently, alternative models with lower-cost operators have been proposed for national language processing (NLP), including S4 [13], RWKV [38], and RetNet [46]. Among these, Mamba [11] shows the best performance on long-range and causal tasks such as language understanding [31] and content-based reasoning [37].

Motivated by the favorable computational cost, researchers have recently extended Mamba from the NLP domain to the computer vision domain. The core adaptation involved splitting the input image into multiple regions and embedding these as continuous tokens [61]. For video understanding, the recently proposed VideoMamba [22] extracts key frames from videos as the continuous input sequence. However, compared to previous transformer-based methods, VideoMamba's performance on video benchmarks is markedly lower. For example, VideoMamba achieves 82.4% top-1 on Kinetics-400, compared to 85.2% for VideoMAE [47], indicating room for improvement.

In this paper, we first analyze differences in the feature extraction capabilities of transformers and Mamba. We identify two limitations of Mamba: historical decay and element contradiction. We then extend VideoMamba [22] to mitigate these limitations. The proposed VideoMambaPro (VMP) addresses historical decay through masked backward computation in the bi-directional Mamba process, allowing the network to better handle historical tokens. We introduce residual connections to Mamba's matrix elements to tackle element contradiction. VideoMambaPro consistently improves the performance of VideoMamba on video understanding tasks, positioning it as a strong, efficient competitor to transformers. Our contributions are:

- We derive a formal representation of Mamba from the perspective of self-attention and identify two limitations of Mamba in the video analysis domain.
- We propose VideoMambaPro, effectively addressing Mamba's limitations for video understanding.
- We report strong video action recognition performance compared to recent Vision Mamba-based models, and surpass the original VideoMamba by clear margins.

We first discuss related work. Then, we provide a theoretical analysis and introduce VideoMambaPro. Experiments are presented in Section 5. We conclude in Section 6.

ICCV #8505

2. Related Work

Transformers. One core aspect of transformers is self-attention [49] to achieve long-range interactions by measuring the similarity between tokens. Self-attention was introduced in the computer vision domain for tasks such as image recognition [26, 45] and object detection [7, 58]. Subsequent works (e.g., [9, 21, 47, 53] extended vision transformers to the video domain, to achieve superior performance. However, the mechanism of self-attention introduces significant computational overhead because of the similarity between all pairs of tokens needs to be calculated.

Alternative models. Recent work has introduced alternative models with reduced computational complexity, while maintaining the advantages of self-attention [30, 40, 59]. SOFT [30] uses Gaussian kernel functions to replace the dot-product similarity, which enables a full self-attention matrix to be approximated with a low-rank matrix decomposition. Combiner [40] employs a structured factorization to approximate full self-attention, realizing low computation and memory complexity.

Peng et al. [38] propose the Receptance Weighted Key Value (RWKV) architecture that combines self-attention training with an efficient recurrent neural network (RNN) inference using a linear attention mechanism. Through parallel computation, a lower, constant-level computational and memory complexity is achieved. RetNet [46] includes another variant of self-attention, by dividing the input into multiple chunks. Within each chunk, the self-attention mechanism can be computed in parallel, while information is transmitted between chunks based on an RNN.

State-space models. The S4 model completely abandons self-attention and, instead, builds upon a state space model [13]. Instead of calculatings a similarity matrix by performing matrix multiplications for pairs of tokens, it enables the network to directly learn a global high-order polynomial projection operator (HiPPO) matrix to handle relations between tokens. Additionally, for the simultaneous input of multiple tokens, S4 proposes a convolutional processing approach, enabling parallel training and thereby accelerating the training process.

Based on S4, Mamba [11] proposes a selection mechanism where, for each input token, a unique HiPPO matrix [12] is generated. This allows the model to selectively process input tokens, enabling it to focus on or ignore specific inputs. Due to Mamba's strong representation ability in NLP, and linear-time complexity, it has garnered attention as a promising alternative to transformers. In the computer vision domain, researchers have proposed Vision Mamba [61] and VMamba [27] for tasks such as image classification and object detection.

In the video domain, several Mamba variants have been proposed [22, 23, 34]. Their performance is somewhat lower than expected, with limited understanding of the causes. We argue that a systematic, mathematical analysis of Mamba from the perspective of self-attention could reveal shortcomings of Mamba's inner workings. Better understanding of these limitations allow us to develop improvements, and to close the accuracy performance gap with transformers, while enjoying Mamba's efficiency.

3. Theoretical Analysis

We revisit Mamba from the perspective of self-attention. Then, we analyze its limitations for video understanding. To address these, we propose VideoMambaPro in Section 4.

3.1. Mamba from the perspective of self-attention

Self-attention. Given an input sequence $\boldsymbol{X} := [\boldsymbol{x_1}, \cdots, \boldsymbol{x_N}] \in \mathbb{R}^{N \times D_x}$ of N feature vectors of depth D_x, self-attention [49, 60] computes the output sequence $\mathbf{Y}$ from $\boldsymbol{X}$ following two steps:

Step 1: Similarity matrix computation. The input sequence $\boldsymbol{X}$ is linearly projected onto the three different subspaces query $\mathbf{Q} \in \mathbb{R}^{N \times D}$, key $\mathbf{K} \in \mathbb{R}^{N \times D}$, and value $\mathbf{V} \in \mathbb{R}^{N \times D_V}$:

$$\mathbf{Q} = \boldsymbol{X}\mathbf{W}_Q^\top, \mathbf{K} = \boldsymbol{X}\mathbf{W}_K^\top, \mathbf{V} = \boldsymbol{X}\mathbf{W}_V^\top. \quad (1)$$

with $\mathbf{W}_Q, \mathbf{W}_K \in \mathbb{R}^{D \times D_x}$, and $\mathbf{W}_V \in \mathbb{R}^{D_v \times D_x}$ the corresponding weight matrices. Specifically, $\mathbf{Q} := [\boldsymbol{q_1}, \cdots, \boldsymbol{q_N}]^\top$, $\mathbf{K} := [\boldsymbol{k_1}, \cdots, \boldsymbol{k_N}]^\top$, and $\mathbf{V} := [\boldsymbol{v_1}, \cdots, \boldsymbol{v_N}]^\top$ with vectors $\boldsymbol{q_i}, \boldsymbol{k_i}, \boldsymbol{v_i}$ for $i = 1, \cdots, N$ the query, key, and value vectors, respectively, for input i. Based on $\mathbf{Q}$ and $\mathbf{K}$, similarity matrix $\mathbf{S} \in \mathbb{R}^{N \times N}$ contains the correlations between all query and key vectors, with a softmax function applied to each row of $\mathbf{S}$:

$$\mathbf{S} = \text{softmax}(\mathbf{Q}\mathbf{K}^\top / \sqrt{D}). \quad (2)$$

Each component s_{ij} $(i, j = 1, \cdots, N)$ represents the similarity score between $\boldsymbol{q_i}$ and $\boldsymbol{k_j}$.

Step 2: Output computation. Output sequence $\mathbf{Y} := [\boldsymbol{y_1}, \cdots, \boldsymbol{y_N}]^\top$ is then calculated based on $\mathbf{S}$ as:

$$\mathbf{Y} = \mathbf{S}\mathbf{V}. \quad (3)$$

It follows that each output vector $\boldsymbol{y_i}$ $(i = 1, \cdots, N)$ can be written in vector form as:

$$\boldsymbol{y_i} = \sum_{j=1}^{N} s_{ij} \boldsymbol{v_j}. \quad (4)$$

Any output vector $\boldsymbol{y_i}$ is a linear combination of vectors $\boldsymbol{v_j} (j = 1, \cdots, N)$, with similarity score s_{ij} serving as coefficient. The larger the similarity score, the greater the influence of $\boldsymbol{v_j}$ on output $\boldsymbol{y_i}$ [43].

Mamba. State Space Models (SSMs) serve as the foundation of Mamba [11]. They are based on continuous systems that map 1D functions or sequences, $x(t) \in \mathbb{R}^L \rightarrow$

ICCV #8505

$y(t) \in \mathbb{R}^L$ to output sequences $y(t)$ through a hidden state $h(t) \in \mathbb{R}^N$. Formally, SSM implements the mapping as[1]:

$$h(t) = \mathbf{A}h(t-1) + \mathbf{B}x(t), \tag{5}$$

$$y(t) = \mathbf{C}h(t) \tag{6}$$

where $\mathbf{A} \in \mathbb{R}^{N\times N}$ is the evolution matrix of the system, and $\mathbf{B} \in \mathbb{R}^{N\times 1}$, $\mathbf{C} \in \mathbb{R}^{N\times 1}$ are the projection matrices. Often, the input is discrete rather than a continuous function $x(t)$. Therefore, Mamba performs discretization, effectively creating a discrete version of the continuous system. A timescale parameter $\mathbf{\Delta}$ is used to transform the continuous parameters $\mathbf{A} and \mathbf{B}$ into their discrete counterparts $\overline{\mathbf{A}}, \overline{\mathbf{B}}$, and the transformation typically employs the zero-order hold method [57]. This process is expressed as:

$$\overline{\mathbf{A}} = \exp(\mathbf{\Delta A}), \tag{7}$$

$$\overline{\mathbf{B}} = (\mathbf{\Delta A})^{-1}(\exp(\mathbf{\Delta A}) - \mathbf{I}) \cdot \mathbf{\Delta B}, \tag{8}$$

$$h_t = \overline{\mathbf{A}}h_{t-1} + \overline{\mathbf{B}}x_t, \tag{9}$$

$$y_t = \mathbf{C}h_t. \tag{10}$$

Considering that parameters $\overline{\mathbf{A}}, \overline{\mathbf{B}}, \mathbf{C}$ in the original SSM are independent of the input data $x(t)$ and cannot be tailored to specific input data, Mamba employs a Selective Scan Mechanism as its core operator. More precisely, three functions $S_B(x), S_C(x), S_\Delta(x)$ are introduced to associate parameters $\overline{\mathbf{B}}, \mathbf{C}, and \mathbf{\Delta}$ in Eqs. 7–10 to the input data x. Based on $S_\Delta(x)$, $\overline{\mathbf{A}}$ can also be associated with the input data x. For example, given the input x_1, functions $S_\Delta(x)$ will produce the corresponding $\overline{\mathbf{A}}_1$ based on Eq. 7, and functions $S_B(x)$ and $S_\Delta(x)$ will produce the corresponding $\overline{\mathbf{B}}_1$ based on Eq. 8. $\overline{\mathbf{C}}_1$ is obtained based on function $S_C(x)$. Following Eqs. 9 and 10, we analyze the process to obtain output sequence $\mathbf{Y}$ when given an input sequence $\boldsymbol{X} := [\boldsymbol{x_1}, \cdots, \boldsymbol{x_N}] \in \mathbb{R}^{N\times D_x}$ of N feature vectors. Each vector's hidden state is denoted as:

$$h_1 = \overline{\mathbf{B}}_1 \boldsymbol{x_1}, \tag{11}$$

$$\begin{aligned} h_2 &= \overline{\mathbf{A}}_2 h_1 + \overline{\mathbf{B}}_2 \boldsymbol{x_2} \\ &= \overline{\mathbf{A}}_2\overline{\mathbf{B}}_1 \boldsymbol{x_1} + \overline{\mathbf{B}}_2 \boldsymbol{x_2}, \end{aligned} \tag{12}$$

$$\begin{aligned} h_3 &= \overline{\mathbf{A}}_3 h_2 + \overline{\mathbf{B}}_3 \boldsymbol{x_3} \\ &= \overline{\mathbf{A}}_3\overline{\mathbf{A}}_2\overline{\mathbf{B}}_1 \boldsymbol{x_1} + \overline{\mathbf{A}}_3\overline{\mathbf{B}}_2 \boldsymbol{x_2} + \overline{\mathbf{B}}_3 \boldsymbol{x_3}, \end{aligned} \tag{13}$$

$$\cdots$$

$$\begin{aligned} h_N &= \overline{\mathbf{A}}_N h_{N-1} + \overline{\mathbf{B}}_N \boldsymbol{x_N} \\ &= \overline{\mathbf{A}}_N\overline{\mathbf{A}}_{N-1}\cdots\overline{\mathbf{A}}_2\overline{\mathbf{B}}_1 \boldsymbol{x_1} + \overline{\mathbf{A}}_N\overline{\mathbf{A}}_{N-1}\cdots\overline{\mathbf{A}}_3\overline{\mathbf{B}}_2 \boldsymbol{x_2} \\ &\quad + \overline{\mathbf{A}}_N\overline{\mathbf{B}}_{N-1} \boldsymbol{x_{N-1}} + \overline{\mathbf{B}}_N \boldsymbol{x_N}. \end{aligned} \tag{14}$$

[1]The original SSM [13] employs $h'(t) = Ah(t) + Bx(t)$, with $h(t)$ the hidden state from previous time step $t-1$, and $h'(t)$ the updated current hidden state, replacing $h(t)$. Considering this approach may lead to ambiguity, we have adopted the updated description.

Eqs. 11–14 can be written in matrix form:

$$\begin{aligned} \mathbf{H} &= [h_1, h_2, h_3, \cdots, h_N]^\top \\ &= \begin{bmatrix} \overline{\mathbf{B}}_1 & 0 & 0 & \cdots & 0 \\ \overline{\mathbf{A}}_2\overline{\mathbf{B}}_1 & \overline{\mathbf{B}}_2 & 0 & \cdots & 0 \\ \overline{\mathbf{A}}_3\overline{\mathbf{A}}_2\overline{\mathbf{B}}_1 & \overline{\mathbf{A}}_3\overline{\mathbf{B}}_2 & \overline{\mathbf{B}}_3 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ (\prod_{\mathbf{j=N}}^{\mathbf{2}} \overline{\mathbf{A}}_\mathbf{j})\overline{\mathbf{B}}_1 & (\prod_{\mathbf{j=N}}^{\mathbf{3}} \overline{\mathbf{A}}_\mathbf{j})\overline{\mathbf{B}}_2 & (\prod_{\mathbf{j=N}}^{\mathbf{4}} \overline{\mathbf{A}}_\mathbf{j})\overline{\mathbf{B}}_3 & \cdots & \overline{\mathbf{B}}_N \end{bmatrix} \begin{bmatrix} \boldsymbol{x_1} \\ \boldsymbol{x_2} \\ \boldsymbol{x_3} \\ \vdots \\ \boldsymbol{x_N} \end{bmatrix}. \end{aligned} \tag{15}$$

For output sequence $\mathbf{Y} := [\boldsymbol{y_1}, \cdots, \boldsymbol{y_N}]^\top$, each vector $\boldsymbol{y_i}$ $(i = 1, \cdots, N)$ can be expressed as:

$$\boldsymbol{y_N} = \overline{\mathbf{C}}_N h_N, \tag{16}$$

and in matrix form as:

$$\mathbf{Y} = \begin{bmatrix} \overline{\mathbf{C}}_1 & 0 & 0 & \cdots & 0 \\ 0 & \overline{\mathbf{C}}_2 & 0 & \cdots & 0 \\ 0 & 0 & \overline{\mathbf{C}}_3 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & \overline{\mathbf{C}}_N \end{bmatrix} \begin{bmatrix} h_1 \\ h_2 \\ h_3 \\ \vdots \\ h_N \end{bmatrix} = \overline{\mathbf{C}}h. \tag{17}$$

By substituting Eq. 15 into Eq. 17, we obtain:

$$\mathbf{Y} = \overline{\mathbf{C}} \begin{bmatrix} \overline{\mathbf{B}}_1 & 0 & 0 & \cdots & 0 \\ \overline{\mathbf{A}}_2\overline{\mathbf{B}}_1 & \overline{\mathbf{B}}_2 & 0 & \cdots & 0 \\ \overline{\mathbf{A}}_3\overline{\mathbf{A}}_2\overline{\mathbf{B}}_1 & \overline{\mathbf{A}}_3\overline{\mathbf{B}}_2 & \overline{\mathbf{B}}_3 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ (\prod_{\mathbf{j=N}}^{\mathbf{2}} \overline{\mathbf{A}}_\mathbf{j})\overline{\mathbf{B}}_1 & (\prod_{\mathbf{j=N}}^{\mathbf{3}} \overline{\mathbf{A}}_\mathbf{j})\overline{\mathbf{B}}_2 & (\prod_{\mathbf{j=N}}^{\mathbf{4}} \overline{\mathbf{A}}_\mathbf{j})\overline{\mathbf{B}}_3 & \cdots & \overline{\mathbf{B}}_N \end{bmatrix} \begin{bmatrix} \boldsymbol{x_1} \\ \boldsymbol{x_2} \\ \boldsymbol{x_3} \\ \vdots \\ \boldsymbol{x_N} \end{bmatrix}, \tag{18}$$

which can be expressed as:

$$\mathbf{Y} = \overline{\mathbf{C}}(\mathbf{M}\boldsymbol{X}), \tag{19}$$

where $\mathbf{M}$ represents the second term on the right-hand side of Eq. 18. Recall from Eq. 3 that the result $\mathbf{Y}$ obtained by self-attention processing can be expressed as:

$$\mathbf{Y} = \mathbf{S}\mathbf{V} = (\mathbf{S}\boldsymbol{X})\mathbf{W}_V^\top \tag{20}$$

From the perspective of self-attention, by comparing Eqs. 19 and 20, the essence of Mamba is to generate a matrix $\mathbf{M}$ similar to similarity matrix $\mathbf{S}$, such that the result of $\mathbf{M}\boldsymbol{X}$ is based on the correlation between vectors of $\boldsymbol{X}$. Although the final result of $\mathbf{M}\boldsymbol{X}$ is left multiplied by a mapping matrix $\mathbf{C}$, while the result of $\mathbf{S}\boldsymbol{X}$ is right multiplied by a mapping matrix $\mathbf{W}_V^\top$, the geometric meaning of the two are the same.

3.2. Limitations of Mamba in video understanding

From the perspective of self-attention, the concept of Mamba is similar: both use similarity matrices. We now analyze the differences between the similarity matrices of Mamba and self-attention, and discuss the limitations of Mamba in the context of the video understanding task.

Limitation 1: Historical decay. Matrix $\mathbf{M}$ in Eq. 19 corresponds to the second right-hand term in Eq. 18, which is a lower triangular matrix of the form:

$$\mathbf{M} = \begin{bmatrix} m_{11} & 0 & 0 & \cdots & 0 \\ m_{21} & m_{22} & 0 & \cdots & 0 \\ m_{31} & m_{32} & m_{33} & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ m_{N1} & m_{N2} & m_{N3} & \cdots & m_{NN} \end{bmatrix}. \quad (21)$$

By comparing $\mathbf{M}$ with matrix $\mathbf{S}$ in self-attention, we find that outputs in Mamba favor more recent information, because the more weights are zero, the earlier the token is observed. For example, for input $[\boldsymbol{x_1}, \boldsymbol{x_2}, \boldsymbol{x_3}]$, the output $\mathbf{M}\boldsymbol{x_1}$ in Mamba is $m_{11}\boldsymbol{x_1}$ while the output $\mathbf{S}\boldsymbol{x_1}$ is $s_{11}\boldsymbol{x_1} + s_{12}\boldsymbol{x_2} + s_{13}\boldsymbol{x_3}$ in self-attention. This indicates that, in Mamba, the influence of earlier observed tokens on the final result is greatly diminished. We refer to this limitation as historical decay.

In the NLP domain, more recent dialogue information often has more impact on the final judgment, so this effect is acceptable. However, in the computer vision domain, the order of the tokens has less meaning. Previous works such as Vision Mamba [61] and VMamba [27] tried to solve this issue by processing the token sequence in both forward and backward directions. This produces better results, but significant deficiencies still exist. We will explain this theoretically below.

When processing bi-directionally, the results generated from input forward tokens $[\boldsymbol{x_1}, \cdots, \boldsymbol{x_N}]$, denoted as $\mathbf{M}_f\boldsymbol{X}$, and the results generated from input backward tokens $[\boldsymbol{x_N}, \cdots, \boldsymbol{x_1}]$, denoted as $\mathbf{M}_b\boldsymbol{X}$, are linearly combined to generate the final result $\mathbf{M}_{bi}\boldsymbol{X}$ with $\mathbf{M}_{bi}$ a dense matrix. As a result, the influence of historical information on the result is increased, consequently leading to better results.

For example, for the input tokens $[\boldsymbol{x_1}, \boldsymbol{x_2}, \boldsymbol{x_3}]$, $\mathbf{M}_f\boldsymbol{X}$ and $\mathbf{M}_b\boldsymbol{X}$ can be expressed as:

$$\mathbf{M}_f\boldsymbol{X} = \begin{bmatrix} f_{11} & 0 & 0 \\ f_{21} & f_{22} & 0 \\ f_{31} & f_{32} & f_{33} \end{bmatrix} \begin{bmatrix} \boldsymbol{x_1} \\ \boldsymbol{x_2} \\ \boldsymbol{x_3} \end{bmatrix} = \begin{bmatrix} h_{1f} \\ h_{2f} \\ h_{3f} \end{bmatrix}, \quad (22)$$

$$\mathbf{M}_b\boldsymbol{X} = \begin{bmatrix} b_{33} & 0 & 0 \\ b_{23} & b_{22} & 0 \\ b_{13} & b_{12} & b_{11} \end{bmatrix} \begin{bmatrix} \boldsymbol{x_3} \\ \boldsymbol{x_2} \\ \boldsymbol{x_1} \end{bmatrix} = \begin{bmatrix} h_{3b} \\ h_{2b} \\ h_{1b} \end{bmatrix}, \quad (23)$$

where f_{ij} represents the similarity score during the forward process, and b_{ij} is the similarity score in the backward direction. After bi-directional computation, with the outputs linearly combined, the results are expressed as:

$$\begin{aligned} h_1 &= h_{1f} + h_{1b} = f_{11}\boldsymbol{x_1} + b_{13}\boldsymbol{x_3} + b_{12}\boldsymbol{x_2} + b_{11}\boldsymbol{x_1} \\ h_2 &= h_{2f} + h_{2b} = f_{21}\boldsymbol{x_1} + f_{22}\boldsymbol{x_2} + b_{23}\boldsymbol{x_3} + b_{22}\boldsymbol{x_2} \\ h_3 &= h_{3f} + h_{3b} = f_{31}\boldsymbol{x_1} + f_{32}\boldsymbol{x_2} + f_{33}\boldsymbol{x_3} + b_{33}\boldsymbol{x_3} \end{aligned} \quad (24)$$

We can write Eq. 24 in matrix form:

$$\begin{aligned} \begin{bmatrix} h_1 \\ h_2 \\ h_3 \end{bmatrix} &= \begin{bmatrix} f_{11} + b_{11} & b_{12} & b_{13} \\ f_{21} & f_{22} + b_{22} & b_{23} \\ f_{31} & f_{32} & f_{33} + b_{33} \end{bmatrix} \begin{bmatrix} \boldsymbol{x_1} \\ \boldsymbol{x_2} \\ \boldsymbol{x_3} \end{bmatrix} \\ &= \mathbf{M}_{bi} \begin{bmatrix} \boldsymbol{x_1} \\ \boldsymbol{x_2} \\ \boldsymbol{x_3} \end{bmatrix}. \end{aligned} \quad (25)$$

The bi-directional computation transforms matrix $\mathbf{M}$ from a lower triangular matrix to a dense matrix $\mathbf{M}_{bi}$, thereby capturing more historical information and effectively avoiding historical decay. When extending to the case of N input tokens $[\boldsymbol{x_1}, \cdots, \boldsymbol{x_N}]$, $\mathbf{M}_{bi}$ can be written as:

$$\mathbf{M}_{bi} = \begin{bmatrix} f_{11} + b_{11} & b_{12} & b_{13} & \cdots & b_{1N} \\ f_{21} & f_{22} + b_{22} & b_{23} & \cdots & b_{2N} \\ f_{31} & f_{32} & f_{33} + b_{33} & \cdots & b_{3N} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ f_{N1} & f_{N2} & f_{N3} & \cdots & f_{NN} + b_{NN} \end{bmatrix}. \quad (26)$$

The diagonal elements of $\mathbf{M}_{bi}$ contain duplicates of the similarity between a token and itself. For example, f_{33} and b_{33} each represent the similarity between token $\boldsymbol{x_3}$ and itself. Consequently, the similarity is effectively doubled which weakens the association with other tokens. One possible approach is to adjust $\mathbf{M}_f$ and $\mathbf{M}_b$ using a weight coefficient z through a linear combination. However, this method is not very effective. For example, to prevent the similarity of diagonal elements from being doubled, the z value is set to 0.5. However, after multiplying other elements by 0.5, their weight values are significantly reduced, which means that the association of each token with other tokens is weakened.

Limitation 2: Element contradiction. By analyzing non-zero elements m_{ij} in $\mathbf{M}$ of Eq. 18, we derive:

$$m_{ij} = \overline{\mathbf{A}}_i m_{i-1,j} \quad (27)$$

After multiple iterations, the above equation results in implicit consideration of the correlation between previous tokens and token j when computing the correlation between token i and token j. As a result, m_{ij} exhibits stronger contextual dependencies compared to the elements s_{ij} in the matrix $\mathbf{S}$. This might explain why Mamba achieves better performance than transformers in the field of NLP. While this is advantageous in the NLP domain, for the computer vision domain, input tokens often lack semantic connections. The consideration of the influence of other tokens on each element can lead to significant drawbacks.

Interleaved token structures are common when processing images. Tokens that "belong together" might not be subsequently processed. For example, in an image classification task, input tokens $[\boldsymbol{x_1}, \boldsymbol{x_2}, \boldsymbol{x_3}]$ might represent image regions $[dog, other, dog]$. Ideally, m_{31} should be high and

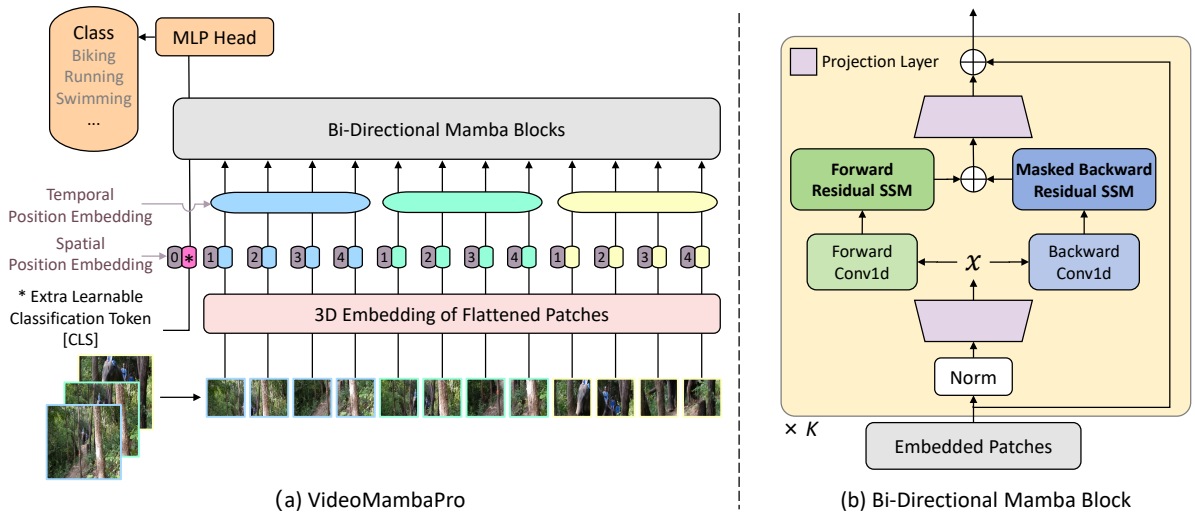

Figure 1. (a) Framework of VideoMambaPro with K bi-directional Mamba blocks. (b) In each bi-directional Mamba block, we employ forward residual SSM and masked backward Residual SSM.

m_{21} low. Following Eq. 27, $m_{31} = \overline{\mathbf{A}}_3 m_{21}$, so $\overline{\mathbf{A}}_3$ needs to increase. However, this causes $m_{32} = \overline{\mathbf{A}}_3 m_{22}$ to increase because m_{22} is also high. But, theoretically, m_{32} should be low. This causes an element contradiction. Especially for video understanding, such contradictions are common because most video regions contain background and other irrelevant information, making relevant tokens sparse. Consequently, the performance of Mamba applied to video analysis tasks is underwhelming [22, 24, 56].

4. VideoMambaPro

We use VideoMamba [22] as backbone, and propose two adaptations to address historical decay and element contradiction. The resulting architecture is termed VideoMambaPro (VMP). With minor adjustments, our adaptations can also be applied to related Mamba models.

To address historical decay, we keep the result of $\mathbf{M}_f \boldsymbol{X}$ but we use masked computation in the backward process. Specifically, we assign a mask to the diagonal elements of $\mathbf{M}_b$, setting their values to 0, and then proceed with the calculations in Eqs 21–25. We thus eliminate the duplicate similarity on the diagonal, without affecting other elements. The final $\mathbf{M}_{bi}$ is expressed as:

$$M_{bi} = \begin{bmatrix} f_{11} & b_{12} & b_{13} & \cdots & b_{1N} \\ f_{21} & f_{22} & b_{23} & \cdots & b_{2N} \\ f_{31} & f_{32} & f_{33} & \cdots & b_{3N} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ f_{N1} & f_{N2} & f_{N3} & \cdots & f_{NN} \end{bmatrix}. \quad (28)$$

To solve element contradiction, we propose residual SSM. This solution is inspired by residual connections, to distribute the requirement for $\overline{\mathbf{A}}_i$ in m_{ij} over multiple $\overline{\mathbf{A}}_i$. This helps to avoid contradictions caused by interleaved sequence structures. For example, for our previous example input sequence $[\boldsymbol{x_1}, \boldsymbol{x_2}, \boldsymbol{x_3}]$, which represents regions $[dog, other, dog]$, we let $m_{31} = \overline{\mathbf{A}}_3 m_{21} + \overline{\mathbf{A}}_3$. This way, the requirement for a single $\overline{\mathbf{A}}_3$ can be split into two parts, thus avoiding contradictions. This can be expressed as:

$$m_{ij} = \overline{\mathbf{A}}_i m_{i-1,j} + \overline{\mathbf{A}}_i \quad (29)$$

We implement these solutions into VideoMamba [22], to form VideoMambaPro (see Figure 1). Given input video $\boldsymbol{X}^v \in \mathbb{R}^{3\times T\times H\times W}$, we first use a 3D convolution with a $1 \times 16 \times 16$ kernel to convert $\boldsymbol{X}^v$ into L non-overlapping patch-wise tokens $\boldsymbol{X}^p \in \mathbb{R}^{L\times C}$ with $L = t \times h \times w$ ($t = T, h = \frac{H}{16}, w = \frac{W}{16}$). Because SSM is sensitive to token positions, and in line with [22], we include learnable spatial and temporal position embeddings $\boldsymbol{p_s} \in \mathbb{R}^{(hw+1)\times C}$ and $\boldsymbol{p_t} \in \mathbb{R}^{t\times C}$. Input tokens $\boldsymbol{X}$ are expressed as:

$$\boldsymbol{X} = [\boldsymbol{X}_{cls}, \boldsymbol{X}] + \boldsymbol{p_s} + \boldsymbol{p_t}, \quad (30)$$

where $\boldsymbol{X}_{cls}$ is a learnable classification token positioned at the start of the sequence. Input tokens $\boldsymbol{X}$ pass through K Mamba blocks, and the final layer's [CLS] token is used for classification, after normalization and linear projection.

5. Experiments

5.1. Experimental setup

Datasets. We evaluate VideoMambaPro on five video benchmarks: (a) Kinetics-400 (K400, [3]) comprises ~240K training and ~20K validation videos, each with an average duration of 10 seconds and categorized into 400 classes. (b) Something-Something V2 (SSv2, [10]) includes ~160K training and ~20K validation videos with

an average duration of 4 seconds, and 174 motion-centric classes. (c) UCF-101 [44] is a relatively small dataset, consisting of ∼9.5K training and ∼3.5K validation videos. (d) HMDB51 [18] is also a compact video dataset, containing ∼3.5K training and ∼1.5K validation videos. (e) AVA [14] is a dataset for spatio-temporal localization of human actions with ∼211k and ∼57k validation video segments.

Implementation. In line with VideoMamba, we introduce three models with increasing embedding dimension and number of bi-directional Mamba blocks K: Tiny, Small, and Middle (details in supplementary material). To compare with VideoMamba, we pre-train VideoMambaPro on ImageNet-1K (IN-1K). On K400, we also pre-train with IN-1K, fine-tune on the training set and report on the validation set. For K400, we also report on the larger 336^2 input size. During pre-training, we follow DeiT [48] by applying a center crop to obtain the 224^2 sized images. We apply random cropping, random horizontal flipping, label-smoothing regularization, mix-up, and random erasing as data augmentations. We use AdamW [28] with a momentum of 0.9, a batch size of 1024, and a weight decay of 0.05. We employ a cosine learning rate schedule during training, 1×10^{-3} initial learning rate over 300 epochs. The fine-tuning settings follow VideoMAE [47]. We resize frames to 224^2, and use AdamW with a momentum of 0.9 and a batch size of 512. The evaluation process is the same as VideoMamba, and more details are in the supplementary materials. These materials also include results on IN-1K image classification.

Method	Pre-train	Input	Crops	Param	FLOP	Top1	Top5
MViTv1-B [6]		32×224^2	5×1	37M	350G	80.2	94.4
MViTv2-S [25]		16×224^2	5×1	35M	320G	81.0	94.6
Uniformer-S [20]	IN-1K	16×224^2	4×1	21M	168G	80.8	94.7
Uniformer-B [20]	IN-1K	16×224^2	4×1	50M	388G	82.0	95.1
Uniformer-B [20]	IN-1K	32×224^2	4×3	50M	3.1T	83.0	95.4
STAM [42]	IN-21K	64×224^2	1×1	121M	1.0T	79.2	-
TimeSformer-L [2]	IN-21K	96×224^2	1×3	121M	7.1T	80.7	94.7
ViViT-L [1]	IN-21K	16×224^2	4×3	311M	47.9T	81.3	94.7
Mformer-HR [36]	IN-21K	16×336^2	10×3	311M	28.8T	81.1	95.2
VideoMAE-H [47]	IN-21K	16×224^2	5×3	633M	17.9T	86.6	97.1
X-CLIP-L/14 [32]	CLIP-400M	16×336^2	4×3	453M	37.0T	87.7	—
MTV-H [55]	60M[1]	32×224^2	4×3	1120M	44.5T	89.1	98.2
InternVideo-1B [51]	412M[2]	64×224^2	16×4	1300M	86.2T	91.1	**98.9**
InternVideo2-1B [52]	414M[3]	16×224^2	16×4	1000M	—	91.6	—
InternVideo2-6B [52]	414M[3]	16×224^2	16×4	5903M	—	**92.1**	—
VideoMamba-Ti	IN-1K	32×224^2	4×3	7M	0.4T	78.8	93.9
VideoMamba-Ti	IN-1K	64×384^2	4×3	7M	2.4T	80.3	94.8
VideoMamba-S	IN-1K	32×224^2	4×3	26M	1.6T	81.5	95.2
VideoMamba-S	IN-1K	64×384^2	4×3	26M	4.7T	82.7	95.6
VideoMamba-M	IN-1K	32×224^2	4×3	74M	4.8T	82.4	95.7
VideoMamba-M	IN-1K	64×384^2	4×3	74M	28.4T	83.3	96.1
VideoMambaPro-Ti	IN-1K	32×224^2	4×3	7M	0.4T	81.6	95.9
VideoMambaPro-Ti	IN-1K	64×384^2	4×3	7M	2.2T	83.3	96.1
VideoMambaPro-S	IN-1K	32×224^2	4×3	25M	1.6T	83.3	96.0
VideoMambaPro-S	IN-1K	64×384^2	4×3	25M	4.4T	84.5	96.6
VideoMambaPro-M	IN-1K	32×224^2	4×3	72M	4.7T	84.0	96.4
VideoMambaPro-M	IN-1K	64×384^2	4×3	72M	27.0T	**85.0**	**96.7**

Table 1. Performance on K400. Top part of the table are Transformer models, bottom part are Mamba models. We report crops (temporal × spatial) and FLOPs for inference. —: not reported.
[1] IN-21K+WTS
[2] CLIP-400M+WebVid+HowTo+K710+SSv2+AVA2.2+more.
[3] LAION-300M+KMash+WebVid+InternVid+LLaVA+more.

5.2. Comparison with state-of-the-art

K400. Results appear in Table 1. Compared to VideoMamba, VideoMambaPro has slightly fewer parameters and FLOPs. This is primarily because VideoMamba employs an additional projection layer to generate the weight coefficient z to adjust $\mathbf{A}_f$ and $\mathbf{A}_b$. See the supplementary materials for an architecture comparison. VideoMambaPro outperforms VideoMamba across model and input sizes. With 224^2 inputs and pre-trained only on IN-1K, the best-performing VideoMambaPro-M achieves a top-1 accuracy of 84.0%, 1.6% higher than VideoMamba-M. Further comparisons appear in Section 5.4. Increasing the input size to 336^2 leads to a performance improvement of 1.0-1.7%.

VideoMambaPro scores lower than the recent InternVideo2-1B [52] by 7.6%, but was only pre-trained on IN-1K and has significantly fewer parameters (1000M vs 72M) and inference only takes ∼5.5% of the FLOPs.

Method	Pre-train	Input	Crops	Param	FLOP	Top1	Top5
MViTv1-B [6]	K400	16×224^2	1×3	37M	213G	64.7	89.2
MViTv1-B [6]	K400	32×224^2	1×3	37M	510G	67.1	90.8
MViTv2-S [25]	K400	16×224^2	1×3	35M	195G	68.2	91.4
MViTv2-B [25]	K400	32×224^2	1×3	51M	675G	70.5	92.7
Uniformer-S [20]	IN-1K+K400	16×224^2	1×3	21M	126G	67.7	91.4
Uniformer-B [20]	IN-1K+K400	16×224^2	1×3	50M	291G	70.4	92.8
TimeSformer-L [2]	IN-21K	16×224^2	1×3	121M	5.1T	62.5	-
ViViT-L [1]	IN-21K+K400	16×224^2	4×3	311M	47.9T	65.4	89.8
Mformer-HR [36]	IN-21K+K400	16×336^2	1×3	311M	3.6T	68.1	91.2
MaskFeat-L [53]	IN-21K	64×312^2	4×3	218M	8.5T	75.0	95.0
VideoMAE-L [47]	IN-21K	32×224^2	1×3	305M	4.3T	75.4	95.2
TubeViT-L [39]	IN-1K	32×224^2	4×3	311M	9.5T	76.1	95.2
InternVideo-1B [51]	See Table 1	64×224^2	16×4	1300M	86.2T	77.2	95.9
InternVideo2-1B [52]	See Table 1	64×224^2	16×4	1000M	—	77.1	—
InternVideo2-6B [52]	See Table 1	64×224^2	16×4	5903M	—	77.4	—
VideoMamba-Ti	IN-1K	16×224^2	2×3	7M	102G	66.0	89.6
VideoMamba-S	IN-1K	16×224^2	2×3	26M	408G	67.6	90.9
VideoMamba-M	IN-1K	16×224^2	4×3	74M	2.4T	68.3	91.4
VideoMambaPro-Ti	IN-1K	16×224^2	2×3	7M	96G	67.9	91.2
VideoMambaPro-S	IN-1K	16×224^2	2×3	25M	382G	68.8	91.4
VideoMambaPro-M	IN-1K	16×224^2	4×3	72M	2.2T	69.4	91.6

Table 2. Performance on SSv2. —: not reported. Top part of the table are Transformer models, bottom part are Mamba models.

SSv2. Results appear in Table 2. VideoMambaPro outperforms VideoMamba by 1.1–1.9%. It also outperforms several popular transformer models. Although InternVideo-1B [51] and InternVideo2-6B [52] outperform our VideoMambaPro-M by 7.8% and 8.0%, respectively, they require 18.0-82 times more parameters and at least 39 times more FLOPs. Again, we expect that the performance for VideoMambaPro will increase with more pre-training.

UCF-101/HMDB51/AVA V2.2. From Table 3, it shows that VideoMambaPro-M is competitive, and outperforms VideoMamba by 3.4% and 1.8% on UCF-101 and HMDB51, respectively. VideoMambaPro-M achieves 31.9 mAP on AVA V2.2, which is 10.7% lower than VideoMAE V2 [50] but with an order of magnitude fewer parameters and FLOPs and pre-trained only on IN-1K (see Table 4).

ICCV #8505

Method	Params	UCF-101	HMDB51
VideoMoCo [33]	15M	78.7	49.2
CoCLR [16]	9M	81.4	52.1
MemDPC [15]	32M	86.1	54.5
Vi^{2}CLR [4]	9M	89.1	55.7
VideoMAE [47]	87M	91.3	62.6
GDT [35]	33M	95.2	72.8
VideoMAE V2 [50]	1050M	**99.6**	**88.1**
VideoMamba-M	74M	88.2	60.8
VideoMambaPro-M	72M	91.6	63.2

Table 3. Results on UCF-101 and HMDB51.

Method	FLOPs	Param	mAP
SlowFast R101 [8]	138G	53M	23.8
VideoMAE-B [47]	180G	87M	26.7
MViTv2-B [25]	225G	51M	30.5
ObjectTransformer [54]	243G	86M	31.0
MViTv2-L [25]	2828G	213M	34.4
ST-MAE-H [9]	1193G	632M	36.2
VideoMAE V2 [50]	4220G	1050M	42.6
VideoMamba-M [19]	202G	74M	30.1
VideoMambaPro-M	183G	72M	31.9

Table 4. Results on AVA V2.2.

Models	Input	Top-1	Top-5
VideoMamba-M (baseline)	32×224^2	82.4	95.7
VideoMambaPro-M (w/o residual)	32×224^2	83.6 (+1.2)	96.0 (+0.3)
VideoMambaPro-M (w/o masking)	32×224^2	83.0 (+1.0)	95.8 (+0.1)
VideoMambaPro-M	32×224^2	84.0 (+1.6)	96.4 (+0.7)

Table 5. Ablation study on K400, with and without masked backward computation and elemental residual connections.

5.3. Ablation study

Influence of masked backward computation and elemental residual connection. We have identified two limitations that exist in VideoMamba: historical decay and element contradiction. We introduced masked backward computation and elemental residual connections to address these respective issue. Here, we analyze the impact of each solution. We use the same settings as before, with VideoMambaPro-M and pre-training on IN-1K. We summarize the performance of VideoMambaPro-M on K400 in Table 5. Both solutions contribute to an improved score, and their effect is partly complementary. This indicates that the two limitations exist simultaneously in VideoMamba.

Evidence of limitations of Mamba. We have theoretically identifies historical decay and element contradiction as fundamental problems in Mamba, and propose the masked backward computation and elemental residual connection to solve the limitations. Previous experiments on image and video action recognition demonstrated the effectiveness of our approach, and we further provide concrete evidence that the increased performance stems from addressing these two issues.

We performed two tests on ImageNet-1K image classification task to build the connection between the issues in practice and our approach. First, to investigate whether the residual connection alleviates element contradiction, we randomly replaced a percentage of the patches by randomly selected patches from other classes, which will increase element contradiction because more irrelevant information is present. The table 6 shows that the relative performance of VideoMambaPro over VideoMamba increases to 9.2% when 20% of the patches is replaced. For higher percentages, the gap is reduced along with the overall score. This demonstrates that irrelevant (or even misleading) information can better be dealt with using our approach.

	Replacement ratio						
	0	5	10	20	30	50	80
VideoMamba-Ti	76.9	73.2	68.0	59.4	52.9	47.6	28.9
VideoMambaPro-Ti	78.9	78.0	76.1	68.6	55.9	48.2	29.1

Table 6. Top-1 accuracy of VideoMambaPro-Ti and VideoMamba-Ti with various replacement ratios (in %) on ImageNet-1K.

Second, to verify that masking addresses the issue of historical decay, we adopted a progressive masking approach, gradually decreasing the mask values of the diagonal elements of $\mathbf{M}_b$ from 1 to 0, and then retrained the model on ImageNet-1K. The results below show that VideoMambaPro's accuracy gradually increases when more masking is applied, evidencing the merits of the approach.

	Mask value			
	1	0.7	0.5	0
VideoMambaPro-Ti	76.9	77.0	77.2	77.8

Table 7. Top-1 accuracy across mask values on ImageNet-1K.

5.4. Comparison with VideoMamba on K400

We more thoroughly compare the differences between VideoMamba and VideoMambaPro by investigating the relative performance per class. We then present a statistical comparison between the results of both backbones.

Class analysis. We compare VideoMambaPro-M with 224^2 image size pre-trained on IN-1K to a VideoMamba-M baseline with the same settings. We show the relative performance for all classes of K400 in Figure 2. For over 95% of the classes, VideoMambaPro shows improvement. Although there is a lower performance for certain classes, the decrease is typically limited.

The majority of the classes sees a $\sim$1.8% improvement, which is substantial. For a small number of classes, VideoMambaPro performs $>$2% better than VideoMamba. Only a fraction of the classes is negatively affected by the solutions introduced in VideoMambaPro.

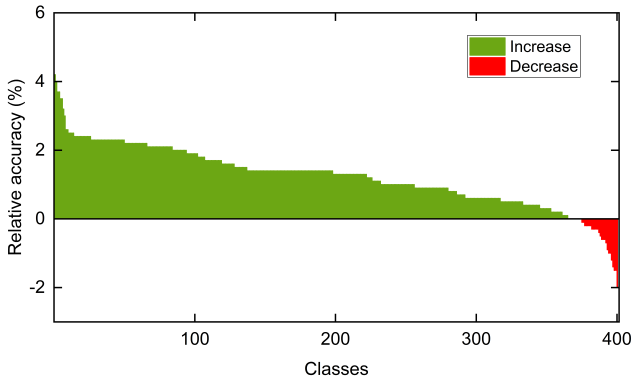

Figure 2. Relative accuracy per class on Kinetics-400 by comparing VideoMambaPro-M to VideoMamba-M.

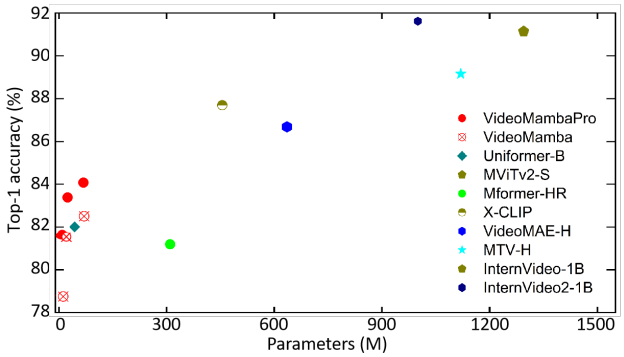

Figure 3. Top-1 accuracy versus number of parameters of VideoMambaPro and other models on Kinetics-400.

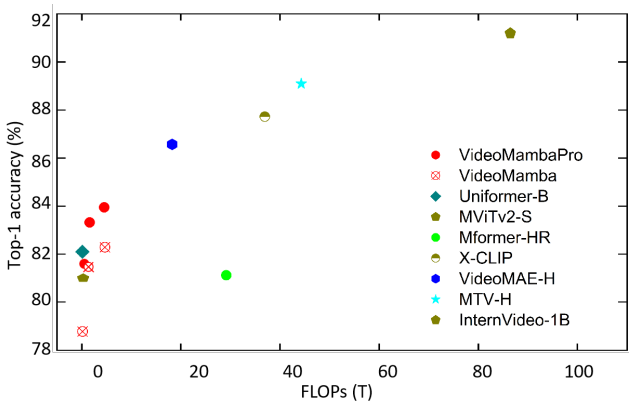

Figure 4. Top-1 accuracy versus number of FLOPs of VideoMambaPro and other models on Kinetics-400.

Statistical comparison. In order to understand whether the improvements of VideoMambaPro over VideoMamba are statistically significant, we compare the K400 results of the respective Middle models, both pre-trained on IN-1K and size 224×224. Other settings are also the same. For each test sample, we check whether if it correctly classified by either model. The results appear in Table 8.

		VideoMambaPro-M		
		True	False	Total
VideoMamba-M	True	14,302	469	14,771
	False	1,833	3,302	5,135
	Total	16,135	3,771	19,906

Table 8. Contingency table for K400 test items for VideoMamba-M and VideoMambaPro-M.

We used the McNemar test, a non-parametric test with a single degree of freedom. It checks whether the number of items that are incorrectly classified by VideoMambaPro (n_{10}) but not VideoMamba is substantially lower than the number of items misclassified by VideoMamba but not VideoMambaPro (n_{01}). The test is calculated as $\chi^2 = \frac{(n_{01}-n_{10})^2}{(n_{01}+n_{10})}$. The resulting value of 808.2 corresponds to a significance level of $p < 0.001$. We can thus conclude that VideoMambaPro-M is statistically significantly better than VideoMamba-M. Because we relied on the aggregated performance reported in papers for other methods, we cannot report statistical comparisons here.

5.5. Computation cost analysis

Finally, we compare the performance of VideoMambaPro with various model sizes with other approaches on K400. We map the top-1 to the number of parameters and FLOPs in Figures 3 and 4, respectively. VideoMambaPro performs favorably in terms of the accuracy-compute trade-off. Importantly, VideoMambaPro was trained on much less data than other models, which might provide opportunities for further accuracy gains without additional compute and memory requirements.

6. Conclusion

From a mathematical comparison with self-attention, we have identified two limitations in how Mamba processes token sequences. We argue that these limitations constrain Mamba's potential, especially in video understanding tasks. To address the two limitations, we have introduced VideoMambaPro (VMP). It takes VideoMamba and introduces the masked backward State Space Model (SSM), and adds residual connections in both forward and backward SSM. In experiments on Kinetics-400, Something-Something V2, HMDB51, UCF-101, and AVA V2.2, VideoMambaPro consistently demonstrates improved performance over the vanilla VideoMamba. We expect that extensive pre-training further elevates the performance of Mamba models for video tasks, making it an increasingly attractive, efficient alternative to large transformer models.

References

[1] Anurag Arnab, Mostafa Dehghani, Georg Heigold, Chen Sun, Mario Lučić, and Cordelia Schmid. Vivit: A video vision transformer. In *ICCV*, pages 6836–6846, 2021. 6

[2] Gedas Bertasius, Heng Wang, and Lorenzo Torresani. Is space-time attention all you need for video understanding? In *ICML*, pages 813–824, 2021. 6

[3] Joao Carreira and Andrew Zisserman. Quo vadis, action recognition? A new model and the Kinetics dataset. In *CVPR*, pages 6299–6308, 2017. 5

[4] Ali Diba, Vivek Sharma, Reza Safdari, Dariush Lotfi, Saquib Sarfraz, Rainer Stiefelhagen, and Luc Van Gool. Vi2CLR: Video and image for visual contrastive learning of representation. In *CVPR*, pages 1502–1512, 2021. 7

[5] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *ICLR*, 2021. 1

[6] Haoqi Fan, Bo Xiong, Karttikeya Mangalam, Yanghao Li, Zhicheng Yan, Jitendra Malik, and Christoph Feichtenhofer. Multiscale vision transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 6824–6835, 2021. 6

[7] Yuxin Fang, Bencheng Liao, Xinggang Wang, Jiemin Fang, Jiyang Qi, Rui Wu, Jianwei Niu, and Wenyu Liu. You only look at one sequence: Rethinking transformer in vision through object detection. *NeurIPS*, 34:26183–26197, 2021. 2

[8] Christoph Feichtenhofer, Haoqi Fan, Jitendra Malik, and Kaiming He. Slowfast networks for video recognition. In *ICCV*, pages 6202–6211, 2019. 7

[9] Christoph Feichtenhofer, Haoqi Fan, Yanghao Li, and Kaiming He. Masked autoencoders as spatiotemporal learners. *NeurIPS*, 35:35946–35958, 2022. 2, 7

[10] Raghav Goyal, Samira Ebrahimi Kahou, Vincent Michalski, Joanna Materzynska, Susanne Westphal, Heuna Kim, Valentin Haenel, Ingo Fruend, Peter Yianilos, Moritz Mueller-Freitag, et al. The "Something Something" video database for learning and evaluating visual common sense. In *ICCV*, pages 5842–5850, 2017. 5

[11] Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. *arXiv preprint arXiv:2312.00752*, 2023. 1, 2

[12] Albert Gu, Tri Dao, Stefano Ermon, Atri Rudra, and Christopher Ré. Hippo: Recurrent memory with optimal polynomial projections. *NeurIPS*, 33:1474–1487, 2020. 2

[13] Albert Gu, Karan Goel, and Christopher Ré. Efficiently modeling long sequences with structured state spaces. *arXiv preprint arXiv:2111.00396*, 2021. 1, 2, 3

[14] Chunhui Gu, Chen Sun, David A Ross, Carl Vondrick, Caroline Pantofaru, Yeqing Li, Sudheendra Vijayanarasimhan, George Toderici, Susanna Ricco, Rahul Sukthankar, et al. AVA: A video dataset of spatio-temporally localized atomic visual actions. In *CVPR*, pages 6047–6056, 2018. 6

[15] Tengda Han, Weidi Xie, and Andrew Zisserman. Memory-augmented dense predictive coding for video representation learning. In *ECCV*, pages 312–329, 2020. 7

[16] Tengda Han, Weidi Xie, and Andrew Zisserman. Self-supervised co-training for video representation learning. *Advances in neural information processing systems*, 33:5679–5690, 2020. 7

[17] Feyza Duman Keles, Pruthuvi Mahesakya Wijewardena, and Chinmay Hegde. On the computational complexity of self-attention. In *International Conference on Algorithmic Learning Theory*, pages 597–619. PMLR, 2023. 1

[18] Hildegard Kuehne, Hueihan Jhuang, Estíbaliz Garrote, Tomaso Poggio, and Thomas Serre. HMDB: a large video database for human motion recognition. In *ICCV*, pages 2556–2563, 2011. 6

[19] Kunchang Li, Yali Wang, Yinan He, Yizhuo Li, Yi Wang, Limin Wang, and Yu Qiao. UniformerV2: Spatiotemporal learning by arming image vits with video uniformer. *arXiv preprint arXiv:2211.09552*, 2022. 1, 7

[20] Kunchang Li, Yali Wang, Gao Peng, Guanglu Song, Yu Liu, Hongsheng Li, and Yu Qiao. Uniformer: Unified transformer for efficient spatial-temporal representation learning. In *ICLR*, 2022. 6

[21] Kunchang Li, Yali Wang, Yizhuo Li, Yi Wang, Yinan He, Limin Wang, and Yu Qiao. Unmasked teacher: Towards training-efficient video foundation models. In *ICCV*, pages 19948–19960, 2023. 2

[22] Kunchang Li, Xinhao Li, Yi Wang, Yinan He, Yali Wang, Limin Wang, and Yu Qiao. VideoMamba: State space model for efficient video understanding. *arXiv preprint arXiv:2403.06977*, 2024. 1, 2, 5

[23] Shufan Li, Harkanwar Singh, and Aditya Grover. Mamba-nd: Selective state space modeling for multi-dimensional data. In *European Conference on Computer Vision*, pages 75–92. Springer, 2024. 2

[24] Wenrui Li, Xiaopeng Hong, and Xiaopeng Fan. Spikemba: Multi-modal spiking saliency mamba for temporal video grounding. *arXiv preprint arXiv:2404.01174*, 2024. 5

[25] Yanghao Li, Chao-Yuan Wu, Haoqi Fan, Karttikeya Mangalam, Bo Xiong, Jitendra Malik, and Christoph Feichtenhofer. MViTv2: Improved multiscale vision transformers for classification and detection. In *CVPR*, pages 4804–4814, 2022. 6, 7

[26] Qin Liu, Zhenlin Xu, Gedas Bertasius, and Marc Niethammer. SimpleClick: Interactive image segmentation with simple vision transformers. In *ICCV*, pages 22290–22300, 2023. 2

[27] 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. 2, 4

[28] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017. 6

[29] Hui Lu, Hu Jian, Ronald Poppe, and Albert Ali Salah. Enhancing video transformers for action understanding with vlm-aided training. *arXiv preprint arXiv:2403.16128*, 2024. 1

[30] Jiachen Lu, Jinghan Yao, Junge Zhang, Xiatian Zhu, Hang Xu, Weiguo Gao, Chunjing Xu, Tao Xiang, and Li Zhang. Soft: Softmax-free transformer with linear complexity. *NeurIPS*, 34:21297–21309, 2021. 2

[31] Harsh Mehta, Ankit Gupta, Ashok Cutkosky, and Behnam Neyshabur. Long range language modeling via gated state spaces. *arXiv preprint arXiv:2206.13947*, 2022. 1

[32] Bolin Ni, Houwen Peng, Minghao Chen, Songyang Zhang, Gaofeng Meng, Jianlong Fu, Shiming Xiang, and Haibin Ling. Expanding language-image pretrained models for general video recognition. In *ECCV*, pages 1–18, 2022. 6

[33] Tian Pan, Yibing Song, Tianyu Yang, Wenhao Jiang, and Wei Liu. VideoMoCo: Contrastive video representation learning with temporally adversarial examples. In *CVPR*, pages 11205–11214, 2021. 7

[34] Jinyoung Park, Hee-Seon Kim, Kangwook Ko, Minbeom Kim, and Changick Kim. Videomamba: Spatio-temporal selective state space model. In *European Conference on Computer Vision*, pages 1–18. Springer, 2024. 2

[35] Mandela Patrick, Yuki Markus Asano, Ruth Fong, João F. Henriques, Geoffrey Zweig, and Andrea Vedaldi. Multi-modal self-supervision from generalized data transformations. *arXiv preprint arXiv:2003.04298*, 2020. 7

[36] Mandela Patrick, Dylan Campbell, Yuki Asano, Ishan Misra, Florian Metze, Christoph Feichtenhofer, Andrea Vedaldi, and Joao F Henriques. Keeping your eye on the ball: Trajectory attention in video transformers. *Advances in neural information processing systems*, 34:12493–12506, 2021. 6

[37] Badri Narayana Patro and Vijay Srinivas Agneeswaran. Mamba-360: Survey of state space models as transformer alternative for long sequence modelling: Methods, applications, and challenges. *arXiv preprint arXiv:2404.16112*, 2024. 1

[38] Bo Peng, Eric Alcaide, Quentin Anthony, Alon Albalak, Samuel Arcadinho, Huanqi Cao, Xin Cheng, Michael Chung, Matteo Grella, Kranthi Kiran GV, et al. RWKV: Reinventing RNNs for the transformer era. *arXiv preprint arXiv:2305.13048*, 2023. 1, 2

[39] AJ Piergiovanni, Weicheng Kuo, and Anelia Angelova. Rethinking video ViTs: Sparse video tubes for joint image and video learning. In *CVPR*, pages 2214–2224, 2023. 6

[40] Hongyu Ren, Hanjun Dai, Zihang Dai, Mengjiao Yang, Jure Leskovec, Dale Schuurmans, and Bo Dai. Combiner: Full attention transformer with sparse computation cost. *NeurIPS*, 34:22470–22482, 2021. 2

[41] Chaitanya Ryali, Yuan-Ting Hu, Daniel Bolya, Chen Wei, Haoqi Fan, Po-Yao Huang, Vaibhav Aggarwal, Arkabandhu Chowdhury, Omid Poursaeed, Judy Hoffman, et al. Hiera: A hierarchical vision transformer without the bells-and-whistles. *arXiv preprint arXiv:2306.00989*, 2023. 1

[42] Gilad Sharir, Asaf Noy, and Lihi Zelnik-Manor. An image is worth 16x16 words, what is a video worth? *arXiv preprint arXiv:2103.13915*, 2021. 6

[43] Yuanyuan Shen, Edmund M-K Lai, and Mahsa Mohaghegh. Effects of similarity score functions in attention mechanisms on the performance of neural question answering systems. *Neural Processing Letters*, 54(3):2283–2302, 2022. 2

[44] Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah. UCF101: A dataset of 101 human actions classes from videos in the wild. *arXiv preprint arXiv:1212.0402*, 2012. 6

[45] Robin Strudel, Ricardo Garcia, Ivan Laptev, and Cordelia Schmid. Segmenter: Transformer for semantic segmentation. In *ICCV*, pages 7262–7272, 2021. 2

[46] Yutao Sun, Li Dong, Shaohan Huang, Shuming Ma, Yuqing Xia, Jilong Xue, Jianyong Wang, and Furu Wei. Retentive network: A successor to transformer for large language models. *arXiv preprint arXiv:2307.08621*, 2023. 1, 2

[47] Zhan Tong, Yibing Song, Jue Wang, and Limin Wang. VideoMAE: Masked autoencoders are data-efficient learners for self-supervised video pre-training. *NeurIPS*, 35:10078–10093, 2022. 1, 2, 6, 7

[48] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. In *ICML*, pages 10347–10357. PMLR, 2021. 6

[49] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *NeurIPS*, 30, 2017. 1, 2

[50] Limin Wang, Bingkun Huang, Zhiyu Zhao, Zhan Tong, Yinan He, Yi Wang, Yali Wang, and Yu Qiao. VideoMAE V2: Scaling video masked autoencoders with dual masking. In *CVPR*, pages 14549–14560, 2023. 1, 6, 7

[51] Yi Wang, Kunchang Li, Yizhuo Li, Yinan He, Bingkun Huang, Zhiyu Zhao, Hongjie Zhang, Jilan Xu, Yi Liu, Zun Wang, et al. InternVideo: General video foundation models via generative and discriminative learning. *arXiv preprint arXiv:2212.03191*, 2022. 1, 6

[52] Yi Wang, Kunchang Li, Xinhao Li, Jiashuo Yu, Yinan He, Guo Chen, Baoqi Pei, Rongkun Zheng, Jilan Xu, Zun Wang, et al. InternVideo2: Scaling video foundation models for multimodal video understanding. *arXiv preprint arXiv:2403.15377*, 2024. 6

[53] Chen Wei, Haoqi Fan, Saining Xie, Chao-Yuan Wu, Alan Yuille, and Christoph Feichtenhofer. Masked feature prediction for self-supervised visual pre-training. In *CVPR*, pages 14668–14678, 2022. 2, 6

[54] Chao-Yuan Wu and Philipp Krahenbuhl. Towards long-form video understanding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1884–1894, 2021. 7

[55] Shen Yan, Xuehan Xiong, Anurag Arnab, Zhichao Lu, Mi Zhang, Chen Sun, and Cordelia Schmid. Multiview transformers for video recognition. In *CVPR*, pages 3333–3343, 2022. 6

[56] Hanwei Zhang, Ying Zhu, Dan Wang, Lijun Zhang, Tianxiang Chen, and Zi Ye. A survey on Visual Mamba. *arXiv preprint arXiv:2404.15956*, 2024. 5

[57] Zheng Zhang and Kil To Chong. Comparison between first-order hold with zero-order hold in discretization of input-delay nonlinear systems. In *2007 International Conference on Control, Automation and Systems*, pages 2892–2896, 2007. 3

ICCV
#8505

[58] Zixiao Zhang, Xiaoqiang Lu, Guojin Cao, Yuting Yang, Licheng Jiao, and Fang Liu. ViT-YOLO: Transformer-based YOLO for object detection. In *ICCV*, pages 2799–2808, 2021. 2

[59] Chen Zhu, Wei Ping, Chaowei Xiao, Mohammad Shoeybi, Tom Goldstein, Anima Anandkumar, and Bryan Catanzaro. Long-short transformer: Efficient transformers for language and vision. *NeurIPS*, 34:17723–17736, 2021. 2

[60] Lei Zhu, Xinjiang Wang, Zhanghan Ke, Wayne Zhang, and Rynson WH Lau. BiFormer: Vision transformer with bi-level routing attention. In *CVPR*, pages 10323–10333, 2023. 2

[61] 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. 1, 2, 4

Snakes and Ladders: Two Steps Up for VideoMamba Supplementary Material

We provide the architectures for VideoMambaPro models in Section 1. A comparison between the architectures of VideoMamba and VideoMambaPro appears in Section 2. Training details are presented in Section 3. Finally, we report ImageNet-1K image classification results in Section 4, to illustrate the potental of the proposed solutions for image classification tasks.

1. VideoMambaPro architectures

We present the architecture details of VideoMambaPro-Tiny (Ti), -Small (S), and -Middle (M) in Tables 1–3. The differences are in the embedding dimension (192, 384, 576) and the number of SSM blocks (24, 24, 32).

Stage	Tiny
Patch Embedding	nn.Conv3d (kernel size = $16 \times 16 \times 1$, **embedding dimension = 192**)
SSM	$\begin{bmatrix} \text{MLP}(768) \\ \text{MLP}(3072) \\ \text{MHA (head = 12)} \end{bmatrix} \times$ **24**
Projection	Layer Normalization Dropout (ratio) Linear layer (1000) Softmax

Table 1. Architecture details of VideoMambaPro-Ti.

Stage	Small
Patch Embedding	nn.Conv3d (kernel size = $16 \times 16 \times 1$, **embedding dimension = 384**)
SSM	$\begin{bmatrix} \text{MLP}(768) \\ \text{MLP}(3072) \\ \text{MHA (head = 12)} \end{bmatrix} \times$ **24**
Projection	Layer Normalization Dropout (ratio) Linear layer (1000) Softmax

Table 2. Architecture details of VideoMambaPro-S.

2. Architecture comparison with VideoMamba

We compare the architectures of VideoMambaPro and VideoMamba [1] in Figure 1. VideoMambaPro does not have the linear layer to generate parameters z. Additionally, our residual SSM and mask scheme do not introduce additional parameters or computational overhead, so our method has slightly fewer parameters and FLOPs.

Stage	Middle
Patch Embedding	nn.Conv3d (kernel size = $16 \times 16 \times 1$, **embedding dimension = 576**)
SSM	$\begin{bmatrix} \text{MLP}(768) \\ \text{MLP}(3072) \\ \text{MHA (head = 12)} \end{bmatrix} \times$ **32**
Projection	Layer Normalization Dropout (ratio) Linear layer (1000) Softmax

Table 3. Architecture details of VideoMambaPro-M.

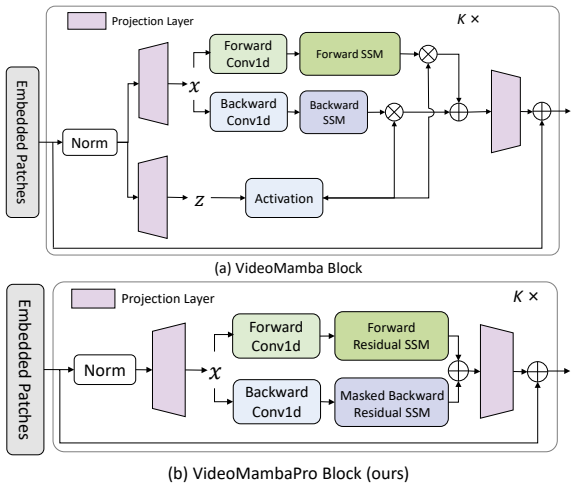

Figure 1. Comparison between the bi-directional VideoMamba (top) and VideoMambaPro (bottom) blocks.

3. Implementation details

We conduct pre-training on ImageNet-1K and fine-tuning on the Something-Something V2 and Kinetics-400 datasets with 16 NVIDIA A100-80G GPUs. Models for UCF101

and HMDB51 are trained with 8 A100-80G GPUs. The experiments on AVA V2.2 are conducted with 32 A100-80G GPUs. The values of the hyperparameters are largely similar to those used in VideoMamba [1]. We linearly scale the base learning rate with respect to the overall batch size, $lr = lr_{base} \times batchsize/256$. The pre-training details are shown in Table 4, and the fine-tuning details on the other datasets are listed in Tables 5–8.

config	image size: 224 × 224
optimizer	AdamW
base learning rate	1.5e-4
weight decay	0.1 (Tiny), 0.05 (Small, Middle)
minimal learning rate	1.0e-6
optimizer momentum	β_1, β_2 = 0.9, 0.95
batch size	512
learning rate schedule	cosine decay
warmup epochs	5 (Tiny), 10 (Small), 40 (Middle)
dropout ratio	0 (Tiny), 0.15 (Small), 0.5 (Middle)
augmentation	MultiScaleCrop
label smoothing	0.1

Table 4. Pre-training setting on ImageNet-1K

config	image size: 224 × 224
optimizer	AdamW
base learning rate	1.5e-4
weight decay	0.1 (Tiny), 0.05 (Small, Middle)
minimal learning rate	1.0e-6
optimizer momentum	β_1, β_2 = 0.9, 0.99
batch size	256
learning rate schedule	cosine decay
warmup epochs	5 (Tiny), 5 (Small) 10 (Middle)
dropout ratio	0.1 (Tiny), 0.35 (Small), 0.6 (Middle)
augmentation	RandAug (7, 0.25) (Tiny), RandAug (9, 0.5) (Small, Middle)
label smoothing	0.1
flip augmentation	yes

Table 5. Fine-tuning setting for Kinetics-400

config	image size: 224 × 224
optimizer	AdamW
base learning rate	4e-4
weight decay	0.1 (Tiny), 0.05 (Small, Middle)
minimal learning rate	1.0e-6
optimizer momentum	β_1, β_2 = 0.9, 0.999
batch size	256
learning rate schedule	cosine decay
warmup epochs	5 (Tiny), 5 (Small) 10 (Middle)
dropout ratio	0.1 (Tiny), 0.35 (Small), 0.6 (Middle)
augmentation	RandAug (7, 0.25) (Tiny), RandAug (9, 0.5) (Small, Middle)
label smoothing	0.1
flip augmentation	no

Table 6. Fine-tuning setting for Something-Something V2

4. Results on ImageNet-1K

We argue that the proposed solutions in handling the feature extraction capabilities of Mamba models are most effective when relevant tokens are more sparsely distributed in the input. While our main focus is on the video domain, we also summary our experiments on image classification. We pre-train VideoMambaPro on ImageNet-1K, which contains 1.28M training images and 50K validation images across 1,000 categories. All models are trained on the training set, and top-1 accuracy on the validation set is reported. For fair comparison, we adopt the same method as VideoMamba, and our training settings primarily follows DeiT [2]. When training on 224^2 input images, we use AdamW with a momentum of 0.9 and a total batch size of 512. Training is performed on 8 A800 GPUs, with more details provided in Table 4. The results are summarized in Table 9. VideoMambaPro achieves accuracy gains of 0.9-2.0% over VideoMamba.

config	image size: 224 × 224
optimizer	AdamW
base learning rate	4e-4
weight decay	0.1 (Tiny), 0.05 (Small, Middle)
minimal learning rate	1.0e-6
optimizer momentum	β_1, β_2 = 0.9, 0.99
batch size	128
learning rate schedule	cosine decay
warmup epochs	5 (Tiny), 5 (Small) 10 (Middle)
dropout ratio	0.1 (Tiny), 0.35 (Small), 0.6 (Middle)
augmentation	RandAug (7, 0.25) (Tiny), RandAug (9, 0.5) (Small, Middle)
label smoothing	0.1
flip augmentation	yes

Table 7. Fine-tuning setting for UCF101/HMDB51

config	image size: 224 × 224
optimizer	AdamW
base learning rate	1.5e-3 (Tiny), 2.5e-4 (Small, Middle)
weight decay	0.051 (Tiny, Small, Middle)
minimal learning rate	1.0e-6
optimizer momentum	β_1, β_2 = 0.9, 0.999
batch size	128
learning rate schedule	cosine decay
warmup epochs	5 (Tiny), 5 (Small) 10 (Middle)
dropout ratio	0.1 (Tiny), 0.35 (Small) 0.6 (Middle)
augmentation	RandAug (7, 0.25) (Tiny), RandAug (9, 0.5) (Small, Middle)
label smoothing	0.1
flip augmentation	yes

Table 8. Fine-tuning setting for AVA 2.2

	Input	Param	FLOPs	Top-1
VideoMamba (Ti)	224^2	7M	1.1G	76.9
VideoMambaPro (Ti)	224^2	7M	1.1G	78.9
VideoMamba (S)	224^2	26M	4.3G	81.2
VideoMambaPro (S)	224^2	25M	4.2G	82.4
VideoMamba (M)	224^2	74M	12.7G	82.8
VideoMambaPro (M)	224^2	72M	12.4G	83.8
VideoMamba (M)	448^2	75M	50.4G	83.8
VideoMambaPro (M)	448^2	73M	49.6G	84.7

Table 9. ImageNet-1K pre-training results for VideoMamba and VideoMambaPro.

ICCV
#8505

References

[1] Kunchang Li, Xinhao Li, Yi Wang, Yinan He, Yali Wang, Limin Wang, and Yu Qiao. VideoMamba: State space model for efficient video understanding. *arXiv preprint arXiv:2403.06977*, 2024. 1, 2

[2] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. In *ICML*, pages 10347–10357. PMLR, 2021. 2