Scaling Action Detection: AdaTAD++ with Transformer-Enhanced Temporal-Spatial Adaptation

Tanay Agrawal* Abid Ali* Antitza Dantcheva Francois Bremond
INRIA Sophia Antipolis – Méditerranée, France
{firstname.lastname}@inria.fr * Joint first authors.

Abstract

Temporal Action Detection (TAD) is essential for analyzing long-form videos by identifying and segmenting actions within untrimmed sequences. While recent innovations like Temporal Informative Adapters (TIA) have improved resolution, memory constraints still limit large video processing. To address this, we introduce AdaTAD++, an enhanced framework that decouples temporal and spatial processing within adapters, organizing them into independently trainable modules. Our novel two-step training strategy first optimizes for high temporal and low spatial resolution, then vice versa, allowing the model to utilize both high spatial and temporal resolutions during inference, while maintaining training efficiency. Additionally, we incorporate a more sophisticated temporal module capable of capturing long-range dependencies more effectively than previous methods. Experiments on benchmark datasets, including ActivityNet-1.3, THUMOS14, and EPIC-Kitchens 100, demonstrate that AdaTAD++ achieves state-of-the-art performance. We also explore various adapter configurations, discussing their trade-offs regarding resource constraints and performance, providing valuable insights into their optimal application.

1. Introduction

Temporal Action Detection (TAD) is crucial in interpreting long-form videos by detecting specific actions and determining their start and end times within untrimmed sequences. Recent advancements, such as the introduction of Temporal Informative Adapters (TIA) in AdaTAD [16], have significantly improved model scalability and input capacity. This method has enabled an approximately 60 times increase in input frames as compared to previous approaches [21, 23, 30]. Our work further extends this capability to nearly 120 times the previous scale. For instance, our approach allows the processing of 6144 frames with high spatial resolution and a large backbone using the same computational resource constraints (same GPUs and CPUs). This remarkable scaling is made possible by parameter-efficient transfer learning (PETL) methods, such as adapters and our customisation of their use. Addressing this challenge of temporal processing is crucial, as real-world videos contain continuous context, and understanding the relationships between action occurrences enhances overall model performance. This is particularly evident in one of the more challenging TAD datasets, Epic-Kitchen 100, a cooking dataset. While segmenting videos into smaller parts has proven effective for identifying individual recipe steps, processing multiple steps within the same input allows the model to capture their relationships better, ultimately leading to improved performance.

We introduce **AdaTAD++**, a framework with efficient memory utilization and improved long-range temporal dependency modelling. Our primary contributions are: **(1)** the decoupling of spatial and temporal adapters, allowing independent optimization to maximize resolution utilisation, and **(2)** introducing a new temporal adapter architecture to improve temporal reasoning over previous approaches. Our approach combines a two-phase training strategy — first optimizing with low spatial and high temporal resolution, then refining with high spatial and low temporal resolution — to achieve high-resolution processing for both dimensions during inference. Interestingly, it also works well as a single-phase, end-to-end trainable model when one of the above training phases is skipped, and pretrained weights are utilised for the corresponding adapter (spatial or temporal), depending on the task or dataset. If spatial distributions are similar, only the temporal part (yellow box) needs training for (c); the blue part from (b) can be reused in figure 1.

Currently, SoTA methods efficiently aggregate spatio-temporal information but employ a unified framework for both dimensions, which can limit scalability when processing long video inputs. Thus, the first enhancement is to separate the trainable components for temporal and spatial dimensions within the adapters. This separation allows for more granular and adaptive handling of long-form videos, where temporal context varies significantly across action segments. While one may intuitively think that space-time separation could lead to a performance drop due to subop-

timal joint feature learning, we observe that performance significantly increases across challenging datasets. This is due to the unlocking of larger input dimensions during inference. More importantly, training is not entirely separate for each dimension—only the resolution of one component is reduced at each step to accommodate the other. Thus, the gains achieved through higher input resolution outweigh any potential performance drop from this separation, as shown by our improved performance. We show in section 4 that there is a drop of 0.5 mAP for the Thumos-14 dataset when this separation is introduced but a gain of 4.1 mAP using a larger input size.

Expanding on the new temporal module, this work enhances the temporal modelling process by introducing an aggregation module, which allows replacing the 1D temporal convolutional layers in previous work [16] with a transformer encoder. This replacement was impossible earlier due to resource constraints, highlighting the need for this work. While 1D convolutions are adequate for local temporal aggregation, they struggle to capture long-range dependencies as effectively as transformers, which excel in modelling global relationships. Although local processing via CNNs is effective in some instances, videos often contain large segments of redundant content, making the global modelling capability of transformers particularly useful in distinguishing meaningful action cues.

Our proposed approach is evaluated on several standard TAD datasets. By decoupling the training of temporal and spatial adapters and leveraging the power of our temporal adapters for temporal reasoning, we achieve state-of-the-art performance while also addressing memory efficiency. We also compare adapter arrangements. The pros and cons of each are discussed, and highlights are given on which arrangement should be used based on use cases or the need for resource efficiency or performance.

2. Related Work

The task of **TAD** has seen substantial progress in recent years, driven by advances in deep learning architectures, transformer-based modelling, and **PETL** techniques. Existing methods primarily fall into three categories: (1) one-stage and two-stage approaches, (2) query-based and transformer-based architectures, and (3) parameter-efficient learning techniques. We review these paradigms and position our work within the broader landscape.

2.1. Two-Stage and One-Stage Approaches

Two-stage approaches, such as BMN [14] and VSGN [31], first generate action proposals and then refine them using additional context-aware modules [1, 11, 27, 31, 33]. While these methods achieve good boundary localisation, they suffer from higher latency and memory overhead, making them impractical for long-form videos.

One-stage methods such as ActionFormer [30] and TriDet [21] simultaneously localize and classify actions in a single pass, leveraging dense feature extraction and anchor-free regression techniques. These methods emphasise end-to-end efficiency, reducing the computational burden typically associated with proposal generation [12, 14, 26] and perform well. However, they often struggle with long-range dependencies due to a small temporal window as input, leading to suboptimal action boundary localisation.

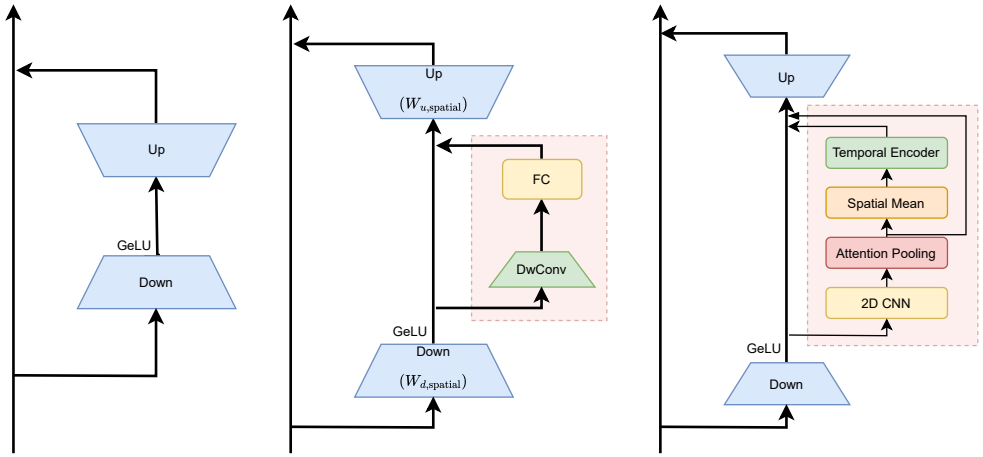
Our work builds upon these ideas by balancing efficiency and accuracy through adapter-based backbone feature refinement. We take inspiration from two-stage approaches and allow adapters to modify the backbone to attend to salient temporal parts (similar to action proposals) while using end-to-end training as in one-stage approaches.

2.2. Query-Based and Pure Transformer-Based Architectures

Recent advances in TAD have seen a growing interest in query-based and transformer-based architectures. Query-based approaches, such as TadTR [17] and DAB-DETR [15], employ learnable queries to predict action segments directly. These methods leverage transformer modules to model global context and improve localisation precision by replacing hand-designed anchors with adaptive query formulations. However, such systems often require high computational resources and memory, mainly when processing high-resolution or long-duration videos.

In parallel, pure transformer-based architectures in TAD, as demonstrated in works like TallFormer [5] and InternVideo [25], exploit self-attention to find action segments along with classification. While these methods achieve impressive performance, they have increased complexity and higher resource demands compared to the abovementioned methods. Most recent works like [7, 18, 22, 29] have addressed different limitations and have processed different datasets compared to us. For e.g., while we focus on the challenges highlighted in the previous section, [18] focuses on occlusions. [22] relies on pre-extracted features without finetuning the backbone, thus performing worse. Additionally, these works process video sequences with a maximum of 100 frames. In contrast, our method can handle up to 6144 frames with high spatial resolution, and even more with lower spatial resolution, which is critical for addressing long-duration videos, as demonstrated by our improved performance on a challenging dataset like EK-100.

Our approach draws inspiration from both paradigms while addressing their common limitations. Instead of using queries, we integrate transformer encoders within lightweight adapter modules attached to a frozen backbone to identify salient parts of the video. This strategy enables us to capture temporal relationships through self-attention.



(a) Standard Adapter (b) TIA (c) TIA++ (ours)

Figure 1. Architecture of (a) standard adapter, (b) temporal informative adapters (TIA) and our (c) TIA++ (ours). Different background colours for the respective parts represent the spatio-temporal separation. We improve upon the temporal part of TIA to support greater variation in the time domain of the input.

2.3. Parameter-Efficient Transfer Learning in Video Models

PETL techniques have become increasingly popular for adapting large-scale vision models to downstream tasks, e.g. TAD, video segmentation and visual grounding. Methods such as LoRA [10], AdaptFormer [3], and Conv-Adapter [2] demonstrate that finetuning only a small set of additional parameters can yield performance comparable to full finetuning while drastically reducing computational costs. In video action detection, AdaTAD [16] introduced TIA as a means to extend frozen vision backbones to input longer sequences. While TIA enables scalable TAD by reducing memory usage, it has two key limitations. First, it does not differentiate between spatial and temporal processing, leading to potential inefficiencies in handling high-resolution data. Second, it relies on simple temporal feature extraction, which restricts the model's ability to capture complex temporal dependencies.

Our proposed AdaTAD++ framework builds upon these insights by separating spatial and temporal adapters, allowing for independent optimisation and enabling higher-resolution processing. Furthermore, we significantly improve temporal reasoning while maintaining a practical memory footprint.

3. Methodology

Glossary:
TIA: Adapter from the work AdaTAD [16].
AdaTAD: Framework using TIA adapter.
TIA++: Adapter introduced by us.
AdaTAD++: Framework using our adapter.
(sp) / (te) / (sp-te): Indicates which part of the adapter is trained - spatial / temporal / both spatial and temporal. When not mentioned, both are trained.
S / P : Series / Parallel. When not mentioned, series is the default arrangement.

In this section, we introduce our AdaTAD++ step-by-step. We start by introducing the notations and architectures of the adapters. Then, we discuss the adapters' serial and parallel arrangements, followed by an alternate arrangement to minimise resource usage. Finally, we discuss our separable training protocol in comparison to single-stage learning. We also discuss the practical cases in which our protocol transforms into a single-stage training method in section 4.

3.1. Notations

Temporal action detection can be described as follows: given an untrimmed video $X \in \mathbb{R}^{3\times H\times W\times T}$, where H and W represent the height and width of each frame, and T denotes the total frame count, the temporal action annotations can be expressed as $\Psi_g = \{\phi_i = (t_s, t_e, c)\}_{i=1}^{N}$. Here, t_s and t_e are the start and end times, respectively, c is the action category for each instance ϕ_i, and N is the total number of ground-truth actions. Temporal action detection (TAD) aims to predict a set of candidate proposals $\Psi_p = \{\hat{\phi}_i = (\hat{t_s}, \hat{t_e}, c, \hat{s})\}_{i=1}^{M}$ that ideally cover Ψ_g, with $\hat{s}$ representing the confidence score.

In a general temporal action detection (TAD) approach using adapters, the input video $X \in \mathbb{R}^{3\times H\times W\times T}$ is divided into sequential chunks $\{X_j\}_{j=1}^{K}$, where each chunk $X_j \in \mathbb{R}^{3\times H\times W\times T_j}$ represents a segment of the video split across the time dimension T, and K is the number of chunks. Each chunk X_j is then passed through a frozen backbone network enhanced with trainable adapters, allow-

ing efficient adaptation without updating the primary backbone parameters. The resulting features from each chunk $\{F_j\}_{j=1}^{K}$ are concatenated to form a unified feature representation of the entire video. This concatenated feature representation is then fed into a detector module, which outputs predictions for each action instance $\hat{\phi}_i = (\hat{t_s}, \hat{t_e}, c, \hat{s})$, capturing the start time $\hat{t_s}$, end time $\hat{t_e}$, action category c, and confidence score $\hat{s}$.

3.2. Adapter Architectures

In TAD, adapters are strategically integrated into a frozen backbone network to adaptively refine video features without altering the backbone parameters. Here, we describe the specific methodology for each adapter type, highlighting their mathematical formulations and operational mechanisms.

3.2.1. Plain Adapters

Plain adapters, specifically Houlsby Adapters, are inserted in each block of the frozen backbone, one after the self-attention module and another after the layer-norm. Given an intermediate feature F, extracted from one input chunk X_j, each plain adapter module applies a bottleneck layer: first, a down-projection that reduces dimensionality, and second, an up-projection that restores the original dimension. Mathematically, this is expressed as:

$$F_{\text{adapted}} = F + W_u^\intercal f(W_d^\intercal F) \tag{1}$$

Where $W_d \in \mathbb{R}^{d \times d_r}$ and $W_u \in \mathbb{R}^{d_r \times d}$ are the trainable down- and up-projection matrices, d_r is the adapter's reduced dimension, and $f(\cdot)$ is a non-linear activation (e.g., GeLU). This formulation allows the adapter to capture task-specific nuances while keeping the structure of the backbone intact. The skip connection with F ensures that the adapted representation F_{adapted} remains aligned with the original features, enabling the model to converge faster and with fewer parameters. Figure 1(a) shows the architecture of these adapters.

3.2.2. TIA++ Adapters

Having established the background, our proposed TIA++ adapters extend the above idea and specifically focus on learning temporal interactions among chunks of the video $\{X_j\}_{j=1}^{K}$. Each adapter processes the features $\{F_k\}_{k=1}^{K}$ of all chunks $\{X_k\}_{k=1}^{K}$. This captures salient video parts, similar to anchors in query-based methods. Meanwhile, the backbone model only sees one chunk, X_j, at a time, and the outputs corresponding to each chunk are concatenated. This setting allows the backbone to have information about the entire input while processing one chunk at a time. This setup facilitates better local feature extraction while the detector focuses on global temporal processing.

TIA adapters [16] used a 3D convolution kernel with kernel size $(k, 1, 1)$ and group size d_r, followed by an FC layer with a weight matrix, $W_m \in \mathbb{R}^{d_r \times d_r}$, to learn cross-channel information. They add this in parallel to the connection from the downsampled part in equation 1 to the upsampling layer. Figure 1(b) shows the architecture of these adapters.

$$F_d = GeLU(W_d^\intercal F) \tag{2}$$

$$F_{temp} = W_m^\intercal \cdot DWConv(F_d) \tag{3}$$

$$F = F + \alpha \cdot (W_u(F_d + F_{temp})) \tag{4}$$

Here, α is used to scale the adapter's output.

We employ an alternative approach to refine temporal feature representations within the TAD task, introducing TIA++ adapters as illustrated in Figure 1(c). Transformers have been shown to outperform convolutions in extracting relations and excel at capturing information when it is scattered in the input. This is true in the case of TAD, as actions sometimes have salient parts that are not consecutive but spread out across time, especially for long and complex actions. But, as discussed in Section 1, the challenge here is to optimise resource requirements as transformers are known to be bulky. So, the more salient novelty is in the aggregation module before the transformer encoder in the temporal module, as seen in 1(c).

TIA++ adapters downsample the spatial part of the input heavily before processing the temporal part. The only concern is that the downsampled features should retain enough spatial information to distinguish between different temporal parts. So, initial processing in the temporal part of the adapters consists of a 2D pooling CNN with kernel size (k, k) and group size d_r which learns spatial information to retain before spatial pooling attention [24] is applied to the input. Then, the spatial part is reduced to one value per convolution kernel by averaging. It is then passed to a transformer encoder whose input has the same temporal resolution as the input. Mathematically, in equation 4, F_{temp} is replaced by:

$$F_{d'} = 2DCNN(F_d) \tag{5}$$

$$F_{d''} = F_{d'} * TransEnc_1(AveragePool_{2\times2}(F_{d'})) \tag{6}$$

$$\overline{F_{d''}} = \frac{1}{H * W} \sum_{h=0}^{H} \sum_{w=0}^{W} F_{d''} \tag{7}$$

$$F_{temp} = TransEnc_2(\overline{F_{d''}}) \tag{8}$$

Where H and W are the number of input patches along the height and width dimensions, $TransEnc$ represents a transformer encoder composed of self-attention and an FC layer. Both transformers are highly parameter-efficient due to the reduced feature dimensionality in the downsampled embedding within the adapter. Additionally, the input token count is minimised, representing less than a quarter of the spatial patch count and the number of tokens/chunks in

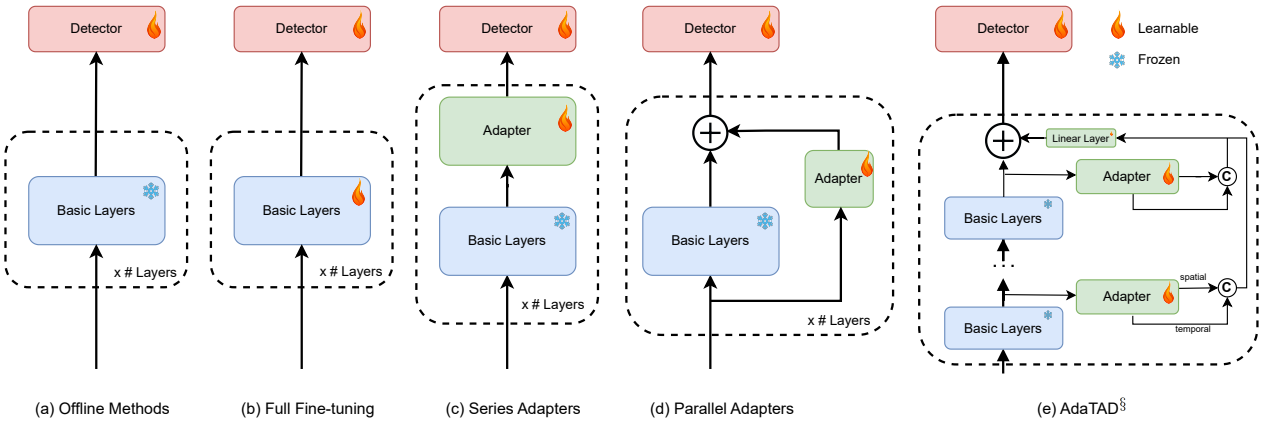


Figure 2. Comparison of various adapter positioning: (a) shows conventional offline methods, (b) represents typical end-to-end full-finetuning, (c) and (d) show series and parallel arrangements of adapters which is employed by TIA++ and represent by an S or P suffix in the text respectively, (e) shows an alternate arrangement used in AdaTAD++ for reducing memory requirements by removing flow of gradient from the basic layer.

time input for each transformer encoder in the model, respectively.

Looking at the big picture, by adding global information while processing local chunks, TIA++ adapters provide a robust means to encode temporal relationships, which is vital in capturing the structure of complex action sequences and improving the precision of temporal action detection.

3.3. Series and parallel adapters

In TAD, series and parallel adapters offer distinct methods for integrating additional trainable parameters, each with specific advantages.

Series adapters operate sequentially within the primary model layers, where each adapter processes the feature output from the previous layer before passing it to the next, as demonstrated in Figure 2(c). By processing inputs sequentially, series adapters allow for a deeper integration of learnt modifications within the primary network. This makes them beneficial for fine-tuning tasks that require detailed transformation across layers. However, the sequential nature can increase inference time, as each transformation must occur consecutively.

Parallel adapters, on the other hand, apply their transformations concurrently with the main layer processing. This parallel structure allows for quicker inference, as both main and adapter transformations co-occur (in Figure 2(d)). However, the adaptation here is often more constrained since it interacts less with the output of previous layers, which may limit the adapter's ability to make complex, layer-dependent adjustments.

The notations for the two variants used in this paper are AdaTad-S++ and AdaTad-P++ for series and parallel, respectively. In the next section, we show experiments for both arrangements to compare the two formally. This gives the readers a quantitative measure under different settings and this qualitative assessment.

3.4. Alternate Arrangement for the TIA++ Adapters

To further reduce resource requirements, TIA adapters have an alternate placement: instead of adding connections back to the main branch, they are directly added to the output of the backbone. This stops the gradient flow from the main branch and, thus, the overall resource requirement. However, their main problem is that the adapters are not complex enough to compute features that give comparable performance to the case where their output is added back to the main branch. As already discussed, TIA++ adapters have more complex temporal processing, and we modify them a bit to be used in a similar alternate manner. The upsampling layer is broken into two FC layers, taking one part for the spatial and the other for the temporal. This is shown in Figure 1(c). Figure 2 shows the addition of these adapters to the overall architecture.

We name the alternate arrangement of the adapters TIA§++ to distinguish between the two architectures. TIA§++ improves upon the lite version of TIA by adapting the architecture to directly add to the output instead of modifying the backbone. TIA adapters directly use the same adapters connected differently, which is sub-optimal.

3.5. Training Protocol: End-to-end vs. Separated training

As described in Section 1, separating the training of spatial and temporal adapters is essential to enable higher input resolutions across both dimensions. This independent training approach allows each adapter to specialise more effectively, optimising performance for spatial detail and temporal sequence recognition. The different parts are highlighted in Figure 1(b,c).

Table 1. Performance comparison on ActivityNet-1.3 and THUMOS14 using mAP at different IoU thresholds. Mem refers to memory usage (GB) per video. Results for variations of arrangements and training protocols are also added below. Results in blue are to be directly compared against the line above. Unless mentioned, series arrangement is used for adapters. *The results for all variations for* (1536×224^2) *without alternate arrangement or separating training for the adapters are achieved when only the last half of backbone layers are adapted; otherwise, full-layer adaptation will lead to out-of-memory on A100-80G. On ActivityNet-1.3, our prediction is combined with CUHK [6] classification results.* $*$ *means we employ stronger video-level classification results used in InternVideo.*

Method	Backbone	Mem	ActivityNet-1.3				THUMOS14					
			0.5	0.75	0.95	Avg.	0.3	0.4	0.5	0.6	0.7	Avg.
BMN [14]	TSN	-	50.07	34.78	8.29	33.85	56.0	47.4	38.8	29.7	20.5	38.5
TadTR [17]	I3D	-	49.10	32.60	8.50	32.00	62.4	57.4	49.2	37.8	26.3	46.6
ActionFormer [30]	SlowFast-R50	-	54.26	37.04	8.13	36.30	78.7	73.3	69.2	54.6	39.7	66.0
ActionFormer [30]	I3D	-	53.50	36.20	8.20	35.60	82.1	77.8	71.0	59.4	43.9	66.8
ASL [20]	I3D	-	54.10	67.40	8.00	36.20	83.1	79.0	71.7	59.7	45.8	67.9
TriDet [21]	I3D	-	54.70	38.00	8.40	36.80	83.6	80.1	72.9	62.4	47.4	69.3
VideoMAEv2 [23]	VideoMAEv2-g	-	-	-	-	-	-	-	-	-	-	69.6
InternVideo [25]	VideoMAE-H+UniformerV2	-	-	-	-	39.00*	-	-	-	-	-	71.5
AFSD [13]	I3D	12	52.40	35.30	6.50	34.40	67.3	62.4	55.5	43.7	31.1	52.0
E2E-TAD [17]	SlowFast-R50	12	50.47	35.99	10.33	34.10	69.4	64.3	56.0	46.4	34.9	54.2
BasicTAD [28]	SlowOnly-R50	12	51.20	33.41	7.57	33.12	75.5	70.8	63.5	50.9	37.4	59.2
TALLFormer [5]	VideoSwin-B	29	54.10	36.20	7.90	35.60	76.0	70.0	63.2	-	34.5	59.4
Re^{2}TAL [32]	Re^{2}VideoSwin-T	24	54.75	37.81	9.03	36.80	77.0	71.5	62.4	49.7	36.3	59.4
LoSA[8]	VideoMAEv2-G	40.6	58.5	39.8	7.8	38.6	85.0	81.1	74.5	65.1	49.3	71.0
ViT-TAD$_{160\times160}$ [29]	ViT-B	8.97	55.87	38.47	8.80	37.40	85.1	80.9	74.2	61.8	45.4	69.5
AdaTAD (768×224^2) [16]	SlowFast-R50	4.3	55.28	38.11	8.87	37.11	81.0	76.2	69.4	59.0	44.5	66.0
AdaTAD (768×224^2) [16]	VideoMAE-B	4.9	56.77	39.35	9.71	38.39	87.0	82.4	75.3	63.8	49.2	71.5
AdaTAD (768×224^2) [16]	VideoMAEv2-G	29.9	58.45	41.16	10.45	39.79	89.5	85.8	78.9	67.6	53.8	75.4
AdaTAD++ (768×224^2) (ours)	SlowFast-R50	7.1	56.28	39.94	9.92	38.03	83.7	78.7	71.6	60.4	44.9	67.5
AdaTAD++ (768×224^2) (ours)	VideoMAE-B	8.2	57.64	40.85	10.31	39.24	88.3	83.7	76.6	65.0	51.8	73.3
AdaTAD++ (768×224^2) (ours)	VideoMAEv2-G	38.5	60.28	43.21	12.05	41.01	89.7	86.4	79.8	69.6	55.9	76.3
Higher temporal resolution (1536×224^2)												
AdaTAD [16] (1536×224^2)	VideoMAEv2-G	50.6	61.72	43.35	10.85	41.93*	89.7	86.7	80.9	71.0	56.1	76.9
AdaTAD++ (1536×224^2) (ours)	VideoMAEv2-G	54.9	63.56	45.84	12.82	43.24*	90.0	87.7	82.6	73.2	59.5	78.6
Separating spatial and temporal training												
AdaTAD[16](sp-te) (1536×224^2)	VideoMAEv2-G	-	62.81	45.29	12.32	42.64*	89.8	87.0	81.3	71.7	57.3	77.4
AdaTAD(sp-te)++ (1536×224^2) (ours)	VideoMAEv2-G	-	64.03	**46.33**	**13.09**	**43.55***	**90.0**	**88.1**	**83.2**	**74.0**	**62.1**	**79.5**
Alternate efficient arrangements												
AdaTAD† [16] (1536×224^2)	VideoMAEv2-G	43.6	60.82	42.69	9.84	41.15*	89.6	85.9	79.4	67.6	53.8	75.8
AdaTAD§++ (1536×224^2) (ours)	VideoMAEv2-G	50.9	63.14	45.83	12.61	43.02*	89.9	87.4	82.0	72.5	58.4	78.0

Spatial Adapter Training Phase: Training begins with the spatial adapters, focusing on high-resolution spatial details. The input is configured as $X_j \in \mathbb{R}^{3\times H_{high}\times W_{high}\times T_{low}}$ to maximise spatial feature extraction while keeping the temporal resolution low to manage memory consumption. The spatially adapted feature representation for each chunk is computed as:

$$F_{j,\text{spatial-adapted}} = F_j + W_{u,\text{spatial}} \cdot g(W_{d,\text{spatial}} \cdot F_j)$$

where $W_{d,\text{spatial}}$ and $W_{u,\text{spatial}}$ are down- and up-projection matrices specifically for spatial adaptation, and $g(\cdot)$ represents a non-linear activation function to enhance spatial feature refinement.

Temporal Adapter Training Phase: After spatial features are refined, the focus shifts to the temporal adapters, and the input configuration changed to $X_j \in \mathbb{R}^{3\times H_{low}\times W_{low}\times T_{high}}$, prioritising temporal resolution. The temporally adapted feature for TIA adapters is computed as [1]:

$$F_{\text{temp}} = W_m \cdot \text{DWConv}(f(W_{d,\text{temp}} \cdot F_j))$$

Where $W_{d,\text{temp}}$ is the down-projection matrix for temporal adaptation, W_m is a weight matrix that enables cross-channel temporal feature learning, and DWConv represents a depthwise convolution. The non-linear activation function $f(\cdot)$ (e.g., ReLU) refines the temporal features, capturing essential temporal dependencies.

This two-phase protocol enables the model to prioritise spatial detail first, followed by temporal sequence learning, improving action detection performance with efficient memory use. A sample notation used in this work for this is AdaTAD(sp), AdaTAD(te) and AdaTAD(sp-te), represent-

[1] For ease of understanding, only the computation for TIA adapters is mentioned; TIA++ adapters follow the same protocol and should be easy to understand following this

ing spatial, temporal and both training, respectively. Table 3 shows two experiments with separable training.

4. Experiments

4.1. Datasets and Metrics

We use the ActivityNet-1.3, THUMOS14, and EPIC-Kitchens 100 datasets to assess our proposed method. ActivityNet-1.3 and THUMOS14 are untrimmed, third-person video datasets gathered from web sources, comprising 19,994 and 413 videos, respectively. In contrast, EPIC-Kitchens 100 includes 700 first-person videos, presenting a unique challenge due to its domain-specific action categories, which differ significantly from conventional pre-training data.

We report mean Average Precision (mAP) at specified IoU thresholds to maintain consistency with established evaluation methods. For ActivityNet-1.3, we use IoU thresholds from 0.5 to 0.95 in increments of 0.1. THUMOS14 employs thresholds of 0.3, 0.4, 0.5, 0.6, and 0.7, while EPIC-Kitchens 100 uses thresholds from 0.1 to 0.5.

Table 2. Results on EPIC-Kitchens 100 validation set.

Method	0.1	0.2	0.3	0.4	0.5	Avg.
Verb Task						
BMN [14]	10.8	8.8	8.4	7.1	5.6	8.4
G-TAD [27]	12.1	11.0	9.4	8.1	6.5	9.4
ActionFormer [30]	16.6	25.4	22.3	21.3	17.4	20.6
ASL [20]	27.9	-	25.5	-	19.8	-
TriDet [21]	32.7	26.1	24.2	21.8	20.5	25.1
AdaTAD (SlowFast-R50) [16]	34.5	27.3	26.1	24.4	22.8	27.4
ActionFormer (VideoMAE-L) [16]	31.7	31.6	29.1	26.7	23.6	28.5
AdaTAD (VideoMAE-L) [16]	33.1	32.2	30.4	27.5	25.1	29.3
AdaTAD(sp-te)-S++ (VideoMAE-L)	**35.1**	**34.2**	**32.9**	**30.0**	**27.9**	**32.0**
Noun Task						
BMN [14]	10.3	8.3	6.2	4.5	3.4	6.5
G-TAD [27]	12.4	10.8	7.0	5.4	4.1	7.9
ActionFormer [30]	25.2	21.4	20.2	17.6	15.2	20.0
ASL [20]	27.1	-	24.4	-	20.3	-
TriDet [21]	31.3	23.9	23.2	20.1	18.4	23.4
AdaTAD (SlowFast-R50) [16]	24.5	23.6	22.6	21.0	19.2	22.2
ActionFormer (VideoMAE-L) [16]	31.2	29.9	27.5	24.6	23.6	27.3
AdaTAD (VideoMAE-L) [16]	32.4	31.6	30.1	27.4	24.9	29.3
AdaTAD(sp-te)-S++ (VideoMAE-L)	**33.1**	**32.6**	**31.5**	**29.8**	**27.1**	**30.8**

4.2. Comparison with SoTA Methods

Table 1 shows comparison against state-of-the-art (SoTA) methods on ActivityNet-1.3 and Thumos14 datasets. AdaTAD++ has two main advantages, low memory consumption and large input size. We see a 75.4 to 76.3 increase in mAP compared to AdaTAD, the previous SoTA.

Higher temporal resolution: Concurrent with prior work, we see that an increase in input size has the maximum impact on performance. We observe a 76.3 to 78.6 increase with a larger input size (1536×224^2).

Separating temporal and spatial training: We see a further increase to 79.5 with separable training as we can add adapters to all blocks compared to adding to only half of the blocks due to resource constraints in the past, like in [16]. For a fair comparison, we applied the same training protocol to AdaTAD and show the results in the table.

Alternate efficient arrangements: We see an improvement for this setting too. It is discussed in detail in sec 4.3.2.

Table 2 presents our results on EPIC-Kitchens 100, a dataset with longer-duration videos that previously led all methods to rely exclusively on pre-extracted features. [16] was the first to introduce end-to-end training for this dataset. Owing to our contributions, we achieve better results for both verb and noun tasks. We show a gain of 2.7 and 1.5 for the two tasks.

4.3. Ablation and Analysis

In this section, unless mentioned otherwise, we use 768 frames as input, THUMOS14 as the dataset, VideoMAEv2-g as the backbone and serial arrangement for the adapters.

Table 3. Table to demonstrate how separable training benefits from high resolution during inference. Two experiments are shown here; the first two rows are training phases, and the third row shows inference results using these two training phases. First, high spatial and low temporal resolution is given as input and only the spatial part of the adapter is trained. In the second experiment, high temporal and low spatial resolution are used and only the temporal part is trained. During inference, no training is done, and high spatial and temporal resolutions are used along with trained spatial and temporal adapters taken directly from each adapter.

Method	Res.	Training		Mem.	mAP
		Spat.	Tem.		
AdaTAD(sp)-S	(384×224^2)	✓	×	18.7G(12.2)	69.4
AdaTAD(te)-S	(768×160^2)	×	✓	18.7G(6.5)	73.5
AdaTAD(sp-te)-S	(768×224^2)	×	×	-	74.9
AdaTAD(sp)-S	(768×224^2)	✓	×	29.9G(18.6)	75.4
AdaTAD(te)-S	(1536×160^2)	×	✓	29.9G(11.3)	76.0
AdaTAD(sp-te)-S	(1536×224^2)	×	×	-	77.9

4.3.1. The Advantage of Separable Training

We have established that increasing spatial and temporal resolution improves performance, although one-stage learning has inherent limitations in scaling up input size. Separable training offers an alternative by allowing for higher input resolutions during inference. Table 3 illustrates that models trained with either low spatial and high temporal resolution or low temporal and high spatial resolution individually perform poorly during training. However, combining spatial and temporal adaptations during inference at high resolutions achieves notably strong results. Although one-stage training is expected to perform better than separable training, it is constrained by resource availability. So, even though separable training falls slightly behind one-stage training, it is a valuable compromise as it allows a larger input. E.g., comparing results in table 1 and table 3, AdaTAD with an input size of 768×224^2 achieves 0.5 mAP lower under separable training than with one-stage training. One-stage training becomes infeasible for AdaTAD at

1536×224^2 resolution when adapters are added to all layers, while separable training not only makes this configuration possible but also yields improvements (0.5 mAP) over one-stage training when adapters are limited to half the layers to allow them to fit on the GPU (as done by [16]).

4.3.2. The Advantage of AdaTAD§++

There are two primary motivations behind this variation (shown in Figure 2). First, it allows us to scale input resolutions further by removing gradient flow through the backbone, thus reducing memory requirements. Additionally, based on the main idea from [19], it is important to note that transformers do not hierarchically extract features like CNNs do. Instead, they tend to generate features that are relatively more independent. Therefore, utilizing the outputs from each of the encoder blocks directly for the final output is an effective strategy that helps enhance the overall results. The difference between ours and the variation used in [16] is mainly the separation of the spatial and temporal outputs of each adapter and their concatenation instead of being projected into the feature space of the corresponding transformer encoder block. As we can see in Table 1 under the section "Alternate efficient arrangements", we improve the results from 75.8 for AdaTAD to 78.0 for us, thus proving the efficacy of our hypothesis. The reason for the improvement is that separating spatial and temporal features and concatenating them before projecting them into the feature space of the backbone allows for lesser loss of information (due to addition) compared to doing the two steps in the opposite order as in AdaTAD.

Table 4. Ablation of different adapter architectural designs. VideoMAE-B is used to conduct the following experiments.

Setting	Param.	Mem.	mAP	gains
Snippet Feature [16]	-	-	64.7	
+ Full FT	86M	5.6G	70.1	+ 5.1
+ LongLoRA [4]	28M	6.2G	71.1	+ 6.1
+ Standard Adapter [9]	3.6M	4.8G	70.2	+ 5.2
+ AdaTAD (w/o residual) [16]	4.0M	4.9G	70.8	+ 5.8
+ AdaTAD [16]	4.0M	4.9G	71.5	+ 6.5
+ AdaTAD++ (w/o separable training)	6.7M	8.2G	**73.3**	+ **8.3**

4.3.3. The Advantage of Adapter Design

Looking at Table 4, AdaTAD already improves on previous work, such as standard adapters and LongLORA, using less memory and having a high gain. While training less than 6.7M parameters (less in the case of separable training), we achieve a gain of +8.3.

4.3.4. Series vs Parallel Setting for AdaTAD and AdaTAD++

As discussed in Section 3, both series and parallel settings of the adapters have their advantages. We discuss these in Table 5. It can be seen that, as hypothesised, parallel variations require less memory than their series counterparts. We have shown results for various input resolutions to show the range of difference between the two. On the other hand, we can see that series adapters perform better than parallel adapters according to the intuition discussed in section 3. It is interesting to note that the performance difference between the two decreases as the input resolution increases. This is a favourable trend, as the training memory requirement will always scale proportionately to the input resolution, allowing the parallel arrangement to be more effective as input resolution increases.

Table 5. Comparison of series and parallel adapter arrangements. Input size is 768×224^2.

Settings	Series/ Parallel	Mem.	mAP
AdaTAD-S	series	29.9	**75.4**
AdaTAD-P	parallel	**26.2**	72.3
AdaTAD-S++	series	38.5	**76.3**
AdaTAD-P++	parallel	**33.7**	75.8

Table 6. Ablation results for spatial pooling CNN, spatial mean and attention pooling (shown in Figure 1(c)). OOM means out of memory, input resolution is 768×224^2.

Setting	Mean	CNN	Mem.	mAP
AdaTAD++ (w/o Att. Pool.)	✓	✓	34.3	73.8
AdaTAD++ (with Att. Pool.)	✓	×	37.7	74.9
	×	✓	OOM	-
	×	×	OOM	-
	✓	✓	38.5	**76.3**

4.3.5. Attention Pooling and Mean

Spatial pooling CNN and spatial mean (shown in Figure 1(c)) allow us to use transformer encoders instead of depthwise convolution in the adapters. Table 6 shows that spatial mean is necessary; otherwise, AdaTAD++ goes out of memory for A100-80GB GPU. There is a performance drop of 2.5 mAP when attention pooling is removed, and 1.4 mAP when spatial pooling CNN is removed. This shows that the two are effective additions in AdaTAD++.

5. Conclusion

In this paper, we present an optimized Temporal Action Detection (TAD) approach that enhances the TIA framework by decoupling temporal and spatial adapter training. Our two-phase protocol and use of a transformer encoder instead of 1D convolutions overcome scalability constraints and efficiently capture long-range dependencies with low memory usage. Extensive evaluation on TAD benchmarks demonstrates significant gains in accuracy and computational efficiency. The advancements demonstrated set a new standard for handling long video sequences and large-scale input data, underscoring the potential for our approach to support increasingly complex action detection applications.

References

[1] Yu-Wei Chao, Sudheendra Vijayanarasimhan, Bryan Seybold, David A Ross, Jia Deng, and Li Fei-Fei. Rethinking the faster r-cnn architecture for temporal action localization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 2

[2] Hao Chen, Ran Tao, Han Zhang, Yidong Wang, Xiang Li, Wei Ye, Jindong Wang, Guosheng Hu, and Marios Savvides. Conv-adapter: Exploring parameter efficient transfer learning for convnets. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1551–1561, 2024. 3

[3] Shoufa Chen, Chongjian Ge, Zhan Tong, Jiangliu Wang, Yibing Song, Jue Wang, and Ping Luo. Adaptformer: Adapting vision transformers for scalable visual recognition. *Advances in Neural Information Processing Systems*, 35:16664–16678, 2022. 3

[4] Yukang Chen, Shengju Qian, Haotian Tang, Xin Lai, Zhijian Liu, Song Han, and Jiaya Jia. Longlora: Efficient fine-tuning of long-context large language models. *arXiv preprint arXiv:2309.12307*, 2023. 8

[5] Feng Cheng and Gedas Bertasius. Tallformer: Temporal action localization with a long-memory transformer. In *European Conference on Computer Vision*, pages 503–521. Springer, 2022. 2, 6

[6] Bernard Ghanem, Juan Carlos Niebles, Cees Snoek, Fabian Caba Heilbron, Humam Alwassel, Ranjay Khrisna, Victor Escorcia, Kenji Hata, and Shyamal Buch. Activitynet challenge 2017 summary. *arXiv preprint arXiv:1710.08011*, 2017. 6

[7] Alexey A. Gritsenko, Xuehan Xiong, Josip Djolonga, Mostafa Dehghani, Chen Sun, Mario Lucic, Cordelia Schmid, and Anurag Arnab. End-to-end spatio-temporal action localisation with video transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 18373–18383, 2024. 2

[8] Akshita Gupta, Gaurav Mittal, Ahmed Magooda, Ye Yu, Graham Taylor, and Mei Chen. LoSA: Long-short-range adapter for scaling end-to-end temporal action localization. In *Proceedings of the Winter Conference on Applications of Computer Vision (WACV)*, pages 2092–2102, 2025. 6

[9] Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for nlp. In *International conference on machine learning*, pages 2790–2799. PMLR, 2019. 8

[10] Edward J Hu, Yelong Shen, Patrick Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Sining Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2021. 3

[11] Matthew Korban, Peter Youngs, and Scott T. Acton. A semantic and motion-aware spatiotemporal transformer network for action detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46(9):6055–6069, 2024. 2

[12] Chuming Lin, Jian Li, Yabiao Wang, Ying Tai, Donghao Luo, Zhipeng Cui, Chengjie Wang, Jilin Li, Feiyue Huang, and Rongrong Ji. Fast learning of temporal action proposal via dense boundary generator. In *Proceedings of the AAAI conference on artificial intelligence*, pages 11499–11506, 2020. 2

[13] Chuming Lin, Chengming Xu, Donghao Luo, Yabiao Wang, Ying Tai, Chengjie Wang, Jilin Li, Feiyue Huang, and Yanwei Fu. Learning salient boundary feature for anchor-free temporal action localization. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 3320–3329, 2021. 6

[14] Tianwei Lin, Xiao Liu, Xin Li, Errui Ding, and Shilei Wen. Ben: Boundary-matching network for temporal action proposal generation. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 3889–3898, 2019. 2, 6, 7

[15] Shilong Liu, Feng Li, Hao Zhang, Xiao Yang, Xianbiao Qi, Hang Su, Jun Zhu, and Lei Zhang. DAB-DETR: Dynamic anchor boxes are better queries for DETR. In *International Conference on Learning Representations*, 2022. 2

[16] Shuming Liu, Chen-Lin Zhang, Chen Zhao, and Bernard Ghanem. End-to-end temporal action detection with 1b parameters across 1000 frames. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 18591–18601, 2024. 1, 2, 3, 4, 6, 7, 8

[17] Xiaolong Liu, Qimeng Wang, Yao Hu, Xu Tang, Shiwei Zhang, Song Bai, and Xiang Bai. End-to-end temporal action detection with transformer. *IEEE Transactions on Image Processing*, 31:5427–5441, 2022. 2, 6

[18] Rajat Modi, Vibhav Vineet, and Yogesh S. Rawat. On occlusions in video action detection: Benchmark datasets and training recipes. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024. 2

[19] Dominick Reilly and Srijan Das. Just add?! pose induced video transformers for understanding activities of daily living. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18340–18350, 2024. 8

[20] Jiayi Shao, Xiaohan Wang, Ruijie Quan, Junjun Zheng, Jiang Yang, and Yi Yang. Action sensitivity learning for temporal action localization. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 13457–13469, 2023. 6, 7

[21] Dingfeng Shi, Yujie Zhong, Qiong Cao, Lin Ma, Jia Li, and Dacheng Tao. Tridet: Temporal action detection with relative boundary modeling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18857–18866, 2023. 1, 2, 6, 7

[22] Yepeng Tang, Weining Wang, Chunjie Zhang, Jing Liu, and Yao Zhao. Learnable feature augmentation framework for temporal action localization. *IEEE Transactions on Image Processing*, 2024. 2

[23] Limin Wang, Bingkun Huang, Zhiyu Zhao, Zhan Tong, Yinan He, Yi Wang, Yali Wang, and Yu Qiao. Videomae v2: Scaling video masked autoencoders with dual masking. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14549–14560, 2023. 1, 6

[24] Yong Wang, Han Li, and Soo Kim. Spatial pooling attention for efficient feature selection in vision tasks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43(4): 1056–1068, 2021. 4

[25] Yi Wang, Kunchang Li, Yizhuo Li, Yinan He, Bingkun Huang, Zhiyu Zhao, Hongjie Zhang, Jilan Xu, Yi Liu, Zun Wang, et al. Internvideo: General video foundation models via generative and discriminative learning. *arXiv preprint arXiv:2212.03191*, 2022. 2, 6

[26] Zhenzhi Wang, Ziteng Gao, Limin Wang, Zhifeng Li, and Gangshan Wu. Boundary-aware cascade networks for temporal action segmentation. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXV 16*, pages 34–51. Springer, 2020. 2

[27] Mengmeng Xu, Chen Zhao, David S Rojas, Ali Thabet, and Bernard Ghanem. G-tad: Sub-graph localization for temporal action detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10156–10165, 2020. 2, 7

[28] Min Yang, Guo Chen, Yin-Dong Zheng, Tong Lu, and Limin Wang. Basictad: an astounding rgb-only baseline for temporal action detection. *Computer Vision and Image Understanding*, 232:103692, 2023. 6

[29] Min Yang, Huan Gao, Ping Guo, and Limin Wang. Adapting short-term transformers for action detection in untrimmed videos. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 18570–18579, 2024. 2, 6

[30] Chen-Lin Zhang, Jianxin Wu, and Yin Li. Actionformer: Localizing moments of actions with transformers. In *European Conference on Computer Vision*, pages 492–510. Springer, 2022. 1, 2, 6, 7

[31] Chen Zhao, Ali K Thabet, and Bernard Ghanem. Video self-stitching graph network for temporal action localization. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 13658–13667, 2021. 2

[32] Chen Zhao, Shuming Liu, Karttikeya Mangalam, and Bernard Ghanem. Re2tal: Rewiring pretrained video backbones for reversible temporal action localization. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10637–10647, 2023. 6

[33] Peisen Zhao, Lingxi Xie, Chen Ju, Ya Zhang, Yanfeng Wang, and Qi Tian. Bottom-up temporal action localization with mutual regularization. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part VIII 16*, pages 539–555. Springer, 2020. 2