




Transformer or Mamba for Temporal Action Localization? Insights from a Comprehensive Experimental Comparison Study

Zejian Zhang¹^a, Cristina Palmero²^b and Sergio Escalera¹^c

¹Universitat de Barcelona and Computer Vision Center, Barcelona, Spain

²Department of Engineering, King's College London, London, U.K.

Keywords: Temporal Action Localization (TAL), Temporal Action Detection (TAD), Transformer, Self-Attention, State Spate Models (SSMs), Mamba, Multi-Scale.

Abstract: Deep learning models need to encode both local and global temporal dependencies for accurate temporal action localization (TAL). Recent approaches have relied on Transformer blocks, which has a quadratic complexity. By contrast, Mamba blocks have been adapted for TAL due to their comparable performance and lower complexity. However, various factors can influence the choice between these models, and a thorough analysis of them can provide valuable insights into the selection process. In this work, we analyze the Transformer block, Mamba block, and their combinations as temporal feature encoders for TAL, measuring their overall performance, efficiency, and sensitivity across different contexts. Our analysis suggests that Mamba blocks should be preferred due to their performance and efficiency. Hybrid encoders can serve as an alternative choice when sufficient computational resources are available.

1 INTRODUCTION

Temporal action localization (TAL) is a challenging yet critical task for video analysis, owing to its wide range of real-world applications such as video surveillance, sports analytics, and human activity understanding (Elharrouss et al., 2021) (Ghosh et al., 2023) (Saleem et al., 2023). TAL involves identifying precise start and end timestamps of actions and assigning their action labels in untrimmed videos. Due to the complexity of video contents, actions in videos often exhibit ambiguity in boundaries and various durations, making it difficult for traditional convolutional neural network (CNN)-based models (Gong et al., 2020) (Zhu et al., 2021) to learn necessary temporal dependencies to output precise action boundaries.

Recent approaches (Zhang et al., 2022) (Shao et al., 2023) (Zhang et al., 2024) in TAL have adapted Transformers (Vaswani et al., 2017) to encode video frames into a multi-scale representation (see Fig. 1A), leveraging the advantages of both self-attention (Vaswani et al., 2017) and multi-resolution action detection. However, the self-attention operation has a

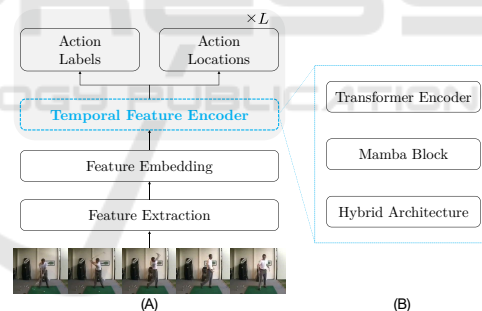





Figure 1: Illustration of the multi-scale TAL architecture. (A) The model involves extracting clip features from video frames, applying a feature embedding for dimensional reduction, and encoding the input into an L -level multi-scale representation by temporal feature encoders, followed by an action localization head. (B) The three types of temporal feature encoder we compare and study in this work.

quadratic complexity, representing one of the biggest hurdles for processing long sequences. Researchers have been exploring new possibilities to break this quadratic computational cost (Gu et al., 2021).

Structured State Space Models (SSMs) (Gu et al., 2021) (Gu et al., 2022), particularly the Mamba block (Gu and Dao, 2023), have been proposed and have demonstrated promising performance in various long sequence modeling tasks (Gu and Dao, 2023). Compared to the state-of-the-art Transformer-based meth-

^a <https://orcid.org/0000-0001-9810-3726>

^b <https://orcid.org/0000-0002-6085-6527>

^c <https://orcid.org/0000-0003-0617-8873>

ods, Mamba has fewer parameters and scales nearly linearly with sequence length, making it effective for processing untrimmed video sequences for TAL. Early adaptations in this direction include ActionMamba (Chen et al., 2024) and CausalTAD (Liu et al., 2024b). Despite these initial attempts to explore Mamba’s potential for TAL, there is a lack of comparison among these encoders, especially when combining them (the Transformer and Mamba block) into a single encoder used to learn temporal dependencies. Such a comparison could be useful for providing insights into further explorations of their capacities in TAL and other video understanding tasks.

In this work, we aim to conduct a comprehensive experimental comparison of the well-established Transformer, the Mamba block, and their integration into a hybrid architecture. We focus on their performance, complexity, and efficiency in handling video sequences for the TAL task. Specifically, we analyze four groups of models equipped with different temporal feature encoders, resulting in a total of 12 distinct models. Our contributions are as follows:

1. We present a comprehensive experimental comparison of the Transformer and Mamba block-based models for TAL. These models are categorized into: Transformer-based models utilizing either global or sliding window attention; pure Mamba block-based models, including the original Mamba block, bidirectional Mamba (ViM) (Zhu et al., 2024), and decomposed bidirectionally Mamba (DBM) (Chen et al., 2024) blocks; a sequential architecture that integrates the Transformer block with Mamba blocks; and a parallel hybrid model that utilizes the Transformer and Mamba blocks concurrently

2. We adapt the MambaFormer block (Park et al., 2024) into a multi-scale architecture, marking, to the best of our knowledge, its first application for TAL tasks. We extend this configuration to include six different temporal feature encoders.

3. We conduct experiments on three commonly used TAL datasets, providing insights for selecting different temporal feature encoders based on datasets and requirements. Our evaluation uses various metrics, including the standard performance measures, efficiency, and sensitivity. Additionally, we analyze the scaling capability of the original Mamba block. Our results suggest that, in most cases, pure Mamba block-based models should be preferred due to their superior performance and lower complexity. Hybrid counterparts may serve as an alternative choice for scenarios with higher performance demands.

2 RELATED WORK

Temporal Action Localization. Research in TAL can be divided into two categories: two-stage and one-stage methods. Two-stage methods (Zhu et al., 2021) (Sridhar et al., 2021) (Zhao et al., 2021) (Chen et al., 2022) involve generating coarse proposals followed by refining these proposals to produce the final outputs. By contrast, one-stage methods (Lin et al., 2021) (Liu et al., 2022) (Zhang et al., 2022) (Kang et al., 2023) (Zhao et al., 2023) (Shi et al., 2023) tackle both sub-tasks simultaneously within a unified framework. An effective TAL model relies on encoding relevant long and short temporal dependencies in the input data. Various deep learning methods have been explored for this purpose, including CNN (Lecun et al., 1990), graph neural networks (GNN) (Scarselli et al., 2009), and attention-based (Vaswani et al., 2017) methods. Recent advancements (Zhang et al., 2022) (Shao et al., 2023) (Zhang et al., 2024) (Chen et al., 2024) (Liu et al., 2024b) have particularly focused on one-stage methods, which adapt temporal encoders such as the Transformer block (Vaswani et al., 2017) and Mamba block (Gu and Dao, 2023) to transform input videos into a multi-scale representation, where the temporal resolution is downsampled at higher levels, thereby enabling the localization of actions across different resolutions. These methods have demonstrated superior performance compared to previous CNN- or GNN-based approaches thanks to their multi-scale design and the ability to process long sequence data. In this work, we focus on analyzing the well-established Transformer block, the emerging Mamba block, and the combination of both into a hybrid architecture (Park et al., 2024) (Liu et al., 2024b), exploring their performance and computational efficiency as temporal feature encoders in the multi-scale architecture for TAL task on different benchmarks.

State Space Models. Inspired by the traditional SSMs in control theory (Kalman, 1960), structured SSMs (Gu et al., 2021) (Gu et al., 2022) (Fu et al., 2023) were proposed for modeling long-range sequences such as in natural language processing, overcoming the performance issues in CNNs (Lecun et al., 1990) and recurrent neural networks (RNNs) (Bisong and Bisong, 2019), as well as the quadratic scaling issue in the Transformer (Vaswani et al., 2017) encoders. Mamba, in particular, combines an SSM layer (Fu et al., 2023) with gated MLP (multi-layer perceptron) (Liu et al., 2024a), achieving on par performance in sequence data modeling, such as language and audio (Gu and Dao, 2023), while demonstrating linear scaling with sequence length. Mamba

has been explored for visual applications, such as Vision Mamba (Zhu et al., 2024) for images and Video Mamba Suite (Chen et al., 2024) for video understanding. We study and analyze the original Mamba block (Gu and Dao, 2023) and its variants, namely, ViM (Zhu et al., 2024) and DBM (Chen et al., 2024), as the sole encoder in the multi-scale architecture for TAL due to their strong capability to process long sequence data.

Hybrid Transformer and Mamba. While Mamba has shown promising performance in certain tasks, it has been found that a pure Mamba-equipped encoder does not perform as well as Transformer-based counterparts, e.g., in in-context learning tasks that require non-standard retrieval capabilities (Park et al., 2024). Various studies have explored integrating the Transformer block with Mamba blocks into a hybrid architecture to capitalize on their complementary strengths for capturing long-range dependencies. For instance, MambaVision (Hatamizadeh and Kautz, 2024) augments the Mamba block with Transformer layers to enhance global context recovery. MambaFormer (Park et al., 2024) combines Mamba with self-attention blocks. Instead of a sequential combination of Transformer and Mamba blocks, CausalTAD (Liu et al., 2024b) introduces a dual architecture for TAL, where the input is processed simultaneously by a Transformer block and a Mamba block. Despite the existence of other possible combinations, this study concentrates on two architectures: MambaFormer (a *sequential* model) and CausalTAD (a *parallel* model). Our goals thus are: 1) adapt the MambaFormer architecture for TAL tasks, and 2) evaluate their performance as temporal feature encoders for TAL applications.

3 METHODOLOGY

3.1 TAL Problem Definition

Given an input video, it is represented by a set of pre-extracted features $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_T\}$, where $t \in \{1, 2, \dots, T\}$ are the number of frames and T varies across videos. The goal is to predict all the action instances \mathbf{Y} based on \mathbf{X} , where $\mathbf{Y} = \{\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_N\}$ are the ground truth action instances. Each action instance is defined as $\mathbf{y}_i = (s_i, e_i, c_i)$, where $i \in \{1, 2, \dots, N\}$ are the number of action instances, $s_i, e_i \in [1, T]$ are the start and end timestamps ($s_i < e_i$), and $c_i \in \{1, 2, \dots, C\}$ is the corresponding action label.

3.2 Model Overview

We employ a multi-scale architecture, as illustrated in Fig. 1, to construct the model used for evaluation, due to its straightforward design and superior performance demonstrated in recent one-stage TAL methods (Zhang et al., 2022) (Shi et al., 2023) (Zhang et al., 2024). The model comprises three parts: feature extraction and embedding, multi-scale temporal feature encoding, and action localization. The input videos are processed by pre-trained CNN models, such as I3D (Carreira and Zisserman, 2017) on Kinetics (Kay et al., 2017), to extract clip features. The extracted features are then embedded into a latent space that matches the dimensional requirements of the temporal feature encoder. Action localization is performed by a two-branch head, one for classifying action labels and the other for regressing action boundaries. The head is shared across all levels of the multi-scale feature representation. Since our primary objective is to analyze the performance of different temporal feature encoders, we fix the architecture design of the feature embedding and localization head while varying the temporal feature encoders.

In the following section, we present the temporal feature encoders utilized for encoding the input video into a multi-scale representation.

3.3 Temporal Feature Encoders

The evaluated temporal feature encoders are categorized into three types: pure Transformer blocks, pure Mamba blocks, and hybrid architectures that combine self-attention layers with Mamba blocks.

3.3.1 Preliminaries

In this section, we provide a brief review of the self-attention mechanism and the SSMS, which serve as the core operation for the Transformer block and Mamba block, respectively.

Self-Attention. The self-attention operation takes an arbitrary length of token embedding (\mathbf{X}) as input and aggregates relevant information by measuring the similarity among tokens. The attention score is computed by $\text{Attn}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{Softmax}\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_k}}\right)\mathbf{V}$, where \mathbf{Q}, \mathbf{K} , and \mathbf{V} are the queries, keys and values projected from the input embedding. To encode richer and more robust temporal relationships, the multi-head attention is commonly used, with each attention head focusing on different aspects simultaneously. Subsequently, the outputs from these heads are concatenated to form the final attention scores. In order to mitigate the quadratic complexity, *sliding window* at-

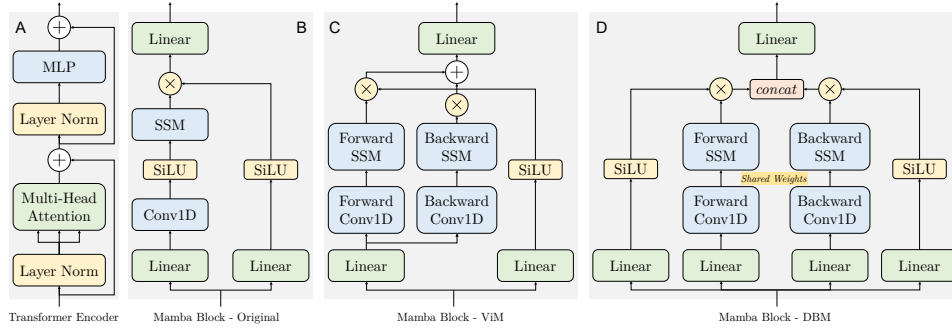


Figure 2: Illustration of temporal feature encoders. (A) The Transformer (Vaswani et al., 2017) block, which utilizes (*sliding window*) multi-head self-attention. (B) The original Mamba block (Gu and Dao, 2023), which utilizes SSMs. (C) The ViM block (Zhu et al., 2024), which adds a backward scanning to the original Mamba block. (D) The DBM block (Chen et al., 2024), which separates the input projections and adds a backward scanning with shared weights.

tion is often used instead, restricting the attention calculation to a local window of fixed size.

State Space Models. State space models were introduced in control theory for modeling continuous systems using state variables (Kalman, 1960). These models map an input to an output through a hidden state. Inspired by this approach, the structured SSMs, such as S4 (Gu et al., 2022) and Mamba (Gu and Dao, 2023), have been proposed for processing language and vision data. The continuous parameters are discretized to $\bar{\mathbf{A}}$ and $\bar{\mathbf{B}}$ through a transformation method, e.g., zero-order hold (ZOH). Subsequently, the SSMs for processing discrete input at time t is defined as: $h_t = \bar{\mathbf{A}}h_{t-1} + \bar{\mathbf{B}}x_t$, $y_t = \mathbf{C}h_t$. Finally, for an input \mathbf{X} , the computation of the output \mathbf{Y} involves a global convolution, denoted as $\mathbf{Y} = \mathbf{X} * \bar{\mathbf{K}}$, where $\bar{\mathbf{K}} = (\mathbf{C}\bar{\mathbf{B}}, \mathbf{C}\bar{\mathbf{A}}\bar{\mathbf{B}}, \dots, \mathbf{C}\bar{\mathbf{A}}^{M-1}\bar{\mathbf{B}})$ is a structured convolutional kernel, and M is the length of \mathbf{X} .

3.3.2 Transformer Block

A Transformer block (see Fig. 2A) consists of a (*sliding window*) multi-head self-attention (MHA) layer followed by an MLP block. To facilitate stable and efficient training, Layer Normalization (LN) is applied before the MHA and MLP layers, and residual connections are added after each. Given an input \mathbf{X}_i , the output of the Transformer block \mathbf{X}_o is obtained by: $\mathbf{X}' = \mathbf{X}_i + \text{MHA}(\text{LN}(\mathbf{X}_i))$, $\mathbf{X}_o = \mathbf{X}' + \text{MLP}(\text{LN}(\mathbf{X}'))$.

3.3.3 Mamba Blocks

The Mamba blocks used for temporal feature encoding are the original Mamba block (Gu and Dao, 2023), incorporating a forward SSM, and two variants (i.e., ViM (Zhu et al., 2024) and DBM (Chen et al., 2024)) which, in addition to the forward SSM, include a backward SSM.

Mamba Block. A Mamba block (see Fig. 2B) combines the SSM layer proposed in (Fu et al., 2023)

with gated MLP (Liu et al., 2024a). For an input \mathbf{X}_i , it is mapped to \mathbf{x} and \mathbf{z} with linear projections. \mathbf{x} undergoes transformation via a Conv1D layer followed by an SSM layer. The transformed \mathbf{x} is then gated by \mathbf{z} , after which a projection layer is applied to produce the final output \mathbf{X}_o . The process is defined as follows:

$$\begin{aligned} \mathbf{x} &= \text{Proj}_x(\mathbf{X}_i), \mathbf{z} = \text{Proj}_z(\mathbf{X}_i), \\ \mathbf{x}' &= \text{SSM}(\text{SiLU}(\text{Conv1D}(\mathbf{x}))), \\ \mathbf{X}_o &= \text{Proj}_{out}(\text{SiLU}(\mathbf{z}) * \mathbf{x}'). \end{aligned} \quad (1)$$

ViM Block. In comparison to the Mamba block, the ViM block (Zhu et al., 2024) (as shown in Fig. 2C) features a forward branch and a backward branch that process the projected input simultaneously in both directions. The ViM block transforms an input \mathbf{X}_i into an output \mathbf{X}_o through the following process:

$$\begin{aligned} \mathbf{x} &= \text{Proj}_x(\mathbf{X}_i), \mathbf{z} = \text{Proj}_z(\mathbf{X}_i), \\ \mathbf{x}'_f &= \text{SSM}_f(\text{SiLU}(\text{Conv1D}_f(\mathbf{x}))), \\ \mathbf{x}'_b &= \text{SSM}_b(\text{SiLU}(\text{Conv1D}_b(\mathbf{x}))), \\ \mathbf{X}_o &= \text{Proj}_{out}(\text{SiLU}(\mathbf{z}) * (\mathbf{x}'_f + \mathbf{x}'_b)). \end{aligned} \quad (2)$$

DBM Block. A DBM block (see Fig. 2D) shares a similar idea with the ViM block, i.e., introducing a backward branch. Differently, DBM blocks separate the projection processes to create distinct inputs for the Conv1D and SSM transformations. The parameters of the Conv1D and SSM operations are shared. The outputs from both branches are separately gated, which are then concatenated and projected to generate the final output. The compressed features, \mathbf{X}_o , for an input, \mathbf{X}_i , are defined as follows:

$$\begin{aligned} \mathbf{x}_f &= \text{Proj}_{x_f}(\mathbf{X}_i), \mathbf{z}_f = \text{Proj}_{z_f}(\mathbf{X}_i), \\ \mathbf{x}_b &= \text{Proj}_{x_b}(\mathbf{X}_i), \mathbf{z}_b = \text{Proj}_{z_b}(\mathbf{X}_i), \\ \mathbf{x}'_f &= \text{SSM}(\text{SiLU}(\text{Conv1D}(\mathbf{x}_f))), \\ \mathbf{x}'_b &= \text{SSM}(\text{SiLU}(\text{Conv1D}(\mathbf{x}_b))), \\ \mathbf{X}_o &= \text{Proj}_{out}(\text{SiLU}(\mathbf{z}_f) * (\mathbf{x}'_f) \parallel \text{SiLU}(\mathbf{z}_b) * \mathbf{x}'_b). \end{aligned} \quad (3)$$

3.3.4 Hybrid Architecture

A hybrid architecture combines the Transformer block (or self-attention layer) with the Mamba block, enabling the model to encode diverse dynamics. We focus on the MambaFormer architecture (Park et al., 2024) and the concurrent application of self-attention and the Mamba block, e.g., CausalTAD (Liu et al., 2024b), as described below.

MambaFormer. The MambaFormer (see Fig. 3A) combines Mamba with attention blocks. It replaces the positional encoding and MLP block in the standard Transformer block with two Mamba blocks. The skip connections after each block, as used in the Transformer block, are retained.

CausalTAD. CausalTAD (see Fig. 3B) extends the dual design in the DBM block (Chen et al., 2024) to the multi-head self-attention operation, resulting in a causal attention block (CAB). Specifically, CAB restricts the context of attention to only past or future tokens. A CAB and a DBM-like block are utilized to process the input simultaneously, aiming to capture both long-range temporal relationships and causality information.

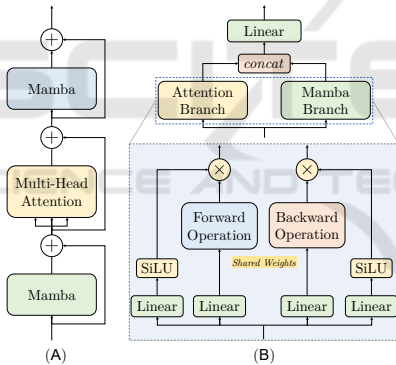


Figure 3: Illustration of the hybrid architectures studied. (A) The MambaFormer. (B) The concurrent application of self-attention and Mamba block (*top*). The Attention and Mamba Branches share a similar architecture (*bottom*), except that in forward and backward operations, the Attention Branch uses multi-head self-attention while the Mamba Branch uses Conv1D followed by an SSM layer. The parameters of the both operations are shared inside the Attention and Mamba Branches, respectively.

3.4 Training and Inference

For training, we employ a focal loss (Tian et al., 2019) and distance Intersection over Union (IoU) loss (Zheng et al., 2020) to optimize the action classification and boundary distance regression, respectively. For inference, we select a subset of the predictions whose classification score is higher than a pre-defined threshold. Finally, we use Soft-NMS (Bodla et al.,

2017) to remove duplicates.

4 EXPERIMENTS

In this section, we conduct experiments on different TAL datasets to evaluate the performance of the studied temporal feature encoders, as discussed in Section 3.3. Given that the architecture of the models is fixed, the only aspect that varies among them is the applied temporal feature encoder, which influences their performance. As a result, the performance of the models reflects the capabilities of the temporal feature encoders. Therefore, we assess the performance of the trained models.

4.1 Datasets

We choose THUMOS14 (Idrees et al., 2017), ActivityNet-1.3 (Heilbron et al., 2015), and EPIC-KITCHENS-100 (Damen et al., 2018) for our experiments. THUMOS14 and ActivityNet-1.3 are third-person view human activity datasets, differing in size and action durations. EPIC-Kitchens-100 is a first-person view dataset with fine-grained in-door actions. Such properties can assess TAL models from different aspects. Specifically, THUMOS14 consists of 413 videos with 20 action classes, which we divided into a training set of 213 videos and a test set of 200 videos, following previous work (Zhang et al., 2022). The models were trained on the training set and evaluated on the test set. ActivityNet-1.3 (Heilbron et al., 2015), with nearly 20,000 videos across 200 action classes, was split into a training set of 10,024 videos, a validation set of 4,926 videos, and a test set of 5,044 videos. Following previous practice (Zhang et al., 2022), we trained the models on the training set and evaluated them on the validation set. EPIC-KITCHENS-100, which contains 100 hours of egocentric view activities, was split into a training and a validation set. Similar to previous approach (Zhang et al., 2022), we trained separate models for Verb and Noun actions on the training set and assessed them on the validation set.

4.2 Performance Metrics

The mean average precision (mAP) was used as a metric to evaluate the overall performance of the trained models. Following previous practice (Zhang et al., 2022) (Zhang et al., 2024), the mAP calculated at different temporal IoU (tIoU) thresholds and the averaged mAP across all these thresholds are reported. Specifically, the thresholds were set to [0.3:0.1:0.7]

for THUMOS14, [0.5:0.05:0.95] for ActivityNet-1.3, and [0.1:0.1:0.5] for EPIC-KITCHENS-100. Besides, the number of trainable parameters and FLOPs (floating point operations) are also reported to evaluate the model’s efficiency.

4.3 Implementations

Feature Extraction. Following previous work (Zhang et al., 2022) (Zhang et al., 2024), we extract I3D (Carreira and Zisserman, 2017) features for THUMOS14 and ActivityNet-1.3, and SlowFast (Feichtenhofer et al., 2019) features for EPIC-KITCHENS-100. Note that while TSP features (Alwassel et al., 2021) have demonstrated superior performance on ActivityNet-1.3 (Zhang et al., 2022) (Shi et al., 2023) compared to other feature extractors such as I3D, we prioritize investigating temporal feature encoders over using different features to enhance the model’s performance. Using a consistent feature extractor across datasets enables a more insightful comparison of the generalizability of these encoders.

Experimental Setup. We re-implemented the temporal feature encoders using PyTorch 2.1.2 with CUDA version 12.1. The configuration of the feature embedding layers and the action localization head remained consistent with the methods described in (Zhang et al., 2022) and (Chen et al., 2024). The preprocessing step for handling variable-length inputs (fixing the maximum input sequence length through either padding or cropping) followed a similar approach to that in (Zhang et al., 2022). Each model was trained from scratch on all datasets. The Adam optimizer (Kingma and Ba, 2017), along with a warm-up strategy (Liu et al., 2020), was employed for training. During training, we optimized the number of epochs, learning rate, batch size, window size (when using *sliding window* attention), and the balance coefficient for the classification and distance regression losses. The training and testing processes were conducted on a single NVIDIA GeForce RTX 3090 GPU.

4.4 Overall Performance

The overall performance, measured by mAP, is presented in Table 1 for THUMOS14 and ActivityNet-1.3, and in Table 2 for EPIC-Kitchens-100.

We selected the Transformer block with a sliding window attention-based model, i.e., ActionFormer (Zhang et al., 2022), as the comparison point for analyzing the performance of the trained models, specifically the pure Mamba block-based methods and the hybrid architectures that combine the Transformer block and Mamba block.

It is worth noting that the retrained ActionFormer models achieve different overall performance levels compared to the reported results in (Zhang et al., 2022). Similar differences are also observed for the CausalTAD (Liu et al., 2024b) architecture. These discrepancies may be attributed to variations in hardware and software setups. Additionally, we used the implementation from (Zhang et al., 2022) to handle the training and testing data, which differs from the implementation in CausalTAD (Liu et al., 2024b). This may be another factor that has led to the observed differences in performance for CausalTAD.

Performance on THUMOS14. On THUMOS14, the original Mamba block-based model surpasses the baseline at all tIoU thresholds, achieving an average mAP of 68.5%, which is slightly higher than the baseline of 67.9%. Furthermore, as can be observed, this method achieves the best average mAP and outperforms all other studied methods at every tIoU threshold, except at the threshold of 0.6, where its performance is slightly lower than that of the CausalTAD method (60.8% vs. 61.1%).

However, the other temporal feature encoders do not demonstrate similar superior performance, even for the ViM and DBM blocks, which share the most similar architecture. This suggests that the backward scanning implemented in these models may not enhance the representativeness of the learned features. Additionally, the backward design may also influence the hybrid models, as evidenced by the MambaFormer architectures, where the combination of attention blocks—whether global attention or sliding window attention—outperforms other configurations, such as ViM or DBM in combination with attention blocks. By contrast, the CausalTAD method, which achieves the same average mAP as the baseline and outperforms other hybrid variants, indicates that the concurrent application of the Mamba block with attention layers, along with the causal information encoded by the causal attention, may enhance the discriminative power of the learned representations.

Despite these differences, we observed that the gaps in average mAP among these models are not significant. Additionally, we noticed that the original Mamba block or DBM block-based methods generally perform better than others at higher tIoU thresholds, such as at 0.6 and 0.7.

Performance on ActivityNet-1.3. As shown in Table 1, both the pure Mamba block-based methods and hybrid approaches outperform the pure Transformer block-based methods, with the CausalTAD method achieving the best performance, surpassing the baseline by an average margin of 0.9%. Additionally, the CausalTAD method demonstrates supe-

Table 1: Overall performance measured by mAP on THUMOS14 and ActivityNet-1.3. The mAP values calculated are those at different tIoU thresholds and the averaged ones, i.e., [0.3:0.1:0.7] for THUMOS14 and [0.5:0.05:0.95] for ActivityNet-1.3. TFE: Temporal feature encoder, Attn: attention, Win: Window.

Method	TFE	Block	THUMOS14					ActivityNet-1.3				
			tIoU (%)					tIoU (%)				
			0.3	0.4	0.5	0.6	0.7	Avg.	0.5	0.75	0.95	Avg.
ActionFormer	Transformer	Global Attn	83.3	78.8	71.7	59.8	44.5	67.6	54.4	36.4	7.3	35.7
		Win Attn *	83.3	79.5	71.9	60.2	45.0	67.9	54.4	36.8	8.4	36.0
ActionMamba	Mamba	Mamba	83.3	79.7	72.3	60.8	46.1	68.5	54.5	37.5	7.4	36.3
		ViM	82.6	79.1	71.5	59.6	44.6	67.5	55.4	37.8	8.0	36.8
		DBM	82.9	78.8	71.3	60.2	45.6	67.8	55.2	37.5	7.8	36.7
		Mamba + Global Attn	82.3	79.1	70.9	60.3	45.7	67.7	55.0	37.3	7.4	36.5
MambaFormer	Mamba + Attn	Mamba + Win Attn	82.4	78.8	70.8	60.2	44.4	67.3	54.7	37.1	7.8	36.3
		ViM + Global Attn	81.5	78.4	70.3	59.2	45.5	67.0	55.1	37.4	8.1	36.6
		ViM + Win Attn	82.0	78.1	70.6	59.4	44.8	67.0	55.0	37.3	7.6	36.6
		DBM + Global Attn	82.6	78.2	71.0	59.6	44.6	67.2	54.6	<u>37.6</u>	8.3	36.6
		DBM + Win Attn	81.4	77.2	70.2	58.7	44.1	66.3	54.9	37.4	7.6	36.5
CausalTAD	Mamba + Attn	DBM + Global Causal Attn *	82.4	79.0	71.9	61.1	44.9	67.9	55.3	37.8	8.9	36.9

* Retained models. **Bold** and **underlined** numbers indicate the best and second-best performances in each column, respectively.

rior performance at higher tIoU thresholds, particularly at 0.95. Similar to THUMOS14, while these methods exhibit differences in performance, they are not significant.

Another interesting observation is that causal encoding appears to be an important factor for boosting the model’s performance, as can be seen from the models that integrate the ViM or DBM blocks.

Performance on EPIC-Kitchens-100. As shown in Table 2, The best-performing methods on EPIC-Kitchens-100 are the pure DBM block-based model for Verb actions and the sequential integration of the original Mamba block with sliding window attention model for Noun actions. The highest average mAP surpasses the baseline by 1.1% for both Verb and Noun actions. Additionally, pure Mamba block-based models outperform others for Verb actions, particularly the ViM and DBM blocks, likely due to their backward scanning design. For Noun actions, the MambaFormer architecture appears to be the preferred choice, as the models in this category achieve both the best and second-best overall performances.

Discussion. Based on the performance results, it is clear that no single method consistently outperforms others across all the datasets. This suggests that a customized dataset may require a tailored model. For instance, the pure original Mamba block-based method may be suitable for datasets like THUMOS14, where action instances with short durations account for the majority (Wu et al., 2021). A more complex model, such as CausalTAD, should be chosen for ActivityNet-1.3. For EPIC-Kitchens-100 Verb actions, the pure DBM-based model is recommended, while for Noun actions, the MambaFormer model—specifically, the combination of the original Mamba block with sliding window attention or the ViM block with global attention—should be utilized. Alternatively, the pure Mamba block-based models or their

hybrid counterparts could be selected for better generalization across all the datasets.

4.5 Efficiency Analysis

Model efficiency is a crucial factor to consider when selecting a deep learning model for processing real-world, large-scale data, especially when hardware resources are limited. We present the number of trainable parameters and FLOPs as efficiency metrics to measure the model’s resource consumption. Specifically, a higher number of parameters indicates a larger model that requires more memory, while FLOPs reflect the computational complexity.

We use Python library `fvcore` to count the number of parameters and FLOPs. For calculating FLOPs, we follow the method described in (Zhang et al., 2022), using a fixed-size tensor as input to the model. Specifically, we use a tensor size of [1, 2408, 2304], i.e., an input of 2304 time steps, for THUMOS14 models, [1, 2304, 160] for ActivityNet-1.3 models, and [1, 2304, 2304] for EPIC-Kitchens-100 models.

As shown in Table 3, pure Mamba block-based models exhibit the fewest parameters and consume the least FLOPs. In contrast, CausalTAD has the highest number of parameters and consumes the most FLOPs. Compared to the Transformer models, MambaFormer models generally consume a similar number of FLOPs while having more parameters due to the integration of the Mamba blocks.

Discussion. Based on the efficiency performance, pure Mamba block-based models are preferable when computational resources are limited. Hybrid models can serve as an alternative choice for improved performance across different datasets when resources are more readily available.

Table 2: Overall performance on EPIC-Kitchens-100. The presented results include mAP values at different tIoU thresholds, ranging from [0.1:0.1:0.5], along with the averaged ones for both Verb and Noun actions. TFE: Temporal feature encoder, Attn: attention, Win: Window.

Method	TFE	Block	Verb - tIoU (%)					Noun - tIoU (%)						
			0.1	0.2	0.3	0.4	0.5	Avg.	0.1	0.2	0.3	0.4	0.5	Avg.
ActionFormer	Transformer	Global Attn	26.9	25.9	24.5	22.5	18.9	23.7	25.2	24.1	22.4	20.0	16.2	21.6
		Win Attn *	26.5	25.4	24.3	22.4	19.3	23.6	24.7	23.6	22.0	19.6	16.0	21.2
ActionMamba	Mamba	Mamba	26.6	25.6	23.9	22.2	19.6	23.6	25.5	24.2	22.6	20.4	16.8	21.9
		ViM	<u>27.5</u>	26.3	25.1	22.6	19.0	<u>24.1</u>	25.2	24.2	22.4	20.0	16.8	21.7
		DBM	28.0	27.2	25.7	23.2	19.4	24.7	24.8	23.7	21.9	19.8	15.3	21.1
MambaFormer	Mamba + Attn	Mamba + Global Attn	27.0	25.9	24.5	<u>22.8</u>	19.0	23.9	25.5	24.2	22.6	20.4	16.8	21.9
		Mamba + Win Attn	26.8	25.8	23.9	22.2	19.3	23.6	26.0	24.8	23.2	20.8	<u>17.0</u>	22.3
		ViM + Global Attn	26.3	<u>25.6</u>	23.9	21.9	18.7	23.3	25.6	<u>24.4</u>	22.7	<u>20.6</u>	17.5	<u>22.2</u>
		ViM + Win Attn	26.2	<u>25.6</u>	24.1	21.6	17.6	23.0	25.0	23.9	22.2	19.9	16.6	21.5
		DBM + Global Attn	27.1	26.4	24.3	22.3	19.2	23.8	25.4	24.5	<u>22.8</u>	20.3	16.3	21.8
		DBM + Win Attn	26.8	25.9	24.4	22.3	19.3	23.7	25.1	24.1	22.3	20.0	16.6	21.6
CausalTAD	Mamba + Attn	DBM + Global Causal Attn *	26.9	26.2	<u>24.6</u>	22.4	18.8	23.8	25.6	24.5	22.7	20.2	16.3	21.9

* Retrained models. **Bold** and **underlined** numbers indicate the best and second-best performances in each column, respectively.

Table 3: Efficiency measurements include the number of trainable parameters and FLOPs consumed by different models. The number of parameters was calculated separately for the THUMOS14, ActivityNet-1.3, and EPIC-Kitchens-100 models using the Python library *fvcore*. FLOPs were calculated for processing tensors of sizes [1, 2408, 2304] for THUMOS14 models, [1, 2304, 160] for ActivityNet-1.3 models, and [1, 2304, 2304] for EPIC-Kitchens-100 models.

Method	Block	THUMOS14		ActivityNet-1.3		EPIC - Verb		EPIC - Noun	
		# Params	FLOPs	# Params	FLOPs	# Params	FLOPs	# Params	FLOPs
ActionFormer	Global Attn	29.3M	115.4	8.1M	18.1	29.8M	118.4	30.1M	121.5
	Win Attn	29.3M	45.3	8.1M	17.4	29.8M	93.2	30.1M	96.4
ActionMamba	Mamba	19.0M	<u>33.0</u>	5.6M	<u>13.4</u>	19.5M	69.0	19.8M	<u>72.4</u>
	ViM	19.9M	37.8	5.9M	15.0	20.4M	78.5	20.7M	81.6
	DBM	<u>18.6M</u>	37.8	<u>5.5M</u>	15.0	<u>19.1M</u>	78.7	<u>19.4M</u>	82.6
MambaFormer	Mamba + Global Attn	42.8M	106.0	11.6M	16.5	43.3M	109.0	43.6M	111.9
	Mamba + Win Attn	42.8M	40.5	11.6M	15.9	43.3M	83.8	43.6M	86.8
	ViM + Global Attn	43.8M	115.5	12.0M	18.1	44.3M	118.5	44.6M	121.4
	ViM + Win Attn	43.8M	45.3	12.0M	17.5	44.3M	93.3	44.6M	96.3
	DBM + Global Attn	42.3M	115.5	11.5M	18.1	42.8M	118.5	43.2M	121.8
	DBM + Win Attn	42.3M	45.3	11.5M	17.5	42.8M	93.4	43.2M	96.7
CausalTAD	DBM + Global Causal Attn	52.1M	161.5	13.9M	29.4	52.6M	164.6	52.9M	167.6

Bold and **underline** numbers indicate highest and lowest number of parameters and FLOPs in each column.

4.6 Sensitivity Analysis

Actions in videos often vary in length, ranging from seconds to minutes. In addition, the number of action instances is not uniformly distributed and the type of actions may also vary a lot. These characteristics make TAL a challenging task for deep learning models, which should effectively identify all actions. In this section, we present a sensitivity analysis of the models evaluated on the THUMOS14 and ActivityNet-1.3 datasets, offering an additional metric for understanding the temporal feature encoders.

We use the tool presented in (Alwassel et al., 2018) for analysis. To perform the analysis, we define coverage, length, and the number of instances. Coverage refers to the relative length of the actions compared to the entire video, while length denotes the absolute duration of actions in seconds. Both coverage and length are categorized into Extra Small (XS), Small (S), Medium (M), Large (L), and Extra Large (XL). The number of instances refers to the total count of instances from the same class in a

video, categorized into Extra Small (XS), Small (S), Medium (M), and Large (L). The defined thresholds for Coverage on both datasets are XS: (0,0.02], S: (0.02,0.04], M: (0.04,0.06], L: (0.06,0.08], and XL: (0.08,1.0]. For Length, the thresholds are XS: (0,3], S: (3,6], M: (6,12], L: (12,18], and XL: >18 for THUMOS14, and XS: (0,30], S: (30,60], M: (60,120], L: (120,180], and XL: >180 for ActivityNet-1.3. For the number of instances, XS: 1, S: [2,40], M: [40,80], L: >80 and XS: 1, S: [2,4], M: [4,8], L: >8 are used for THUMOS14 and ActivityNet-1.3, respectively.

For each dataset, we selected the models with the highest overall performance (see Table 1) from each category to generate predictions for analysis.

On THUMOS14, the models perform well at predicting actions with short to large durations (see Fig. 4), particularly for small, medium, and large-duration actions. However, performance drops significantly for extra-large duration actions. The models also become more sensitive to videos with a large number of action instances. This behavior can be attributed to the characteristics of the dataset: 1) action durations in THU-

MOS14 are relatively short, and 2) each video typically contains only one action class. Regarding 1), the lack of sufficient long-duration actions makes the models more adept at encoding context for shorter actions. For 2), videos with many action instances provide the model with more data, improving its generalization capabilities on this kind of samples. Despite these commonalities, pure Transformer block-based models seem more effective at handling videos with a higher number of action instances, and they also perform well with respect to absolute duration. Thus, for videos featuring long durations and numerous action instances, pure Transformer block-based models may be a suitable choice. In other cases, the Mamba block-based models might be more appropriate. Another viable option, though resource-intensive, is the CausalTAD method, which demands greater computational resources.

For ActivityNet-1.3, the selected models demonstrate stronger regression capabilities (see Fig. 5) as action durations increase (likely because more continuous frames provide additional context about the actions), but they perform less effectively on videos with a large number of action instances. This decline is particularly noticeable in pure Transformer block-based models. Interestingly, however, these pure Transformer block-based models excel at handling extra-large duration, as the attention mechanism allows the model to effectively encode necessary context from the entire input. Therefore, they are more suitable for videos with long-duration actions but fewer action instances. For videos with short to relatively long-duration actions, where Transformer models show weaker performance, Mamba block-enabled models, especially the Mamba ViM block, are a better choice. These models offer comparable performance while being less complex than the alternatives that deliver superior performance but require more hardware resources due to the global attention computation.

Discussion. Overall, for THUMOS14, the original Mamba block-based model or the CausalTAD model should be chosen based on performance requirements. In the case of ActivityNet-1.3, the CausalTAD should be prioritized for performance needs, while the Mamba ViM-based method may be more suitable if hardware resources are a concern. However, it is important to note that the selection is highly dataset-dependent. Pure Transformer block-based methods can still be an excellent choice for developing real-world TAL applications, as they outperform other models in specific scenarios (as discussed previously) while maintaining lower complexity than their hybrid counterparts.

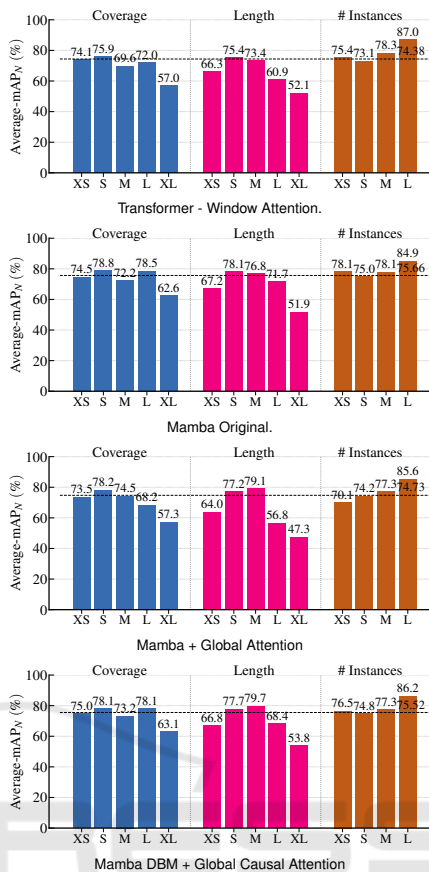


Figure 4: Sensitivity analysis on THUMOS14. The predictions were produced by the models with the highest overall performance from each category. These include the Transformer block with sliding window attention, the original Mamba block, the MambaFormer architecture featuring the original Mamba block with global attention, and the CausalTAD architecture. The dashed line represents the performance of the method across all instances in the analyzed dataset.

4.7 Class Level Analysis

In this section, we analyze the class-level performance of the selected models. Fig. 6 presents the per-class performance on THUMOS14. For ActivityNet-1.3 and EPIC-Kitchens-100, we select a subset of 20 classes for illustration, based on the number of action instances in each class. The selected classes range from small to large in the number of instances. Fig. 7 shows the results for ActivityNet-1.3, while Fig. 8 and Fig. 9 present the results for the Verb and Noun actions in EPIC-Kitchens-100, respectively.

For THUMOS14, the pure Mamba block-based methods perform well on many classes. However, they underperform on certain classes, such as “Soccer Penalty”, “Tennis Swing”, and “Throw Discus”, compared to hybrid models. This suggests that for classes

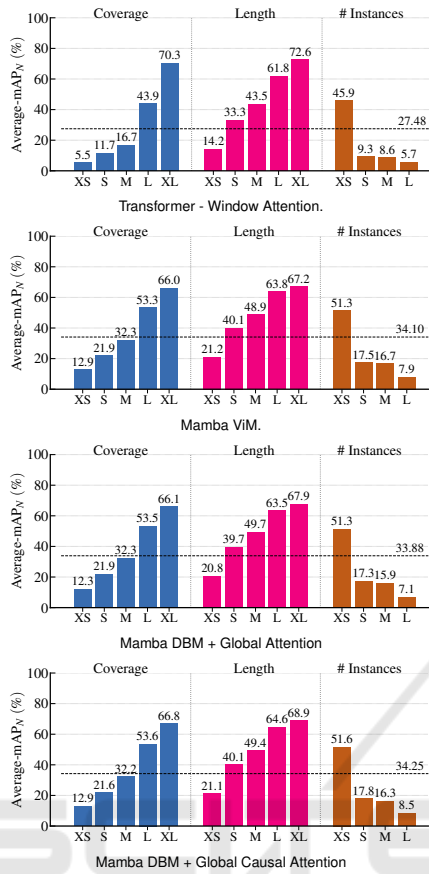


Figure 5: Sensitivity analysis on ActivityNet-1.3. The best-performing models in each category were selected for generating the outputs, specifically the Transformer block with sliding window attention, the Mamba ViM block, the MambaFormer architecture equipped with Mamba DBM block and global attention, and the CausalTAD architecture. The dashed line represents the performance of the method across all instances in the analyzed dataset.

where background scenes dominate the frames, a hybrid encoder may be preferable for capturing stronger contextual information.

On ActivityNet-1.3, it is surprising that none of the models can handle certain activities, such as “Skiing” and “Shot Put”, despite these activities having a relatively large number of action instances (over 210) and total durations (over 10,000 seconds). The wide variation in durations (ranging from 0.01 to over 210 seconds) may be a contributing factor. Nevertheless, pure Mamba block-based and hybrid models outperform pure Transformer-based models.

For EPIC-Kitchens-100, we observe a similar issue where none of the models perform well on certain actions, such as “brush”, “choose”, and “break” in the verb category, and “bacon”, “floor”, and “sausage” in the noun category. These actions exhibit wide variations in duration and a small number of instances,

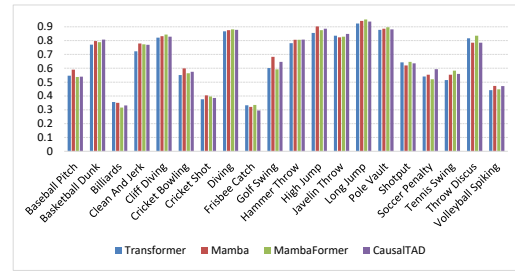


Figure 6: Per-class performance on THUMOS14, reported as averaged mAPs across tIoU thresholds [0.3:0.1:0.7].

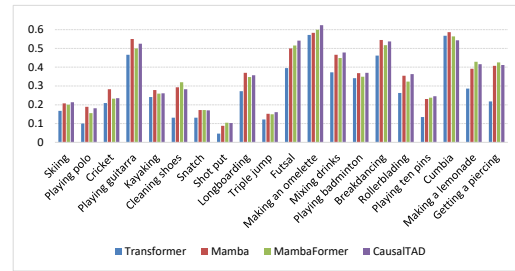


Figure 7: Per-class performance on ActivityNet-1.3 with 20-class subset, reported as averaged mAPs across tIoU thresholds [0.5:0.05:0.95].

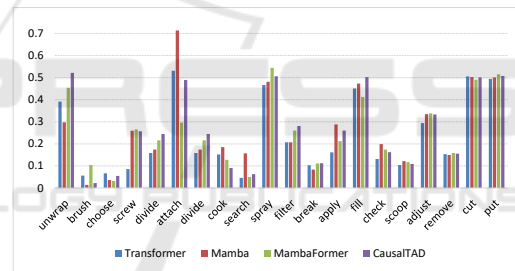


Figure 8: Per-class performance on EPIC-Kitchens-100 Verb actions with a 20-class subset, reported as averaged mAPs across tIoU thresholds [0.1:0.1:0.5].

which may limit the models’ effectiveness. However, in most cases, pure Mamba block-based and hybrid models outperform pure Transformer-based models and are generally the better choice.

Finally, we visualize some predicted outputs from THUMOS14 test set. As shown in Fig. 10, the pure Mamba block-based method aligns better for both very short and relatively long-duration actions, in line with previous analysis.

Discussion. When handling actions with varied durations, which may lack sufficient data for training or cause confusion due to background scenes, Mamba block-enabled models should be preferred.

4.8 Scalability Analysis

Scaling up a deep learning model by increasing the number of parameters may improve its performance.

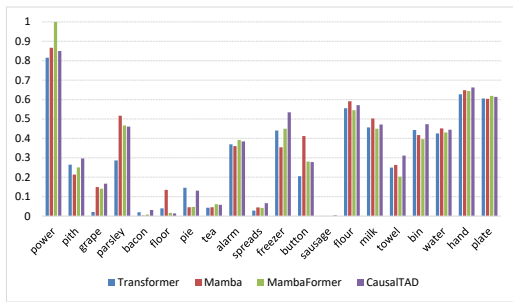


Figure 9: Per-class performance on EPIC-Kitchens-100 Noun actions with a 20-class subset, reported as averaged mAPs across tIoU thresholds [0.1:0.1:0.5].

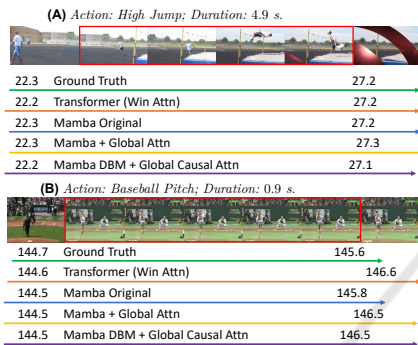


Figure 10: Predictions from THUMOS14 test set. (A) All methods are capable of handling relatively long actions. (B) For shorter actions, the pure Mamba block-based methods demonstrate better performance.

In this section, we perform an ablation analysis by adding an additional original Mamba block to each level of the encoder, aiming to match the parameter count of the Transformer-based method. The model’s performance after scaling is evaluated on the THUMOS14 dataset. The resulting mAPs, the number of parameters, and FLOPs are presented in Table 4.

Table 4: Scalability Analysis for the original Mamba block.

Encoder	0.3	0.4	0.5	0.6	0.7	Avg	# Params	FLOPs
Win Attn	83.3	79.5	71.9	60.2	45.0	67.9	29.3M	115.4
Mamba	83.3	79.7	72.3	60.1	46.1	68.5	19.0M	33.0
Scaled	83.5	79.0	71.9	60.2	46.8	68.3	30.9M	40.2

The number of parameters after scaling is 30.9M, which is comparable to that of the Transformer block with sliding window attention. As observed, the overall performance has slightly decreased compared to the original Mamba block prior to scaling. It is possible that, at this stage, the model is beginning to experience overfitting.

Discussion. The results indicate that scaling up the model does not improve performance on THUMOS14, suggesting that simpler models are better suited for smaller datasets to prevent overfitting during training. For larger datasets like ActivityNet-1.3,

more complex models may be necessary to meet performance requirements. For example, the CausalTAD method achieves the best results (see Table 1) on ActivityNet-1.3 with the highest number of parameters (see Table 3). Additionally, data augmentation techniques can be used for smaller datasets to enrich training samples to exploit the learning capacity of complex models.

4.9 Summary and Suggestion

Finally, we summarize our suggestions (in Table 5) based on the analysis, including performance (P), model efficiency (E), handling short (S) and long (L) duration actions, dealing with varied durations and limited data in each class (C), and data efficiency (D).

Table 5: Suggested feature encoders on each dataset. Small number indicates better method. The last column shows the best method, considering all performance requirements.

THUMOS14					
	Win Attn	Mamba Original	Mamba + Global Attn	CausalTAD	
P	2	1	4	3	
E	2	1	3	4	
S	4	1	3	2	
L	4	2	3	1	Mamba Original
C	4	1	2	3	
D	3	1	4	2	

ActivityNet-1.3					
	Win Attn	Mamba ViM	DBM + Global Attn	CausalTAD	
P	4	2	3	1	
E	2	1	3	4	
S	4	1	3	2	
L	4	2	3	1	Mamba ViM
C	4	1	2	3	
D	4	3	2	1	

EPIC-Kitchens-100 Verb					
	Global Attn	Mamba DBM	Mamba + Global Attn	CausalTAD	
P	4	1	2	3	
E	2	1	2	4	
S	3	4	2	1	
L	1	2	4	3	Mamba DBM
C	3	4	1	2	
D	1	3	4	2	

EPIC-Kitchens-100 Noun					
	Global Attn	Mamba	Mamba + Win Attn	CausalTAD	
P	4	2	1	3	
E	2	1	3	4	
S	1	4	3	2	
L	4	2	3	1	Mamba + Win Attn
C	4	3	2	1	
D	4	2	1	3	

5 CONCLUSIONS

In this work, we conducted a comparative analysis of the Transformer encoder, Mamba block, and their combinations as temporal feature encoders for TAL tasks across three commonly used benchmarks of various human actions. This comparison addresses the question of how to choose between the Transformer encoder and the Mamba block. Our findings suggest that both pure Mamba block-based models and hybrid models generally outperform those based on the Transformer encoder. However, the improved performance of the hybrid models comes at the expense of increased complexity. Therefore, pure Mamba

block-based models should be the preferred choice for developing TAL applications, especially given performance requirements and potential limitations in high-performance hardware resources. Additionally, we found that learning temporal dependencies in sequences from both directions—specifically through the ViM and DBM blocks, which incorporate a backward scanning process—can enhance the model’s performance for TAL.

We focused our analysis on a limited set of hybrid models. However, there are several other approaches to building hybrid models, such as those proposed in (Hatamizadeh and Kautz, 2024) and (Behrouz et al., 2024). One potential direction for future work could be exploring the adaptation of these architectures for TAL. Another avenue for future research would be developing models that leverage both the simplicity and performance of Mamba blocks, as well as their dual scanning capability.

ACKNOWLEDGEMENTS

This work has been partially supported by the Spanish project PID2022-136436NB-I00 and by ICREA under the ICREA Academia programme.

REFERENCES

- Alwassel, H., Caba Heilbron, F., Escorcia, V., and Ghanem, B. (2018). Diagnosing error in temporal action detectors. In *ECCV*.
- Alwassel, H., Giancola, S., and Ghanem, B. (2021). Tsp: Temporally-sensitive pretraining of video encoders for localization tasks. In *ICCV*, pages 3173–3183.
- Behrouz, A., Santacatterina, M., and Zabih, R. (2024). Mambamixer: Efficient selective state space models with dual token and channel selection. *arXiv 2403.19888*.
- Bisong, E. and Bisong, E. (2019). Recurrent neural networks (rnns). *Building Machine Learning and Deep Learning Models on Google Cloud Platform: A Comprehensive Guide for Beginners*, pages 443–473.
- Bodla, N., Singh, B., Chellappa, R., and Davis, L. S. (2017). Soft-nms — improving object detection with one line of code. In *2017 ICCV*, pages 5562–5570.
- Carreira, J. and Zisserman, A. (2017). Quo vadis, action recognition? a new model and the kinetics dataset. In *2017 CVPR*, pages 4724–4733.
- Chen, G., Huang, Y., Xu, J., Pei, B., Chen, Z., Li, Z., Wang, J., Li, K., Lu, T., and Wang, L. (2024). Video mamba suite: State space model as a versatile alternative for video understanding. *arXiv 2403.09626*.
- Chen, G., Zheng, Y., Wang, L., and Lu, T. (2022). DCAN: improving temporal action detection via dual context aggregation. In *AAAI 2022*, pages 248–257.
- Damen, D., Doughty, H., Farinella, G. M., Fidler, S., Furnari, A., Kazakos, E., Moltisanti, D., Munro, J., Perrett, T., Price, W., and Wray, M. (2018). Scaling egocentric vision: The epic-kitchens dataset. In *ECCV*.
- Elharrouss, O., Almaadeed, N., and Al-Maadeed, S. (2021). A review of video surveillance systems. *J Vis Commun Image Represent*, 77:103116.
- Feichtenhofer, C., Fan, H., Malik, J., and He, K. (2019). Slowfast networks for video recognition. In *2019 ICCV*, pages 6201–6210, Los Alamitos, CA, USA.
- Fu, D. Y., Dao, T., Saab, K. K., Thomas, A. W., Rudra, A., and Ré, C. (2023). Hungry Hungry Hippos: Towards language modeling with state space models. In *International Conference on Learning Representations*.
- Ghosh, I., Ramasamy Ramamurthy, S., Chakma, A., and Roy, N. (2023). Sports analytics review: Artificial intelligence applications, emerging technologies, and algorithmic perspective. *WRIEs Data Min. Knowl. Discov.*, 13(5):e1496.
- Gong, G., Zheng, L., and Mu, Y. (2020). Scale matters: Temporal scale aggregation network for precise action localization in untrimmed videos. In *2020 ICME*.
- Gu, A. and Dao, T. (2023). Mamba: Linear-time sequence modeling with selective state spaces. *arXiv 2312.00752*.
- Gu, A., Goel, K., and Ré, C. (2022). Efficiently modeling long sequences with structured state spaces. In *The International Conference on Learning Representations*.
- Gu, A., Johnson, I., Goel, K., Saab, K., Dao, T., Rudra, A., and Ré, C. (2021). Combining recurrent, convolutional, and continuous-time models with linear state space layers. *NIPS’21*, 34:572–585.
- Hatamizadeh, A. and Kautz, J. (2024). Mambavision: A hybrid mamba-transformer vision backbone. *arXiv 2407.08083*.
- Heilbron, F. C., Escorcia, V., Ghanem, B., and Niebles, J. C. (2015). Activitynet: A large-scale video benchmark for human activity understanding. In *2015 CVPR*.
- Idrees, H., Zamir, A. R., Jiang, Y.-G., Gorban, A., Laptev, I., Sukthankar, R., and Shah, M. (2017). The THUMOS challenge on action recognition for videos “in the wild”. *Comput. Vis. Image Underst.*, 155:1–23.
- Kalman, R. E. (1960). A New Approach to Linear Filtering and Prediction Problems. *J. Basic Eng.*, 82(1):35–45.
- Kang, T.-K., Lee, G.-H., Jin, K.-M., and Lee, S.-W. (2023). Action-aware masking network with group-based attention for temporal action localization. In *2023 WACV*, pages 6047–6056.
- Kay, W., Carreira, J., Simonyan, K., Zhang, B., Hillier, C., Vijayanarasimhan, S., Viola, F., Green, T., Back, T., Natsev, P., Suleyman, M., and Zisserman, A. (2017). The kinetics human action video dataset.
- Kingma, D. P. and Ba, J. (2017). Adam: A method for stochastic optimization. In *3rd ICLR Proceedings*.
- Lecun, Y., Boser, B., Denker, J., Henderson, D., Howard, R., Hubbard, W., and Jackel, L. (1990). Handwritten

- digit recognition with a back-propagation network. In Touretzky, D., editor, *NIPS 1989*, volume 2.
- Lin, C., Xu, C., Luo, D., Wang, Y., Tai, Y., Wang, C., Li, J., Huang, F., and Fu, Y. (2021). Learning salient boundary feature for anchor-free temporal action localization. In *CVPR*, pages 3320–3329.
- Liu, H., Dai, Z., So, D. R., and Le, Q. V. (2024a). Pay attention to mlps. In *NIPS'21*, Red Hook, NY, USA.
- Liu, L., Liu, X., Gao, J., Chen, W., and Han, J. (2020). Understanding the difficulty of training transformers. In *Proceedings of EMNLP 2020*.
- Liu, S., Sui, L., Zhang, C.-L., Mu, F., Zhao, C., and Ghanem, B. (2024b). Harnessing temporal causality for advanced temporal action detection. *arXiv 2407.17792*.
- Liu, X., Wang, Q., Hu, Y., Tang, X., Zhang, S., Bai, S., and Bai, X. (2022). End-to-end temporal action detection with transformer. *IEEE Trans Image Process*.
- Park, J., Park, J., Xiong, Z., Lee, N., Cho, J., Oymak, S., Lee, K., and Papailiopoulos, D. (2024). Can mamba learn how to learn? A comparative study on in-context learning tasks. In Salakhutdinov, R., Kolter, Z., Heller, K., Weller, A., Oliver, N., Scarlett, J., and Berkenkamp, F., editors, *Proceedings of the 41st International Conference on Machine Learning*, pages 39793–39812.
- Saleem, G., Bajwa, U. I., and Raza, R. H. (2023). Toward human activity recognition: a survey. *Neural Computing and Applications*, 35(5):4145–4182.
- Scarselli, F., Gori, M., Tsoi, A. C., Hagenbuchner, M., and Monfardini, G. (2009). The graph neural network model. *IEEE Trans Neural Networks*, 20(1):61–80.
- Shao, J., Wang, X., Quan, R., Zheng, J., Yang, J., and Yang, Y. (2023). Action sensitivity learning for temporal action localization. In *2023 IEEE ICCV Proceeding*, pages 13411–13423, Los Alamitos, CA, USA.
- Shi, D., Zhong, Y., Cao, Q., Ma, L., Lit, J., and Tao, D. (2023). Tridet: Temporal action detection with relative boundary modeling. In *2023 CVPR*, pages 18857–18866, Los Alamitos, CA, USA.
- Sridhar, D., Quader, N., Muralidharan, S., Li, Y., Dai, P., and Lu, J. (2021). Class semantics-based attention for action detection. In *2021 ICCV*, pages 13719–13728, Los Alamitos, CA, USA.
- Tian, Z., Shen, C., Chen, H., and He, T. (2019). Fcos: Fully convolutional one-stage object detection. In *2019 ICCV*, pages 9626–9635.
- Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. u., and Polosukhin, I. (2017). Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30.
- Wu, J., Sun, P., Chen, S., Yang, J., Qi, Z., Ma, L., and Luo, P. (2021). Towards high-quality temporal action detection with sparse proposals. *arXiv 2109.08847*.
- Zhang, C.-L., Wu, J., and Li, Y. (2022). Actionformer: Localizing moments of actions with transformers. In *ECCV 2022*, volume 13664, pages 492–510.
- Zhang, Z., Palmero, C., and Escalera, S. (2024). Dualh: A dual hierarchical model for temporal action localization. In *2024 FG Conference*, pages 1–10.
- Zhao, C., Liu, S., Mangalam, K., and Ghanem, B. (2023). Re2tal: Rewiring pretrained video backbones for reversible temporal action localization. In *2023 CVPR*, pages 10637–10647, Los Alamitos, CA, USA.
- Zhao, C., Thabet, A., and Ghanem, B. (2021). Video self-stitching graph network for temporal action localization. In *2021 ICCV*, pages 13638–13647.
- Zheng, Z., Wang, P., Liu, W., Li, J., Ye, R., and Ren, D. (2020). Distance-iou loss: Faster and better learning for bounding box regression. In *The AAAI Conference on Artificial Intelligence (AAAI)*, pages 12993–13000.
- Zhu, L., Liao, B., Zhang, Q., Wang, X., Liu, W., and Wang, X. (2024). Vision mamba: Efficient visual representation learning with bidirectional state space model. *arXiv 2401.09417*.
- Zhu, Z., Tang, W., Wang, L., Zheng, N., and Hua, G. (2021). Enriching local and global contexts for temporal action localization. In *2021 ICCV*, pages 13496–13505, Los Alamitos, CA, USA.