PKS^4: PARALLEL KINEMATIC SELECTIVE STATE SPACE SCANNERS FOR EFFICIENT VIDEO UNDERSTANDING

Anonymous authors
Paper under double-blind review

ABSTRACT

Traditional video models relying on dense spatiotemporal attention suffer from quadratic computational costs. To circumvent these costs, recent approaches adapt image models for videos via Parameter-Efficient Fine-Tuning (PEFT) methods such as adapters. However, deeply inserting these modules incurs prohibitive activation memory overhead during back-propagation. Video state-space models (SSMs) with global scanning provide linear complexity but flatten video tokens into a one-dimensional sequence, mixing spatial and temporal ordering.

To address these limitations, we propose ***P**arallel **K**inematic **S**elective **S**tate **S**pace **S**canners* (PKS^4). We retain a 2D vision backbone for spatial semantics and insert a single plug-and-play PKS^4 module with linear-complexity temporal scanning, bypassing the need for temporal attention or multi-layer adapters. We first explicitly extract kinematic priors via a Kinematic Prior Encoder, which captures inter-frame correspondences and feature changes. These priors subsequently drive SSMs to model temporal dependencies, adaptively modulating the update speeds and read-write strategies based on the input content at each time step.

Instead of global scanning, we deploy parallel scanners along the temporal dimension for each spatial location, preserving spatial structures while limiting computational overhead. Extensive experiments on spatial-heavy and temporal-heavy action recognition benchmarks demonstrate that PKS^4 reaches 74.3% Top-1 accuracy on SSV2 and 84.1% on K400. The ViT-B/16 configuration fine-tunes for 20 epochs and uses approximately $10.7\times$ less video-domain training compute than the compared VideoMamba-M$_{800e}$ configuration.

1 INTRODUCTION

Video understanding requires modeling long-range temporal dependencies across frames, which becomes challenging for longer clips. Existing works extend the Vision Transformer (ViT) from images to videos using spatiotemporal attention (Bertasius et al., 2021; Arnab et al., 2021). Global attention has quadratic complexity with respect to the spatiotemporal sequence length, incurring high computational costs. To alleviate this issue, another line of research adapts image backbones to videos through Parameter-Efficient Fine-Tuning (PEFT), such as inserting lightweight temporal adapters into pretrained Vision Transformers (Pan et al., 2022; Yang et al., 2023). Though adapter-based methods reduce the number of trainable parameters, deeply inserting these adapters requires storing massive intermediate activations across the entire backbone during back-propagation, which results in substantial activation memory overhead (Yao et al., 2023). As illustrated in Fig. 2(a), this practical memory burden limits the accuracy-memory trade-off of adapter-based transfer. More recently, State Space Models (SSMs), particularly selective SSMs (Gu & Dao, 2024), have emerged as an attractive alternative. State space models formulate continuous-time dynamics as discrete state updates (Gu et al., 2021), while selective SSMs support linear-time sequence modeling (Gu & Dao, 2024).

Despite the success of SSMs in natural language processing, the naive application of pure SSMs to video understanding leaves two critical issues unresolved. **(i)** Videos are inherently 3D structures comprising spatial and temporal dimensions, yet a common strategy is to flatten the video into a 1D

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

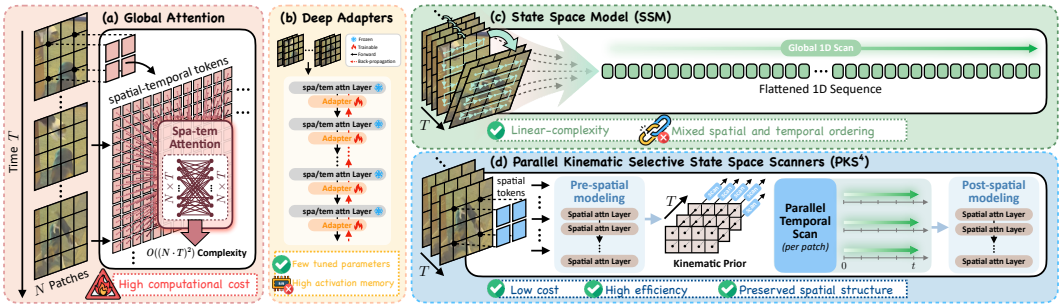

Figure 1: Overview of the motivation and design of PKS^4. (a) Global spatiotemporal attention has quadratic complexity in the number of video tokens. (b) Deeply inserted adapters require intermediate activations throughout the backbone during back-propagation. (c) A global 1D state-space scan serializes the spatiotemporal token grid and can obscure native 2D spatial relationships. (d) PKS^4 retains a 2D spatial backbone and inserts parallel temporal scanners driven by kinematic priors.

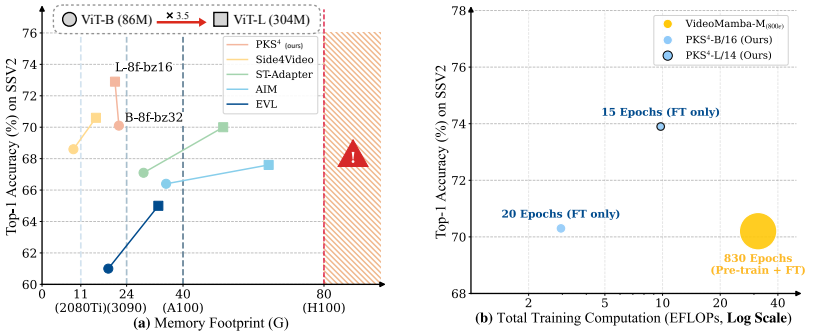

Figure 2: (a) Training-memory and accuracy comparison with PEFT methods under the stated configurations. (b) Video-domain training epochs and computation after the common image or CLIP initialization.

token sequence and subsequently perform continuous scanning. This treatment mixes spatial and temporal ordering into a single sequence. Joint modeling must account for causal temporal cues and non-causal spatial information (Yang et al., 2024). As temporal dependencies are intrinsically tied to spatial variations (*e.g.*, object shifts, boundary changes, and detail transformations), this serialization mixes information about *what* changes and *where* these changes originate. The VideoMamba-M$_{800e}$ configuration in our comparison uses an 800-epoch video pre-training schedule (Li et al., 2024). As illustrated in Fig. 2(b), this additional stage increases the video-specific training compute beyond the CLIP initialization. **(ii)** Furthermore, the content driving state transitions is often generic token content rather than motion-aware kinematic state content. These issues highlight the demand for an SSM-integrated video architecture that is not only scan-efficient but also capable of determining the appropriate content to drive state transitions.

To this end, we propose ***P**arallel **K**inematic **S**elective **S**tate **S**pace **S**canners* (PKS^4). Instead of replacing the spatial backbone or densely injecting temporal adapters across multiple layers, we retain a 2D image-pretrained ViT as the primary backbone for spatial representation learning and insert a *single* plug-and-play temporal module at an intermediate layer. Before performing selective state-space scanning, we explicitly enrich the input content with motion-aware kinematic priors. Concretely, given intermediate tokens from a ViT block, we reshape them back to spatial layout and process them through a Kinematic Prior Encoder (KPE), as illustrated in Fig. 3. This encoder applies a correlation operator and a variation operator sequentially to capture two complementary forms of motion patterns: (i) inter-frame correspondences reflecting local displacement correlations, and (ii) inter-frame changes reflecting temporal feature variations. Such explicit kinematic priors provide a more appropriate state context for SSMs, encouraging the scanner to model motion-driven state transitions rather than directly processing raw video tokens.

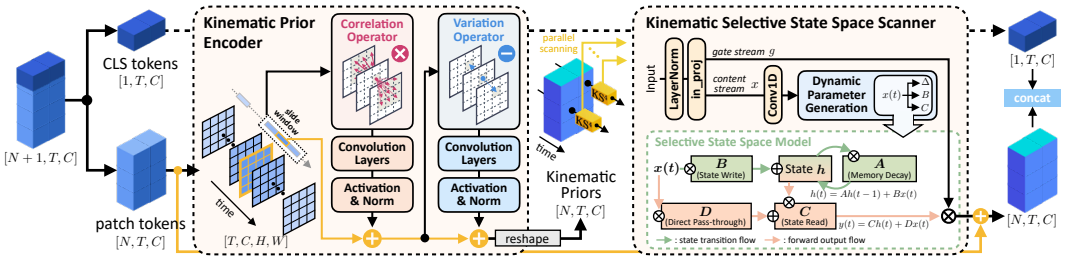

Figure 3: Overview of **PKS**^{4}. Given intermediate token features from a ViT layer, we first split class and patch tokens. The Kinematic Prior Encoder processes patch tokens through a temporal sliding window. The correlation and variation operators are sequentially applied to extract kinematic priors. To preserve the spatial token layout, we deploy parallel scanning along the temporal dimension for each spatial location. Each Kinematic Selective State Space Scanner (**KS**^{4}) dynamically generates data-dependent parameters to adaptively modulate the update and decay processes, achieving linear-complexity temporal modeling. The updated tokens are concatenated and fed back to the next ViT layer.

Building upon the kinematic priors, PKS^4 performs temporal modeling via parallel patch-wise scanning. Instead of flattening all tokens into a 1D sequence, we construct an individual temporal sequence for each spatial location and scan these sequences in parallel along the temporal dimension. This design preserves the 2D spatial structure inherited from the image backbone while maintaining linear complexity for temporal modeling. The parallel scanning mechanism dynamically generates parameters to adaptively modulate update speeds and read-write strategies based on the input content. The updated tokens are then reshaped and fed into the subsequent ViT layer, rendering the module compatible with the original backbone. Consequently, PKS^4 achieves linear-complexity temporal modeling while preserving the spatial structure of the video, providing a route for transferring image backbones to video understanding.

Extensive experiments on temporal-heavy and spatial-heavy benchmarks validate the effectiveness of the design. More importantly, these results support the broader perspective that efficient temporal modeling does not necessitate a strict choice between dense attention architectures and SSMs. Given an appropriate temporal interface, SSMs can be made more compatible with the structure of the ViT.

Our contributions are summarized as follows:

- We propose Parallel Kinematic Selective State Space Scanners (PKS^4), a plug-and-play module that achieves linear-complexity temporal modeling by combining kinematic priors with SSMs.
- We introduce a Kinematic Prior Encoder to explicitly extract motion-aware kinematic priors, providing an appropriate task-specific state context for SSMs.
- We decouple the spatial and temporal dimensions by deploying parallel scanners that preserve the spatial structure inherited from pre-trained backbones.
- Experiments demonstrate 74.3% Top-1 accuracy on temporal-heavy SSV2 and competitive performance on appearance-dominant K400. The ViT-B/16 configuration fine-tunes for 20 epochs and uses approximately 10.7× less video-domain training compute than the compared VideoMamba-M$_{800e}$ configuration.

2 Method

2.1 Overview

Spatial encoding. Given an input video clip $\mathbf{X} \in \mathbb{R}^{B\times T\times 3\times H\times W}$, where B denotes the batch size and T denotes the number of frames, we first reshape the video into $B\cdot T$ images of resolution $H \times W$ (height H and width W). We then feed them into an image-pretrained ViT as the primary spatial encoder. Let the intermediate features at the l-th layer be $\mathbf{Z}^{(l)} \in \mathbb{R}^{B\times T\times (N+1)\times C}$, where N is the number of spatial patch tokens and C is the channel dimension. As shown in Fig. 3, we

separate $\mathbf{Z}^{(l)}$ into a class (CLS) token and patch tokens: $\mathbf{z}_{\text{cls}}^{(l)} \in \mathbb{R}^{B\times T\times C}, \quad \mathbf{Z}_{\text{p}}^{(l)} \in \mathbb{R}^{B\times T\times N\times C}$. For spatial semantics extraction, we enhance the patch tokens with kinematic priors via the KPE module: $\widehat{\mathbf{Z}}_{\text{p}}^{(l)} = \text{KPE}\big(\mathbf{Z}_{\text{p}}^{(l)}\big)$.

Parallel temporal scanning. Given the kinematic-prior-enhanced patch tokens $\widehat{\mathbf{Z}}_{\text{p}}^{(l)}$, we form a temporal sequence for each spatial location. We then apply selective state space scanning only along the temporal axis. For the n-th spatial location, we define a patch-wise temporal sequence as:

$$\mathbf{s}_n = \Big[\widehat{\mathbf{Z}}_{\text{p}}^{(l)}(:,1,n,:), \ldots, \widehat{\mathbf{Z}}_{\text{p}}^{(l)}(:,T,n,:)\Big] \in \mathbb{R}^{B\times T\times C}. \tag{1}$$

All sequences are then processed in parallel with a weight-shared Kinematic Selective State Space Scanner (KS^4):

$$\widetilde{\mathbf{s}}_n = \text{KS}^4(\mathbf{s}_n), \qquad n = 1, \ldots, N. \tag{2}$$

Finally, the scanned sequences are reassembled according to the spatial indices and added back to the patch tokens,

$$\widetilde{\mathbf{Z}}_{\text{p}}^{(l)} = \mathbf{Z}_{\text{p}}^{(l)} + \text{Reassemble}\Big(\{\widetilde{\mathbf{s}}_n\}_{n=1}^{N}\Big) \in \mathbb{R}^{B\times T\times N\times C}. \tag{3}$$

Auxiliary CLS temporal route. In addition, we maintain a lightweight temporal route for the CLS token across backbone layers. Since the CLS token aggregates global semantics, a small temporal operator allows information exchange across frames. The CLS token is processed by a depthwise 1D convolution with kernel size 3 along time: $\widetilde{\mathbf{z}}_{\text{cls}}^{(l)} = \text{Conv1D}_{\text{cls}}^{k=3}\Big(\mathbf{z}_{\text{cls}}^{(l)}\Big)$, where the convolution is initialized in a shift-style manner, enabling temporal propagation. The updated patch tokens are then concatenated with the temporally enhanced CLS tokens and fed back to the next layer, without modifying the internal structure of the original ViT block.

Video-level prediction. After the final ViT layer, we obtain frame-wise representations $\mathbf{v}_1, \ldots, \mathbf{v}_T \in \mathbb{R}^C$. We aggregate them through temporal average pooling: $\mathbf{v} = \frac{1}{T}\sum_{t=1}^{T}\mathbf{v}_t$, and feed the pooled feature into a linear classifier: $\mathbf{p} = \mathbf{W}_{\text{cls}}\mathbf{v} + \mathbf{b}_{\text{cls}}$. Here, K denotes the number of action classes, $\mathbf{W}_{\text{cls}} \in \mathbb{R}^{K\times C}$, and $\mathbf{b}_{\text{cls}} \in \mathbb{R}^K$.

2.2 Spatial Backbone

After positional encoding, the tokens are processed by standard Transformer blocks. Let $\mathbf{Z}^{(i-1)} \in \mathbb{R}^{B\times T\times (N+1)\times C}$ be the input to the i-th block. The forward pass is formulated as:

$$\mathbf{Z}^{(i-\frac{1}{2})} = \mathbf{Z}^{(i-1)} + \text{MSA}\big(\text{LN}(\mathbf{Z}^{(i-1)})\big), \tag{4}$$

$$\mathbf{Z}^{(i)} = \mathbf{Z}^{(i-\frac{1}{2})} + \text{MLP}\big(\text{LN}(\mathbf{Z}^{(i-\frac{1}{2})})\big), \tag{5}$$

where MSA, MLP, and LN denote Multi-head Self-Attention (Dosovitskiy et al., 2020), Multi-Layer Perceptron (Dosovitskiy et al., 2020), and Layer Normalization (Ba et al., 2016), respectively.

2.3 Kinematic Prior Encoder

To provide compatible input with the kinematic nature of videos for the state transition, we introduce the Kinematic Prior Encoder (KPE) that explicitly extracts motion-aware priors. Given patch tokens $\mathbf{Z}_{\text{p}} \in \mathbb{R}^{B\times T\times N\times C}$, we first reshape them back to a spatial grid: $\mathbf{F} \in \mathbb{R}^{B\times T\times C\times H_p\times W_p}, \quad N = H_pW_p$. Here, H_p and W_p denote the height and width of the patch grid. The KPE processes the spatial feature $\mathbf{F}$ sequentially using two complementary operators:

$$\mathbf{F}^{\text{corr}} = \mathcal{O}_{\text{corr}}(\mathbf{F}), \qquad \mathbf{F}^{\text{var}} = \mathcal{O}_{\text{var}}(\mathbf{F}^{\text{corr}}), \tag{6}$$

where $\mathcal{O}_{\text{corr}}$ captures inter-frame correspondence patterns and $\mathcal{O}_{\text{var}}$ encodes temporal feature changes. The features are projected to the token space and added to the original patch tokens:

$$\widehat{\mathbf{Z}}_{\text{p}} = \mathbf{Z}_{\text{p}} + \phi(\mathbf{F}^{\text{var}}), \tag{7}$$

where $\phi(\cdot)$ denotes reshaping and projection back to $\mathbb{R}^{B\times T\times N\times C}$.

CORRELATION OPERATOR

The correlation operator is designed to model displacement consistency across frames. For a target spatial location p at time t, we compare the feature $\mathbf{f}_t(p) \in \mathbb{R}^C$ at p with features within a local search window. For a temporal offset τ and spatial displacement $\delta = (u, v)$, we define the local correlation score as:

$$\mathcal{S}_{t,\tau}(p, \delta) = \left\langle \bar{\mathbf{f}}_t(p), \bar{\mathbf{f}}_{t+\tau}(p + \delta) \right\rangle, \tag{8}$$

where $\bar{\mathbf{f}}$ denotes channel-wise normalized features and $\langle \cdot, \cdot \rangle$ is the inner product. The temporal offset τ is restricted to a local temporal window, and δ is restricted to a local spatial region. In this way, the operator encodes whether a local pattern at time t reappears nearby at time $t + \tau$, thereby extracting local motion and correspondence. All correlation responses within the temporal-spatial neighborhood are then aggregated and mapped back to the feature domain: $\mathbf{F}^{\text{corr}} = \psi_{\text{corr}}\big(\{\mathcal{S}_{t,\tau}(p, \delta)\}_{\tau,\delta}\big)$, where $\psi_{\text{corr}}(\cdot)$ denotes the learnable projection to the visual feature space.

VARIATION OPERATOR

While the correlation operator focuses on matching and displacement, the variation operator focuses on temporal change. It captures motion boundaries and appearance changes through inter-frame difference modeling. Given the intermediate feature map, we compute temporal differences within a local temporal context. For a temporal offset τ, the variation signal is defined as:

$$\Delta^{\text{corr}}_{t,\tau}(p) = \mathbf{f}^{\text{corr}}_{t+\tau}(p) - \mathbf{f}^{\text{corr}}_{t}(p). \tag{9}$$

When multiple neighboring frames are considered, we collect a set of difference responses $\{\Delta^{\text{corr}}_{t,\tau}(p)\}_\tau$ and jointly aggregate them: $\mathbf{F}^{\text{var}} = \psi_{\text{var}}\big(\{\Delta^{\text{corr}}_{t,\tau}(p)\}_\tau\big)$, where $\psi_{\text{var}}(\cdot)$ is a learnable projection. By emphasizing both *where similar content moves* and *where content changes*, KPE provides complementary kinematic priors for the subsequent scanner.

2.4 KINEMATIC SELECTIVE STATE SPACE SCANNER

After KPE, the patch tokens become motion-aware: $\widehat{\mathbf{Z}}_{\text{p}} \in \mathbb{R}^{B\times T\times N\times C}$. We then apply the Kinematic Selective State Space Scanner (KS^4) to each spatial location independently along the temporal axis. Specifically, we reshape the tokens into $\widehat{\mathbf{Z}}_{\text{seq}} \in \mathbb{R}^{(BN)\times T\times C}$ so that each row corresponds to one patch-wise temporal sequence.

State parameterization. As shown in Fig. 3, for a patch-wise temporal sequence $\mathbf{u} \in \mathbb{R}^{T\times C}$, we first apply normalization. Then, a linear projection expands the channels and splits the result into a content stream and a gate stream: $[\mathbf{x}', \mathbf{g}] = \mathbf{W}_{\text{in}}\text{LN}(\mathbf{u})$, where $\mathbf{x}'$ is the content branch and $\mathbf{g}$ is the gating branch. Temporal mixing is then applied to the content embedding: $\mathbf{x} = \text{Conv1D}(\mathbf{x}')$. Following selective SSMs, the dynamic parameters are generated from the input itself:

$$[\boldsymbol{\delta}, \mathbf{B}, \mathbf{C}] = \Gamma(\mathbf{x}), \tag{10}$$

where $\Gamma(\cdot)$ denotes a learnable projection. Here, $\boldsymbol{\delta}$ controls the input-dependent step size, while $\mathbf{B}$ and $\mathbf{C}$ parameterize the input-dependent write and read operations. The step size is constrained to be positive via: $\Delta_t = \text{softplus}(\delta_t + b_\delta)$. Here, b_δ denotes a bias.

Selective state transition. Let $\mathbf{h}_t \in \mathbb{R}^{d_s}$ denote the hidden state at time step t, where d_s is the state dimension. The selective state transition follows the standard SSM form:

$$\mathbf{h}_t = \mathbf{A}(\Delta_t)\mathbf{h}_{t-1} + \mathbf{B}_t\mathbf{x}_t, \tag{11}$$

$$\mathbf{y}_t = \mathbf{C}_t\mathbf{h}_t + \mathbf{D}\mathbf{x}_t, \tag{12}$$

where $\mathbf{A}(\Delta_t)$ is the discretized transition matrix controlled by the step size Δ_t, $\mathbf{B}_t$ and $\mathbf{C}_t$ are write/read parameters, and $\mathbf{D}$ is the direct skip term. The scanner is driven by motion-aware tokens produced by the KPE. The data-dependent parameters in Eqs. (10)–(12) depend on the inter-frame correlation and variation features. The final output of the scanner is obtained through output projection and residual fusion:

$$\widetilde{\mathbf{u}}_t = \mathbf{W}_{\text{out}}\big(\mathbf{y}_t \odot \sigma(\mathbf{g}_t)\big) + \mathbf{u}_t, \tag{13}$$

where $\sigma(\cdot)$ is a gating nonlinearity.

3 EXPERIMENTS

3.1 DATASETS AND EVALUATION METRICS

We evaluate PKS^4 on two public action-recognition benchmarks. **(i) Something-Something V2** (SSV2) (Goyal et al., 2017) consists of 174 fine-grained action classes that are object and motion-centric. The training, validation, and test sets contain $168,913$, $24,777$, and $27,157$ videos, respectively. **(ii) Kinetics-400** (K400) (Kay et al., 2017) contains approximately 240K training videos and 20K validation videos across 400 classes; we report validation accuracy.

Table 1: Training memory usage and performance comparison on SSV2. We report accuracy under the same test setting of $8 \times 3 \times 1$ views. Baseline memory, GFLOPs, and accuracy values are from Side4Video (Yao et al., 2023). For ViT-B/16 and ViT-L/14, the batch sizes are 32 and 16, respectively; GFLOPs correspond to the stated multi-view evaluation.

Method	Training Memory	GFLOPs	Top-1 (%)
ViT-B/16			
ST-Adapter (Pan et al., 2022)	28.8G	455	67.1
AIM (Yang et al., 2023)	35.2G	624	66.4
EVL (Lin et al., 2022)	17.9G	512	61.0
Side4Video (Yao et al., 2023)	**8.9G**	445	68.6
PKS^4 *(ours)*	21.8G	**437**	**70.1**
ViT-L/14			
ST-Adapter (Pan et al., 2022)	51.4G	2062	70.0
AIM (Yang et al., 2023)	64.3G	2877	67.6
EVL (Lin et al., 2022)	33G	2411	65.1
Side4Video (Yao et al., 2023)	**15.3G**	2092	70.6
PKS^4 *(ours)*	20.7G	**1929**	**72.9**

Table 2: Video-domain training computation on SSV2. We report EFLOPs for video pre-training and downstream fine-tuning after image or CLIP initialization; the common upstream initialization cost is excluded.

Method	VideoMamba M_{800e}	PKS^4 B/16	PKS^4 L/14
Training schedule (epochs)			
Pre-train	800	–	–
Fine-tune	30	20	15
Training compute (EFLOPs)			
Pre-train	27.7883	–	–
Fine-tune	3.7538	2.9521	9.7810
Total	31.5421	**2.9521**	9.7810
Top-1 (%)	70.2	70.3	**73.9**

3.2 IMPLEMENTATION DETAILS

We implement PKS^4 with a CLIP-pretrained ViT backbone (Radford et al., 2021). We insert the proposed module once after the 8-th transformer block. We uniformly sample 8, 16, or 32 frames with a random frame gap, and each frame is resized to 224×224. The state dimension of the selective scanner is set to $d_s = 16$. In the Kinematic Prior Encoder, we use a temporal neighborhood of 4 frames and a local spatial search window of size 9×9 for correlation modeling. We train the model using AdamW with a base learning rate of 7e-4 (1e-3 for ViT-L), weight decay of 0.05, and a cosine decay schedule with 5 warm-up epochs. We use the standard cross-entropy loss for video classification. With label smoothing, the target distribution is a smoothed version of the one-hot label distribution. The learning rate is scaled according to the global batch size, and we apply layer-wise learning-rate decay to the ViT backbone.

3.3 TRAINING COST COMPARISON

MEMORY USAGE

We compare PKS^4 with existing PEFT methods on SSV2. We compare training-memory figures using 8 frames as model input. All models are evaluated under the same $8 \times 3 \times 1$ test views (#frames $\times$ #clips $\times$ #crops), while the ViT-B/16 and ViT-L/14 variants use batch sizes of 32 and 16, respectively. As shown in Table 1, PKS^4 achieves higher accuracy than the listed adapter-based baselines while retaining a moderate measured memory footprint. The ViT-L/14 configuration uses slightly less memory than the ViT-B/16 configuration (20.7G vs. 21.8G) with batch sizes of 16 and 32, respectively.

Table 3: Additional SSV2 comparisons. Input denotes frames@resolution; views denote temporal clips × spatial crops; GFLOPs correspond to the stated input and views. "e" denotes VideoMamba video-pre-training epochs.

Model configuration	Pre-train	Input	Views	Reported params	GFLOPs	Top-1	Top-5
Spatiotemporal Attention							
TimeSformer, ViT-B/16	IN-21K	8@224	1×3	121.4M	1770	59.5	–
TimeSformer-L, ViT-B/16	IN-21K	96@224	1×3	121.4M	7140	62.3	–
ViViT-L/16×2 FE	IN-21K + K400	32@224	1×3	612M	11940	65.9	89.9
Mformer, ViT-B/16	IN-21K + K400	16@224	3×1	109.1M	1109	66.5	90.1
Mformer, ViT-L/14	IN-21K + K400	16@224	3×1	381.9M	2876.4	67.1	90.6
Video Swin-B	K400	32@224	1×3	88.8M	963	69.6	92.7
MTV-B	K400	16@224	4×3	310M	11160	67.6	90.1
Parameter-Efficient Fine-Tuning with CLIP-Pretrained ViT							
ST-Adapter, ViT-B/16	CLIP	32@224	3×1	14.1M	1955	69.5	92.6
ST-Adapter, ViT-L/14	CLIP	32@224	3×1	–	8248	72.3	93.9
AIM, ViT-B/16	CLIP	32@224	3×1	14M	2496	69.1	92.3
AIM, ViT-L/14	CLIP	32@224	3×1	50M	11508	70.6	92.7
DUALPATH, ViT-B/16	CLIP	32@224	1×3	13M	716	70.3	92.9
DUALPATH, ViT-L/14	CLIP	32@224	1×3	33M	1932	71.4	93.4
EVL, ViT-B/16	CLIP	32@224	3×1	28.8M	2824	62.4	–
EVL, ViT-L/14	CLIP	32@224	3×1	–	10418	66.7	–
DiST, ViT-B/16	CLIP	32@224	3×1	–	1950	70.9	92.1
DiST, ViT-L/14	CLIP	32@224	3×1	–	8490	73.1	93.2
M^2-CLIP, ViT-B/16	CLIP	32@224	1×3	16M	2526	69.1	–
State Space Models							
VideoMamba-S	IN-1K	8@224	3×2	26M	204	66.6	90.4
VideoMamba-S	IN-1K	16@288	3×2	26M	672	68.1	91.2
VideoMamba-M	IN-1K	8@224	3×4	74M	1212	67.3	91.0
VideoMamba-M	IN-1K	16@288	3×4	74M	3996	68.4	91.6
VideoMamba-M_{800e}	CLIP	8@224	3×2	74M	606	70.2	92.6
VideoMamba-M_{800e}	CLIP	16@288	3×2	74M	1998	71.4	92.9
PKS^4-B/16 *(ours)*	CLIP	8@224	2×3	91.7M	874	70.3	92.4
PKS^4-B/16 *(ours)*	CLIP	32@224	1×3	91.7M	1745	71.7	93.3
PKS^4-L/14 *(ours)*	CLIP	16@224	1×3	312.8M	3857.4	73.9	94.0
PKS^4-L/14 *(ours)*	CLIP	32@224	1×3	312.8M	7713.6	**74.3**	**94.3**

Video-Domain Training Computation

We compare the video-domain training computation of PKS^4 with the VideoMamba-M_{800e} configuration (Li et al., 2024). As shown in Table 2, the reported EFLOPs include video pre-training and downstream fine-tuning. VideoMamba-M_{800e} uses 800 epochs of video pre-training followed by 30 epochs of fine-tuning on SSV2, for 31.5421 EFLOPs. PKS^4-B/16 uses 2.9521 EFLOPs, approximately $10.7\times$ less under this accounting, and PKS^4-L/14 uses 9.7810 EFLOPs, approximately $3.2\times$ less.

Inference Latency

We evaluate inference latency on a single NVIDIA 4090 GPU. Table 5 reports the mean and median latency for each configuration. With ViT-L/14 and $8 \times 2 \times 3$ views, PKS^4 has a mean latency of 230.41 ms, compared with 363.23 ms for TimeSformer and 273.70 ms and 296.71 ms for the two ST-Adapter configurations. With ViT-B/16 and $8 \times 1 \times 1$ views, PKS^4 takes 13.35 ms, compared with 11.15 ms and 12.58 ms for ST-Adapter.

3.4 Comparison with State-of-the-Art

We compare PKS^4 with representative methods across three temporal modeling paradigms: spatiotemporal attention, parameter-efficient fine-tuning (PEFT), and state space models (SSMs). Table 3 and Table 4 show that PKS^4 achieves the highest Top-1 accuracy in the SSV2 comparison and

Table 4: Additional comparisons on K400. Views = #frames × #temporal clips × #spatial crops; GFLOPs correspond to the stated view settings. "e" denotes the number of pre-training epochs for VideoMamba.

Method	**Views**	**GFLOPs**	**Top-1** (%)
TimeSformer-L	96×1×3	7140	80.7
ViViT-L/16×2 FE	32×1×3	11940	81.7
Mformer-B/16	16×3×10	11085	79.7
Mformer-L/14	16×3×10	28764	81.1
Video Swin-B	32×4×3	3384	82.7
Video Swin-L	32×4×3	7248	83.1
MTV-B/16	32×4×3	4790.4	81.8
ST-Adapter-B/16	32×3×1	1821	82.7
EVL-B/16	32×3×1	2813	**84.2**
M^2-CLIP-B/16	32×4×3	10104	84.1
VideoMamba-S	32×4×3	1620	81.5
VideoMamba-M	32×4×3	4836	82.4
VideoMamba-M_{800e}	32×4×3	4836	83.9
PKS^4-B/16 *(ours)*	32×1×3	1745	82.2
PKS^4-L/14 *(ours)*	32×1×3	7713.6	84.1

Table 5: Per-sample inference latency in milliseconds on an NVIDIA 4090 GPU.

Model	Adapter	Views	Mean	Median
TimeSformer				
ViT-B/16	–	$8 \times 1 \times 1$	13.78	13.77
ViT-B/16	–	$8 \times 2 \times 3$	80.23	80.24
ViT-L/14	–	$8 \times 1 \times 1$	62.99	63.03
ViT-L/14	–	$8 \times 2 \times 3$	363.23	363.37
ST-Adapter				
ViT-B/16	12×384	$8 \times 1 \times 1$	11.15	11.14
ViT-B/16	12×384	$8 \times 2 \times 3$	63.90	63.90
ViT-B/16	24×384	$8 \times 1 \times 1$	12.58	12.56
ViT-B/16	24×384	$8 \times 2 \times 3$	70.94	70.94
ViT-L/14	24×384	$8 \times 1 \times 1$	49.58	49.57
ViT-L/14	24×384	$8 \times 2 \times 3$	273.70	273.83
ViT-L/14	48×384	$8 \times 1 \times 1$	53.57	53.56
ViT-L/14	48×384	$8 \times 2 \times 3$	296.71	296.70
PKS^4				
ViT-B/16	–	$8 \times 1 \times 1$	13.35	13.29
ViT-B/16	–	$8 \times 2 \times 3$	62.22	62.19
ViT-L/14	–	$8 \times 1 \times 1$	39.05	39.06
ViT-L/14	–	$8 \times 2 \times 3$	230.41	230.32

Table 6: Ablation of PKS^4 modules on SSV2 with ViT-B/16 and an $8 \times 2 \times 3$ view setting. "Baseline" denotes the pre-trained ViT which only fine-tunes the image encoder and performs temporal average pooling.

Method	**Params**	**Δ**	**Top-1** (%)	**Δ**
Baseline	86.34M	-	44.0	-
+ KS^4	90.11M	↑3.77M	66.1	↑22.1
+ KS^4 + KPE ($\mathcal{O}_{\text{var}}$)	90.86M	↑4.52M	69.1	↑25.1
+ KS^4 + KPE ($\mathcal{O}_{\text{var}}$ + $\mathcal{O}_{\text{corr}}$)	91.70M	↑5.36M	**70.3**	↑26.3

competitive accuracy on K400. Under the respective input and view settings, PKS^4-L/14 with 32 input frames reaches **74.3%** Top-1, 1.2 percentage points above DiST-L/14 and 2.9 points above the VideoMamba-M_{800e} setting, while using fewer reported GFLOPs than DiST-L/14 (7713.6 *vs.* 8490). The contribution of the kinematic priors is examined in the module ablation in Table 6. On K400, PKS^4 remains competitive with 84.1% Top-1 for PKS^4-L/14 at 7713.6 GFLOPs. Although it does not exceed the strongest listed result on K400, the comparison shows that PKS^4 remains competitive with a single inserted temporal module under the reported configurations.

3.5 Ablation Study

In this section, we ablate PKS^4 with different settings. We use a ViT-B/16 backbone and a test-view setting of 8×2×3 by default.

Effectiveness of each module. Table 6 evaluates the contribution of each module in PKS^4. Starting from the baseline ViT with temporal average pooling, introducing KS^4 yields a substantial improvement from 44.0% to 66.1% Top-1. Building on this, equipping KS^4 with the variation-based kinematic prior further improves accuracy to 69.1%, a gain of 3.0 percentage points over KS^4 alone. Finally, incorporating the correlation operator before the variation operator pushes the performance to 70.3%, yielding an overall gain of 26.3 points over the baseline with a parameter increase of 5.36M.

Figure 4 in Appendix A shows KPE response overlays on two video examples.

Table 7: Additional ablation studies on SSV2: (a) insertion layer under the $8 \times 1 \times 3$ view setting; (b) scan method, with total fine-tuning compute reported in EFLOPs; (c) number of KPE context frames under the $8 \times 1 \times 3$ view setting; and (d) CLS-token temporal modeling under the default $8 \times 2 \times 3$ view setting.

(a) Insert position

After l-th layer	Top-1 (%)	Top-5 (%)
5	64.69	89.06
6	66.32	90.69
7	67.92	91.43
8	**68.67**	91.74
9	68.48	**91.80**
10	68.13	91.33
11	67.66	91.13

(b) Scan method

Method	Unidirectional	Bidirectional
Params	91.7M	95.5M
Train epochs	20	20
Train EFLOPs	**2.9521**	3.0404
Top-1 (%)	70.3	**70.4**

(c) Context frames

Context frames	0	2	**4**	6
Top-1 (%)	67.14	67.82	**68.67**	68.06
Top-5 (%)	91.10	91.18	**91.74**	91.47

(d) CLS temporal module

Method	Baseline	TSM	**TokenT1D**	TokenShift
Top-1 (%)	67.43	69.44	**70.31**	68.80
Top-5 (%)	90.44	91.80	**92.36**	91.28

Effectiveness of the insertion position of PKS^4. We investigate the effect of inserting PKS^4 at different depths of the ViT backbone. As shown in Table 7(a), the insertion depth has a clear impact on recognition performance, and the better results are achieved when PKS^4 is inserted in the middle-to-late stages of the backbone. This trend suggests that applying PKS^4 too early is less favorable as the intermediate features are still insufficiently mature, while inserting the module too late also results in a degradation as the remaining layers become insufficient to propagate the temporally enhanced representations. These results demonstrate that the proposed module is most accurate in Top-1 when inserted after layer 8 among the tested positions.

Scan methods of PKS^4. As shown in Table 7(b), bidirectional scanning brings a 0.1-percentage-point gain in Top-1 accuracy over the unidirectional variant, while introducing higher parameter count and training compute. In contrast, the unidirectional design already achieves competitive performance with lower cost.

The number of context frames in KPE. We investigate the effectiveness of different numbers of context frames used in KPE. As shown in Table 7(c), introducing temporal context in KPE improves performance. However, further increasing the context to 6 frames leads to a performance drop, which suggests that relatively large temporal context may introduce less relevant motion cues.

Auxiliary CLS temporal module. We further explore different designs for the auxiliary temporal modeling route on the CLS token, including TSM (Lin et al., 2019), TokenT1D (Wu et al., 2023), and TokenShift (Zhang et al., 2021). As shown in Table 7(d), all temporal variants improve over the baseline, confirming that even a lightweight temporal interaction on the global token is beneficial.

4 Conclusion

In this paper, we introduced ***P**arallel **K**inematic **S**elective **S**tate **S**pace **S**canners* (PKS^4), a framework with linear-complexity temporal scanning. By extracting motion-aware kinematic priors and integrating them with parallel state-space scanning, PKS^4 preserves the 2D spatial token layout while achieving linear-complexity temporal modeling. Our extensive evaluations show that PKS^4 achieves strong performance across both temporal-heavy and spatial-heavy benchmarks, with particularly clear gains on SSV2 where fine-grained motion reasoning is critical. Moreover, the results reveal a broader insight: SSMs become substantially more effective for video understanding when their state transitions are driven by motion-aware priors rather than generic flattened tokens. PKS^4 also demonstrates training efficiency: its ViT-B/16 configuration uses approximately $10.7\times$ less video-domain training compute than the compared VideoMamba-M$_{800e}$ configuration, while its measured GPU-memory footprint remains moderate relative to the evaluated multi-adapter methods.

REFERENCES

Anurag Arnab, Mostafa Dehghani, Georg Heigold, Chen Sun, Mario Lučić, and Cordelia Schmid. Vivit: A video vision transformer. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 6836–6846, 2021.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

Gedas Bertasius, Heng Wang, and Lorenzo Torresani. Is space-time attention all you need for video understanding? 2021.

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.

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 *Proceedings of the IEEE international conference on computer vision*, pp. 5842–5850, 2017.

Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. In *First conference on language modeling*, 2024.

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

Will Kay, Joao Carreira, Karen Simonyan, Brian Zhang, Chloe Hillier, Sudheendra Vijayanarasimhan, Fabio Viola, Tim Green, Trevor Back, Paul Natsev, et al. The kinetics human action video dataset. *arXiv:1705.06950*, 2017.

Kunchang Li, Xinhao Li, Yi Wang, Yinan He, Yali Wang, Limin Wang, and Yu Qiao. Videomamba: State space model for efficient video understanding. In *European conference on computer vision*, pp. 237–255. Springer, 2024.

Ji Lin, Chuang Gan, and Song Han. Tsm: Temporal shift module for efficient video understanding. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 7083–7093, 2019.

Ziyi Lin, Shijie Geng, Renrui Zhang, Peng Gao, Gerard De Melo, Xiaogang Wang, Jifeng Dai, Yu Qiao, and Hongsheng Li. Frozen clip models are efficient video learners. In *European Conference on Computer Vision*, pp. 388–404. Springer, 2022.

Junting Pan, Ziyi Lin, Xiatian Zhu, Jing Shao, and Hongsheng Li. St-adapter: Parameter-efficient image-to-video transfer learning. *Advances in Neural Information Processing Systems*, 35:26462–26477, 2022.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pp. 8748–8763. PmLR, 2021.

Wenhao Wu, Zhun Sun, and Wanli Ouyang. Revisiting classifier: Transferring vision-language models for video recognition. In *Proceedings of the AAAI conference on artificial intelligence*, volume 37, pp. 2847–2855, 2023.

Taojiannan Yang, Yi Zhu, Yusheng Xie, Aston Zhang, Chen Chen, and Mu Li. Aim: Adapting image models for efficient video action recognition. *arXiv preprint arXiv:2302.03024*, 2023.

Yijun Yang, Zhaohu Xing, Lequan Yu, Chunwang Huang, Huazhu Fu, and Lei Zhu. Vivim: A video vision mamba for medical video segmentation. *arXiv preprint arXiv:2401.14168*, 2024.

Huanjin Yao, Wenhao Wu, and Zhiheng Li. Side4video: Spatial-temporal side network for memory-efficient image-to-video transfer learning. *arXiv preprint arXiv:2311.15769*, 2023.

Hao Zhang, Yanbin Hao, and Chong-Wah Ngo. Token shift transformer for video classification. In *Proceedings of the 29th acm international conference on multimedia*, pp. 917–925, 2021.

A KINEMATIC-PRIOR RESPONSES

Figure 4 shows response overlays on frames from two video examples. The overlays highlight regions around the hands and objects in these examples.



Figure 4: Kinematic-prior responses for two video examples. Each example shows three RGB frames in the top row and the corresponding response overlays in the bottom row.

AI USE STATEMENT

We used generative AI tools to search for and review relevant literature, polish the language, and adjust the LaTeX formatting.