End-to-End Temporal Action Detection with 1B Parameters Across 1000 Frames

Shuming Liu[1] Chen-Lin Zhang[2] Chen Zhao[1]* Bernard Ghanem[1]
[1]King Abdullah University of Science and Technology (KAUST) [2]4Paradigm Inc

Abstract

*Recently, temporal action detection (TAD) has seen significant performance improvement with end-to-end training. However, due to the memory bottleneck, only models with limited scales and limited data volumes can afford end-to-end training, which inevitably restricts TAD performance. In this paper, we reduce the memory consumption for end-to-end training, and manage to scale up the TAD backbone to **1 billion parameters** and the input video to **1,536 frames**, leading to significant detection performance. The key to our approach lies in our proposed temporal-informative adapter (TIA), which is a novel lightweight module that reduces training memory. Using TIA, we free the humongous backbone from learning to adapt to the TAD task by only updating the parameters in TIA. TIA also leads to better TAD representation by temporally aggregating context from adjacent frames throughout the backbone. We evaluate our model across four representative datasets. Owing to our efficient design, we are able to train end-to-end on VideoMAEv2-giant and achieve 75.4% mAP on THUMOS14, being the first end-to-end model to outperform the best feature-based methods. Code is available at* https://github.com/sming256/AdaTAD.

1. Introduction

Temporal Action Detection (TAD) plays a vital role in the understanding of long-form videos. Its objective is to pinpoint specific action instances within untrimmed videos, identifying their start and end times, along with their respective categories [15, 20, 44, 60, 62, 67, 71]. This task is crucial for various applications, including highlight detection [36, 64], video-language grounding [38, 48], and action spotting [3]. Though innovations in the detector design have made profound progress in the past years [30, 66, 67], recent research highlights two new trends: *end-to-end training* [10, 32, 69], and *scaling up* [56, 57].

End-to-end training in TAD refers to jointly training the video encoder and action detector [10, 29, 62, 70]. Compared to feature-based methods, end-to-end training offers distinct advantages. First, it can effectively bridge the gap commonly found between pretraining and fine-tuning, such as data and task discrepancy. Second, video spatial augmentations can be utilized in the end-to-end setting, leading to further performance gain.

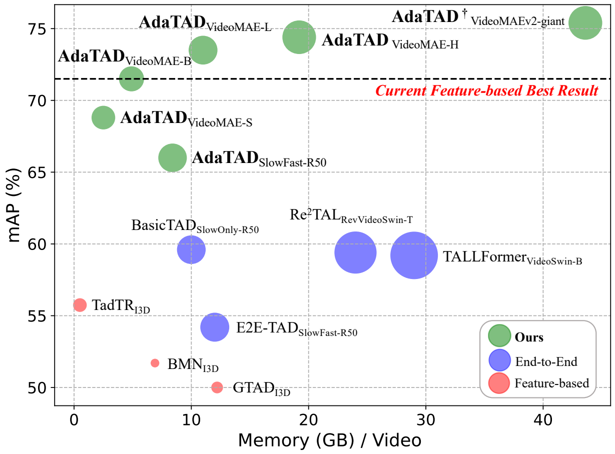


Figure 1. **AdaTAD enjoys the benefit of end-to-end training and scaling up with efficient memory usage.** The bubble size represents the number of the model's learnable parameters. Using SlowFast-R50, AdaTAD achieves better performance compared to E2E-TAD [33] with less memory. When further scaling up the model to VideoMAEv2-gaint and data to 1536 frames, we achieve an impressive avg. mAP of 75.4% on THUMOS14.

Scaling up refers to improving performance by increasing the model size or the input data volume, and has demonstrated its effectiveness in various domains [14, 19, 39, 53]. In TAD, offline methods have attempted to scale up the feature extraction network to reach a higher performance. A notable example includes the work by Wang *et al.*[56], which reports a 10% increase in mean Average Precision (mAP) by scaling from VideoMAE-S to VideoMAEv2-giant, using ActionFormer [66] on THUMOS14 [25].

Intuitively, combining the strengths of end-to-end training and scaling up is expected to be most beneficial for improving TAD performance. However, both strategies demand substantial GPU memory, which restricts end-to-end training to a small model [29, 62, 69], or a small input volume [10, 33]. As shown in Fig. 1, the performance of previous end-to-end methods still significantly lags behind the

[*] Corresponding author.

best results achieved by feature-based approaches. Additionally, current end-to-end methods in TAD use computationally intensive full fine-tuning, risking the issues of catastrophic forgetting and overfitting during transfer learning [50, 63]. These issues can result in less competitive performance, especially when the downstream datasets are small, which is a common scenario in the TAD domain.

In this paper, we aim to overcome the above limitations by harnessing the advantages of both scaling-up and end-to-end training. To achieve this, we introduce adapter tuning for temporal action detection (**AdaTAD**). Our method successfully trains a TAD model in an end-to-end manner, utilizing a backbone with 1 billion parameters and processing input videos of 1,536 frames. As illustrated in Fig. 1, to the best of our knowledge, this is the first end-to-end work that outperforms the best feature-based TAD methods.

Specifically, we employ the following strategies to enhance the TAD performance while maintaining reasonable memory consumption. **First**, we identify that the snippet representation commonly used in feature-based methods is excessively redundant. In response, we have adopted a more memory-efficient frame-representation scheme, establishing an effective end-to-end baseline for TAD. **Second**, we adopt the parameter-efficient fine-tuning technique to minimize memory usage and mitigate overfitting in transfer learning. Notably, we introduce a novel Temporal-Informative Adapter (TIA). This adapter is injected between backbone layers and is the only learnable component in the backbone during fine-tuning. Different from conventional adapters [22], TIA is tailored for the TAD task and integrates temporal depth-wise convolutions to aggregate informative context from adjacent frames. **Third**, for additional memory efficiency, we propose a lighter variant of our method. By positioning the TIAs alongside the original backbone, rather than inside it, backpropagation can be done through the TIAs only. This configuration can further cut down on the need for intermediate activations within the primary backbone, thereby allowing us to scale up the model size and data size to unprecedented levels.

AdaTAD establishes a new state-of-the-art across multiple TAD datasets. Notably, our method achieves an impressive 75.4% mAP on THUMOS14, outperforming the previous feature-based best result of 71.5% by a large margin. This achievement underscores the possible paradigm shift in TAD, *i.e.*, moving from traditional feature extraction plus offline detectors to scaling up end-to-end TAD training. We summarize our contribution as follows:

1. We introduce an efficient end-to-end framework for TAD, scaling up the model size to 1 billion parameters and the input data to 1,536 frames. We achieve a consistent performance improvement with the scaling up, shedding light on the importance of scaling for TAD.
2. We propose a novel temporal-informative adapter to reduce memory as well as aggregate the temporal context for TAD. Different variants of these adapters are designed to trade off between performance and memory cost. To the best of our knowledge, we are the first to introduce the adapter mechanism to TAD.
3. Our method achieves state-of-the-art performance across four TAD datasets. Remarkably, this represents the first end-to-end approach that outperforms the previous feature-based methods by a large margin.

2. Related Work

Temporal Action Detection. Current methods for temporal action detection, also referred to as temporal action localization, can be broadly classified into three categories based on their architectural design: one-stage, two-stage, and DETR-based methods. One-stage methods directly localize actions from a multi-scale feature pyramid, such as ActionFormer [66] and TriDet [47]. These methods integrate action classification and temporal boundary regression in a single step [45, 61, 62]. Two-stage methods, in contrast, involve an additional step of proposal feature extraction [4, 30, 40, 58, 59, 68, 73]. For instance, VSGN [67] employs boundary sampling to further refine proposals. Recently, there is a growing interest in query-based methods [34, 46, 51], which deploy a set of learned queries to interact with the feature maps and directly predict the actions' temporal boundaries and categories.

In addition to the aforementioned categories, TAD can also be divided into feature-based and end-to-end methods. The former relies on pre-extracted RGB features and optionally incorporates optical flow features. On the other hand, end-to-end methods take raw video frames as input and jointly optimize the video encoder and action detector [31]. Due to computational constraints, AFSD [29] downsamples the input to a resolution of 96^2. DaoTAD [55] and E2E-TAD [33] provide evidence that high TAD performance can be achieved by relying solely on the RGB modality with various data augmentations. Further innovations came from SGP [11], TALLFormer [10], and ETAD [32], all of which introduced strategies to backpropagate only through parts of the data. Additionally, Re^{2}TAL [69] and Dr^{2}Net[70] design reversible network architectures to release the memory occupied by intermediate activations. Despite these advancements, all above methods follow the full fine-tuning paradigm, and none has yet surpassed the best results achieved by feature-based approaches.

Scaling Law in Deep Learning. Scaling up model and data has been a prevalent strategy across both computer vision and natural language processing fields to achieve superior performance. The GPT series [5, 39, 41, 42] has consistently demonstrated that larger models pretrained on extensive datasets yield significant gains in language understanding capabilities. Analogously, in the realms of image

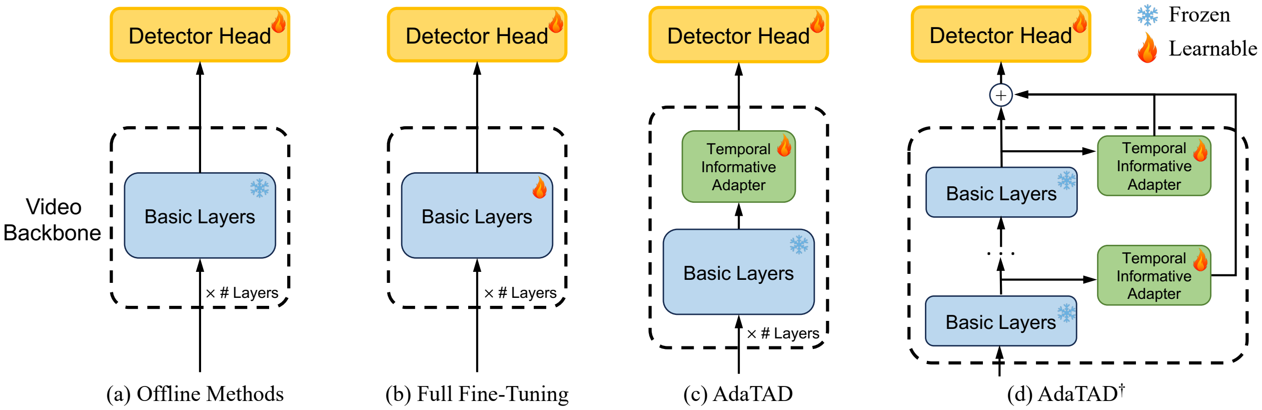


Figure 2. **Comparative illustration of our proposed TAD framework versus popular and widely used alternatives.** (a) represents the typical offline method. (b) is the traditional end-to-end method using full fine-tuning. (c) Tailored for the TAD task, our AdaTAD uses a lightweight temporal-informative adapter inside the backbone to achieve efficient transfer learning. (d) To further reduce memory usage and scale up the model/data, AdaTAD† uses an alternative placement for adapters outside the backbone.

and video understanding, architectures such as ViT [14] and MViT [16], have also witnessed the effectiveness of this scaling strategy. Alabdulmohsin *et al.*[1] present a recipe for estimating scaling law parameters reliably from learning curves in computer vision. To attain even greater performance, several studies have also scaled up image resolution [35] or video clip length [56]. In this paper, we successfully apply this principle within the domain of temporal action detection and achieve state-of-the-art results.

Efficient Transfer Learning. Transfer learning aims to adapt a pretrained model to a new domain. In TAD, typically off-the-shelf action recognition models are employed as the backbone, such as SlowFast [17]. Traditional transfer learning adopts full fine-tuning, meaning that all parameters of the pretrained model are updated. However, studies by [43, 63] have noted that full fine-tuning may harm the pre-leared knowledge, particularly when the downstream dataset is small and less comprehensive. Moreover, as models increase in size, the computational and storage demands of full fine-tuning proportionally increase.

In response to these challenges, several works have investigated parameter-efficient tuning (PEFT) strategies that involve fine-tuning only a fraction of the network. For instance, Adapter [22] inserts lightweight modules analogous to feedforward networks in transformers, and only tunes these elements. LoRA [23] employs low-rank matrices in each transformer layer. Prefix-tuning [28] and prompt-tuning [27] attach learnable prompt tokens at the input stage or within each layer. In computer vision, many PEFT methods [6, 7, 24, 63] have also been explored across various tasks to optimize transfer learning efficiency. Our work represents the first effort to examine the potential of the PEFT mechanism in TAD.

Although PEFT effectively reduces the number of learnable parameters, data-intensive and computationally heavy tasks like video understanding require more memory-efficient techniques. To this end, several works try to externalize the trainable components from the backbone, eliminating the need for backpropagation through the extensive original model. For example, LST [49] introduces a supplementary lightweight network that operates in parallel with the main model. Similarly, E^{3}VA [65] leverages intermediate features with adapters to enable efficient transfer learning while minimizing memory usage. Our work is inspired by these methods yet with a streamlined and simple design.

3. Methodology

In this section, we introduce our AdaTAD step-by-step. We first introduce notations and study the efficient video representation to establish an end-to-end TAD baseline. Next, we introduce a temporal-informative adapter designed for efficient TAD. Finally, we propose an alternative placement for adapters to further alleviate computational demands.

3.1. Notations

Temporal action detection can be formulated as follows: given an untrimmed video $\mathbf{X} \in \mathbb{R}^{3\times H\times W\times T}$, where H and W are the height and width of each frame, and T is the frame number, its temporal action annotations can be denoted as $\Psi_g = \{\varphi_i = (t_s, t_e, c)\}_{i=1}^{N}$, where t_s, t_e, c are the start, end time and category of action instance φ_i, and N is the number of ground truth actions. TAD aims to predict candidate proposal set $\Psi_p = \left\{\hat{\varphi}_i = (\hat{t}_s, \hat{t}_e, \hat{c}, s)\right\}_{i=1}^{M}$ to cover Ψ_g, and s is the confidence score.

3.2. Frame-level representation

Our end-to-end TAD architecture comprises two main components: feature extraction and action detection. Following previous work [69], we select ActionFormer [66] as our action detection head due to its robust performance across various datasets without much hyperparameter tuning. Next, we discuss two ways of encoding raw frames into representative features (feature extraction): snippet representation and frame representation.

Snippet Representation. Snippet-based video representations are popular choices in offline feature extraction. The whole video is divided into several short snippets (or namely clips). Each snippet has a short temporal length, *e.g.*, 16 frames, and different snippets can have overlapping frames. Thus, the video can be conceptualized as T snippets, denoted by $\mathbf{X} \in \mathbb{R}^{T\times 3\times 16\times H\times W}$. Each snippet is processed through the video backbone, and spatial-temporal pooling is applied to extract one snippet feature. This processing yields the feature representation $\mathbf{F} \in \mathbb{R}^{T\times C}$, where C denotes the channel dimension of the pooled features.

Frame Representation. In contrast to snippet-based representations, frame-based video representations consider the entire video as a singular snippet or clip, represented as $\mathbf{X} \in \mathbb{R}^{1\times 3\times T\times H\times W}$. Then, the whole frame sequence is fed into the video backbone, and only spatial pooling is employed to extract features [10, 29, 69]. For attention-based models such as VideoMAE [54], the video is chunked into multiple shorter clips to avoid extensive temporal attention.

Although both representations have been employed in previous studies, a fair comparison between them has not yet been performed. To address this gap, we conduct a comparative analysis of the two representations under the same setting on THUMOS14, measuring their memory usage and detection mAP. The results in Table 1 indicate that **frame representation has comparable or even better performance than snippet representation**, yet with much smaller memory consumption. When the feature extraction backbones are frozen, frame representation yields superior results to snippet representation for both VideoMAE [53] and SlowFast [17] backbones. Only in the end-to-end setting can the snippet representation achieve 1% mAP advantage over frame representations; however, it requires 8 times more memory consumption. Taking into account both performance and memory usage, frame-based representations could be a better choice for end-to-end TAD development.

Table 1. **Snippet representation *vs* frame representation.** We use the end-to-end version of ActionFormer with two representations for comparison. The snippet input is 768×3×16×160×160, and the frame input is 1×3×768×160×160. $*$ means activation checkpointing is utilized to avoid overflowing GPU memory.

Setting	Backbone	Repr.	Avg. mAP	Mem (GB)
Frozen	VideoMAE-S	Frame	59.35	1.9
		Snippet	57.68	13.2
	SlowFast-R101	Frame	61.34	3.6
		Snippet	60.24	17.2
End to End	VideoMAE-S	Frame	67.15	2.8*
		Snippet	68.46	24.6*
	SlowFast-R101	Frame	65.33	5.5*
		Snippet	66.72	51.6*

Therefore, we use frame representation as the default baseline to encode videos in our experiments. Following the previous TALLFormer work [10], we also incorporate activation checkpointing [8] and mixed precision training [37] to fully harness the potential of scaling.

3.3. Temporal-Informative Adapter

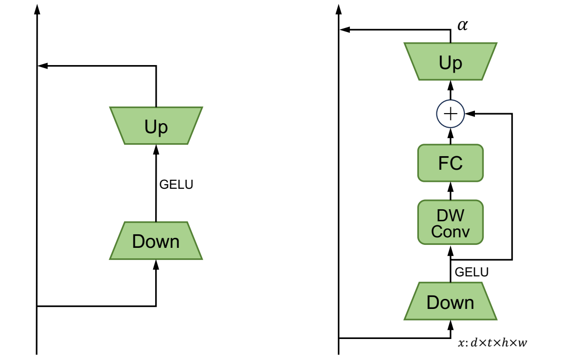
In Section 3.2, we have built a simple end-to-end baseline using full fine-tuning. However, the baseline still suffers from two aspects: **1. Increased computational cost.** In Table 1, we only use small video backbones like VideoMAE-S. When scaling VideoMAE-S to larger models, the computational burden and memory cost will grow rapidly. **2. Inferior transfer learning ability.** More critically, the baseline follows the tradition of full fine-tuning, which may lead to inferior transfer learning. Pointed out by [50, 63], full fine-tuning may result in overfitting or forgetting, especially for large pretrained models. If downstream datasets are not sufficiently diverse, full fine-tuning can even destroy the powerful features learned from large-scale pretraining. Motivated by the above two aspects, we apply the PEFT mechanism and propose to fine-tune a plug-and-play module named **Temporal-Informative Adapter (TIA)** to achieve efficient and effective transfer learning for TAD.

We first review the architecture of the standard adapter proposed by [22]. As formulated in Equation 1, the standard adapter includes a down-projection fully connected (FC) layer with parameter $\boldsymbol{W}_{\text{down}} \in \mathbb{R}^{d\times\frac{d}{\gamma}}$, where $\frac{d}{\gamma}$ represents the intermediate dimension and satisfies $\gamma > 1$. Then, an up-projection layer $\boldsymbol{W}_{\text{up}} \in \mathbb{R}^{\frac{d}{\gamma}\times d}$ is employed to restore the channel dimension. Between these two FC layers, a non-linear activation function $\boldsymbol{\sigma}$ is inserted, such as GELU [21]. Afterward, a residual connection is added to the output of the projection layer. Note that $\boldsymbol{x}$ and $\boldsymbol{x'}$ are the input and output features with the same shape $\mathbb{R}^{d\times t\times h\times w}$.

$$\boldsymbol{x'} = \boldsymbol{W}_{\text{up}}^{\top} \cdot \boldsymbol{\sigma}(\boldsymbol{W}_{\text{down}}^{\top} \cdot \boldsymbol{x}) + \boldsymbol{x}. \tag{1}$$

Although the adapter has achieved great success in natural language processing and computer vision, the standard adapter only focuses on adapting channel information, which neglects the temporal context vital for the TAD task. To address this limitation, we introduce the temporal-informative adapter, as depicted in Fig. 3(b).

The architecture of TIA follows the general bottleneck design of the standard adapter, while we integrate the temporal depth-wise convolution layers, as described in Equation 2. The temporal convolution with a kernel size of k is



(a) Standard Adapter (b) Temporal-Informative Adapter

Figure 3. **Architecture of (a) standard adapter and (b) our temporal-informative adapter**. We incorporate temporal depth-wise convolution to aggregate context from adjacent frames.

designed to aggregate local informative context from adjacent frames and to enrich the representation of the current time step. Practically, this is achieved through the application of a 3D convolution with kernel size $(k,1,1)$ and group size $\frac{d}{\gamma}$ for depth-wise processing. Additionally, an FC layer with weight $\boldsymbol{W}_{\text{mid}} \in \mathbb{R}^{\frac{d}{\gamma}\times\frac{d}{\gamma}}$ is employed to facilitate information exchange across channels. At last, a learnable scalar α is introduced to adjust the amplitude of adapter's output.

$$\begin{aligned} \bar{\boldsymbol{x}} &= \boldsymbol{\sigma}(\boldsymbol{W}_{\text{down}}^{\top} \cdot \boldsymbol{x}), \\ \hat{\boldsymbol{x}} &= \boldsymbol{W}_{\text{mid}}^{\top} \cdot \textbf{DWConv}_k(\bar{\boldsymbol{x}}) + \bar{\boldsymbol{x}}, \\ \boldsymbol{x}' &= \alpha \cdot \boldsymbol{W}_{\text{up}}^{\top} \cdot \hat{\boldsymbol{x}} + \boldsymbol{x}. \end{aligned} \tag{2}$$

As shown in Fig. 2(c), TIA is designed to be inserted between different backbone layers, *e.g.* between each ViT block of VideoMAE or each bottleneck block of SlowFast. To ensure the newly added connection does not affect the original network at the beginning of transfer learning, the weight and bias of the adapter's last projection layer $\boldsymbol{W}_{\text{up}}$ are initialized to 0. The learnable coefficient α is initialized to 1. The temporal kernel size k is set to 3, and the channel downsampling ratio γ is set to 4 by default. Under these settings, the additional trainable parameters coming from TIA only account for 4.7% of the total parameters of the original backbone. Since this backbone is frozen when TIA is used, our proposed strategy constitutes a massive reduction in trainable parameters as compared to full fine-tuning. Our experiments show that TIA can achieve better performance than full fine-tuning with less memory usage.

3.4. Alternative Placement for Adapter

Although the previously described PEFT approach can reduce tunable parameters and memory usage, the gradient still needs to backpropagate over the entire backbone during training. This requirement limits our ability to scale-up the model size or input data size further. As highlighted in prior works [49, 65], if we can stop the gradient backpropagation within the original backbone, additional computational savings can be achieved.

Inspired by this insight, we propose a placement strategy for adapters that position them externally to the backbone, rather than inserting them inside. As illustrated in Fig. 2(d), we utilize the previously introduced TIA module, but its output does not feed back into the middle of the original backbone. It is directly added to the backbone's final layer. This configuration eliminates the need for backpropagation through the original network, as gradients are only tracked to the shallow lightweight adapter. To further diminish computation, we observe that adapting only the last half of backbone layers yields comparable performance while reducing half of the adaptation cost.

To distinguish the different variants, we name the standard adaption design as AdaTAD, and the alternative placement as AdaTAD†. The latter can be considered as a lite version of the former. Compared to directly injecting adapters into the backbone, AdaTAD† may lead to a slight performance drop. However, it enables us to leverage richer models and more data, which should effectively counter this possible drop.

4. Experiments

4.1. Datasets and Metrics

We choose ActivityNet-1.3 [20], THUMOS14 [25], and Epic-Kitchens 100 [13] to evaluate our proposed approach. ActivityNet-1.3 and THUMOS14 are web-collected third-person untrimmed videos, consisting of 19,994 and 413 videos, respectively. EPIC-Kitchens 100 is collected from 700 egocentric videos. Since the action categories of EPIC-Kitchens 100 are more domain-specific and different from common pretraining data, achieving higher performance on this dataset is more challenging. Moreover, we also evaluate our method on the Ego4D-MQ [18] dataset, and the results can be found in the appendix.

Following common practice, we report the mean Average Precision (mAP) at certain IoU thresholds and average mAP as the evaluation metrics. On ActivityNet-1.3, the IoU thresholds are chosen from 0.5 to 0.95 with 10 steps. On THUMOS14, the threshold is chosen from $\{0.3, 0.4, 0.5, 0.6, 0.7\}$. On EPIC-Kitchens 100, the threshold is set to $\{0.1, 0.2, 0.3, 0.4, 0.5\}$.

4.2. Implementation Details

We implement our method with PyTorch 2.0 and MMAction2 [12] with 4 A100 GPUs. By default, mixed-precision training and activation checkpointing are adopted to save memory. We use ActionFormer [66] as our detection head, and keep the hyper-parameters unchanged on each dataset. The learning rate for the adapter in the backbone is grid-

Table 2. **Results on ActivityNet-1.3 and THUMOS14**, measured by mAP (%) at different tIoU thresholds. E2E refers to end-to-end training, and Mem refers to memory usage (GB) per video. On ActivityNet-1.3, our prediction is combined with CUHK [72] classification results. Specifically, $*$ means we employ stronger video-level classification results used in InternVideo [57] for a fair comparison. We report our best results in **bold**, and the previous best results in underline, which was achieved by the feature-based method. The last row is achieved when only the last half of backbone layers are adapted; otherwise, full-layer adaptation will lead to out-of-memory on A100-80G.

Method	Backbone	E2E	Flow	Mem	ActivityNet-1.3				THUMOS14					
					0.5	0.75	0.95	Avg.	0.3	0.4	0.5	0.6	0.7	Avg.
BMN [30]	TSN	✗	✓	-	50.07	34.78	8.29	33.85	56.0	47.4	38.8	29.7	20.5	38.5
TadTR [34]	I3D	✗	✓	-	49.10	32.60	8.50	32.30	62.4	57.4	49.2	37.8	26.3	46.6
ActionFormer [66]	SlowFast-R50	✗	✗	-	54.26	37.04	8.13	36.02	78.7	73.3	65.2	54.6	39.7	62.3
ActionFormer [66]	I3D	✗	✓	-	53.50	36.20	8.20	35.60	82.1	77.8	71.0	59.4	43.9	66.8
ASL [45]	I3D	✗	✓	-	54.10	67.40	8.00	36.20	83.1	79.0	71.7	59.7	45.8	67.9
TriDet [47]	I3D	✗	✓	-	54.70	38.00	8.40	36.80	83.6	80.1	72.9	62.4	47.4	69.3
VideoMAEv2 [56]	VideoMAEv2-g	✗	✗	-	-	-	-	-	-	-	-	-	-	69.6
InternVideo [57]	VideoMAE-H+UniformerV2	✗	✗	-	-	-	-	39.00*	-	-	-	-	-	71.5
AFSD [29]	I3D	✓	✓	12	52.40	35.30	6.50	34.40	67.3	62.4	55.5	43.7	31.1	52.0
E2E-TAD [33]	SlowFast-R50	✓	✗	12	50.47	35.99	10.33	35.10	69.4	64.3	56.0	46.4	34.9	54.2
BasicTAD [62]	SlowOnly-R50	✓	✗	12	51.20	33.41	7.57	33.12	75.5	70.8	63.5	50.9	37.4	59.6
TALLFormer [10]	VideoSwin-B	✓	✗	29	54.10	36.20	7.90	35.60	76.0	-	63.2	-	34.5	59.2
Re^{2}TAL [69]	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
AdaTAD	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	VideoMAE-S	✓	✗	2.5	56.15	38.99	9.07	37.85	84.5	80.2	71.6	60.9	46.9	68.8
AdaTAD	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	VideoMAE-L	✓	✗	11.0	57.69	40.56	10.13	39.22	87.7	84.1	76.7	66.4	52.4	73.5
AdaTAD	VideoMAE-H	✓	✗	19.2	58.04	40.55	9.75	39.37	88.9	85.3	78.6	66.9	52.5	74.4
AdaTAD	VideoMAEv2-g	✓	✗	29.9	58.45	41.16	10.45	39.79	89.5	85.8	78.9	67.3	52.6	74.8
AdaTAD† (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	75.4
AdaTAD (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**

searched from 5e-4 to 5e-5, and other parameters inside the backbone are frozen. On ActivityNet-1.3, we resize the video into a fixed length of 768 frames. On THUMOS14, we randomly truncate a window with 768 frames with a temporal stride of 4. On EPIC-Kitchens 100, we randomly truncate a window with 6144 frames with a temporal stride of 2. After the video encoder, the feature is resized to fixed lengths of 192, 768, and 768, respectively, for the three datasets. Frame resolution is set to 160^2 by default. In all experiments, we report the training memory usage. More implementation details can be found in the appendix.

4.3. Comparison with SoTA Methods

Table 2 compares our AdaTAD with other state-of-the-art (SoTA) methods on ActivityNet-1.3 and THUMOS14 datasets. Initially, we use SlowFast-R50 as the backbone. For comparison, we also extract corresponding offline features, utilizing the snippet representation where each snippet comprises 32 frames with 224^2 resolution. We observe that end-to-end training enhances performance from 62.3% to 66.0% on THUMOS14. Notably, this architecture has also been used in E2E-TAD [33]. However, our method consumes less memory while achieving superior performance. This apple-to-apple comparison underscores the benefits of adapter tuning and the scaling-up principle.

Table 3. **Results on EPIC-Kitchens 100 validation set.** For comparison, the feature-based methods use the same SlowFast-R50.

Method	E2E	0.1	0.2	0.3	0.4	0.5	Avg.
Verb Task							
BMN [30]	✗	10.8	8.8	8.4	7.1	5.6	8.4
G-TAD [60]	✗	12.1	11.0	9.4	8.1	6.5	9.4
ActionFormer [66]	✗	26.6	25.4	24.2	22.3	19.1	23.5
ASL [45]	✗	27.9	-	25.5	-	19.8	24.6
TriDet [47]	✗	28.6	27.4	26.1	24.2	20.8	25.4
AdaTAD (SlowFast-R50)	✓	26.5	25.7	23.9	21.7	17.6	23.1
ActionFormer (VideoMAE-L)	✗	32.7	31.6	29.1	26.7	23.6	28.7
AdaTAD (VideoMAE-L)	✓	**33.1**	**32.2**	**30.4**	**27.5**	**23.1**	**29.3**
Noun Task							
BMN [30]	✗	10.3	8.3	6.2	4.5	3.4	6.5
G-TAD [60]	✗	11.0	10.0	8.6	7.0	5.4	8.4
ActionFormer [66]	✗	25.2	24.1	22.7	20.5	17.0	21.9
ASL [45]	✗	26.0	-	23.4	-	17.7	22.6
TriDet [47]	✗	27.4	26.3	24.6	22.2	18.3	23.8
AdaTAD (SlowFast-R50)	✓	24.5	23.6	22.3	20.0	16.5	21.4
ActionFormer (VideoMAE-L)	✗	31.3	29.7	27.2	25.3	21.3	26.9
AdaTAD (VideoMAE-L)	✓	**32.4**	**31.6**	**30.1**	**27.4**	**24.6**	**29.3**

Furthermore, when adopting the VideoMAE [53] family as our backbone and progressively scaling up the model size, the performance of AdaTAD consistently improves. Using the largest model, *i.e.*, VideoMAEv2-giant with 1.01

Table 4. **Compared to full fine-tuning, our adapter tuning can achieve better performance with less memory.** Param. is the number of tunable parameters in the backbone. $*$ means out of memory on A100-80GB, and we report the estimated number.

Model	Setting	E2E	Param.	Mem.	mAP
VideoMAE-S	Feature	✗	0	-	57.6
	Snippet Full FT	✓	22M	24.6G	68.4
	Frame Full FT	✓	22M	2.8G	67.1
	AdaTAD	✓	**1M**	**2.5G**	**68.8**
VideoMAE-B	Feature	✗	0	-	64.7
	Snippet Full FT	✓	86M	87.4G*	-
	Frame Full FT	✓	86M	5.6G	70.1
	AdaTAD	✓	**4M**	**4.9G**	**71.5**
VideoMAE-L	Feature	✗	0	-	66.5
	Snippet Full FT	✓	304M	193G*	-
	Frame Full FT	✓	304M	13.1G	73.0
	AdaTAD	✓	**14M**	**11.0G**	**73.5**

Table 5. **When scaling up the input data, AdaTAD's performance consistently increases.** $*$ means snippet representation is used in offline feature extraction, and each snippet has 16 frames.

Setting	Model	Resolution	Frames	Mem.	mAP
Feature	VideoMAEv2-g	224^2	768x16*	-	69.6
AdaTAD	VideoMAE-S	160^2	768	2.5G	68.8
		160^2	1536	3.8G	69.7
		160^2	3072	6.5G	**70.6**
		224^2	768	3.8G	70.7
		224^2	1536	6.4G	71.0
		224^2	3072	11.6G	**71.5**

Table 6. **AdaTAD† can further push the boundaries of scaling up.** OOM means out of memory on A100-80GB.

Setting	Model	Resolution	Frame	Mem.	mAP
AdaTAD	VideoMAE-L	160^2	768	11.0G	73.5
	VideoMAEv2-g	160^2	768	29.9G	74.8
	VideoMAEv2-g	224^2	1536	OOM	-
AdaTAD†	VideoMAEv2-g	160^2	768	22.8G	73.7
	VideoMAEv2-g	224^2	768	30.0G	74.6
	VideoMAEv2-g	224^2	1536	43.6G	**75.4**

billion parameters, and larger input data, *i.e.*, 1536 frames with 224^2 resolution, we attain an impressive 41.9% mAP on ActivityNet-1.3 and 76.9% mAP on THUMOS14. It is noteworthy that the previous SoTA was achieved by VideoMAEv2 [56] and InternVideo [57], which utilize the same detector head as ours but with offline snippet features. Our method surpasses these in detection performance by a large margin, marking the first instance where an end-to-end TAD method can outperform SoTA feature-based results.

We also present our results on EPIC-Kitchens 100 in Table 3. Since videos in this dataset have a longer duration, all previous methods rely solely on pre-extracted features [47, 52, 66]. Our approach is the first to adopt end-to-end training on this dataset. For fair comparison, we first utilize the same backbone as used in previous methods, *i.e.*, SlowFast-R50 pretrained on EPIC, and we achieve comparable performance to ActionFormer [66]. Moreover, when we scale up the backbone to VideoMAE-L (it is also trained on EPIC-Kitchens 100 classification task), we achieve SoTA performance of 29.3%.

4.4. Ablation and Analysis

In this section, we present a series of analyses to evaluate our proposed method and affirm the benefits of scaling up in TAD. Unless otherwise stated, our experiments utilize a standard input of 768 frames per video on THUMOS14.

The advantage of adapter tuning. In Table 4, we compare conventional full fine-tuning with our proposed design. It is evident that end-to-end approaches significantly outperform pre-extracted features. Moreover, with full fine-tuning, the snippet representation slightly advances over frame representation but incurs tremendous memory costs, which aligns with our analysis in Section 3.2. However, AdaTAD uses less memory and still achieves better performance than conventional full fine-tuning. This also verifies the limitations of full fine-tuning, as discussed in Section 3.3. Specifically, our method enhances VideoMAE-S backbone with an 11.2% gain using only 1M trainable parameters. Additionally, Table 4 also demonstrates that scaling up the model size of the video backbone is an effective way to improve TAD performance.

The advantage of scaling up the data. In addition to model scaling, Table 5 verifies the effectiveness of data scaling, which involves two aspects: frame number and frame resolution. Firstly, given the same video duration, increasing the frame number from 768 to 3072 can raise the mAP from 68.8% to 70.6%. In the meantime, the memory usage is nearly three times larger. Secondly, increasing the frame resolution from 160^2 to 224^2 also improves the mAP. In the end, by only scaling up the data, we elevate the mAP from 68.8% to 71.5%, already surpassing the current SoTA feature-based approach with a giant backbone model [57].

Moreover, increasing the frame resolution from 160^2 to 224^2 or increasing the frame number from 768 to 1536 results in the same memory usage of 3.8G. However, the former achieves 70.7% mAP while the latter only reaches 69.7%. This suggests that frame resolution may be prioritized under the same memory budget, for the TAD task.

The advantage of AdaTAD†. Given the effectiveness of scaling up the model or data, we further explore combining these approaches. In Table 6, using 768 frames while scaling up the model to VideoMAEv2-giant results in memory usage escalating to 29.9G. In such a scenario, further increasing the data could easily lead to memory overflow, even with the A100-80GB GPU. This indicates that adaptation tuning reached its limit under this extreme case. Therefore, to utilize both the largest models with larger data simultaneously, AdaTAD† shows its advantage.

Concretely, when switching from AdaTAD to AdaTAD†, the memory usage of the VideoMAEv2-giant model is reduced from 29.9G to 22.8G. Although a slight performance drop is observed, its reduced memory footprint enables scaling up data from 768 frames to as much as 1536 frames with a high resolution of 224^2. This scalability helps mitigate the performance drop and achieves a higher mAP of 75.4%.

The ablation of the adapter design. Detailed in Table 7, we compare different adapter architectural designs. The baseline, *i.e.*, offline snippet feature, achieves 64.7% mAP. End-to-end learning in all designs yields at least a 5% improvement. Our AdaTAD achieves 71.5% in the end. Compared to standard adapter [22], ours consumes similar memory but achieves higher mAP. This verifies that local temporal context is vital for the TAD task. In contrast to full finetuning (FT), we tune only 4M parameters using less memory. In our design, we find that removing the residual connection of the depth-wise convolution drops performance by 0.7%, and training becomes unstable. We also implement an adaptation design proposed in LongLoRA [9], which efficiently computes long-range attention and shows decent performance but requires more parameters and memory.

Table 7. **Ablation of different adapter architectural designs.** VideoMAE-B is used to conduct the following experiments.

Setting	E2E	Param.	Mem.	mAP	gains
Snippet Feature	✗	0	-	64.7	
+ Full FT	✓	86M	5.6G	70.1	+ 5.1
+ LongLoRA [9]	✓	28M	6.2G	71.1	+ 6.1
+ Standard Adapter [22]	✓	3.6M	4.8G	70.2	+ 5.2
+ AdaTAD (w/o residual)	✓	4.0M	4.9G	70.8	+ 5.8
+ **AdaTAD**	✓	**4.0M**	**4.9G**	**71.5**	**+ 6.5**

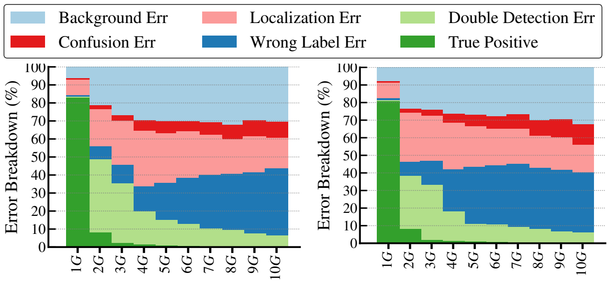
The necessity of end-to-end training for TAD. As previously discussed, end-to-end training can address discrepancies between the pretraining and fine-tuning stages in terms of training data and learning tasks. To corroborate this, we employ models pretrained on different datasets for the EPIC-Kitchens TAD task in Table 8. Kinetics-400 (K400) [26] represents commonly collected third-person web data and exhibits a large domain gap compared to EPIC-Kitchens 100. Using K400 for pretraining, we observe that end-to-end TAD training allows for +5.56 gain. Conversely, using a model already finetuned on EPIC-Kitchens still yields a +2.32 improvement. Unlike K400 pretraining, since this model has already adapted to the data discrepancy, we can infer that this gain leverages differences between the classification task in pretraining and the detection task in fine-tuning. Such improvements further underscore the significance of end-to-end training in TAD.

Table 8. **End-to-end TAD can alleviate the discrepancies between pretraining and finetuning.** VideoMAE-L with different pretrained weights are used on the EPIC-Kitchens 100 Noun task.

Pretrain Dataset	E2E	0.1	0.3	0.5	mAP	gain
K400 [26]	✗	18.69	16.35	11.52	15.77	
K400 [26]	✓	24.33	22.14	16.87	**21.33**	**+ 5.56**
K400 [26] + EPIC [13]	✗	31.32	27.25	21.33	26.98	
K400 [26] + EPIC [13]	✓	32.41	30.13	24.59	**29.30**	**+ 2.32**

4.5. Error Analyses

We also conduct false positive analysis at tIoU=0.5 in Fig. 4. Compared to feature-based training, learning from raw frames produces more helpful positive detections. More importantly, the percentage of wrong label error is reduced after end-to-end training, suggesting its unique advantage in classifying accurate action labels.



(a) End-to-end training (b) Feature-based training

Figure 4. **False Positive Profiling on THUMOS14 using [2].** We use VideoMAEv2-giant as the backbone, and compare end-to-end training with pre-extracted feature-based training.

5. Conclusions

This work introduces a memory-efficient and parameter-efficient end-to-end method named **AdaTAD**. Our key innovation lies in the proposed temporal-informative adapter, which is tailored for TAD with low computation costs. Furthermore, we design an alternative placement for adapters to minimize memory usage. By demonstrating the feasibility and effectiveness of scaling up end-to-end TAD, our work achieves new SoTA performance across multiple datasets. Particularly, this is the first instance of an end-to-end TAD method that surpasses the current best feature-based models. In fact, AdaTAD achieves a groundbreaking 75.4% mAP on THUMOS14. We believe our work underscores the possible paradigm shift in TAD, advocating a move away from the traditional methodology of separate feature extraction and offline detection towards a more integrated approach of scaling up end-to-end TAD training.

Acknowledgement. This work was supported by the King Abdullah University of Science and Technology (KAUST) Office of Sponsored Research through the Visual Computing Center (VCC) funding, as well as the SDAIA-KAUST Center of Excellence in Data Science and Artificial Intelligence (SDAIA-KAUST AI).

References

[1] Ibrahim M Alabdulmohsin, Behnam Neyshabur, and Xiaohua Zhai. Revisiting neural scaling laws in language and vision. In *NeurIPS*, 2022. 3

[2] Humam Alwassel, Fabian Caba Heilbron, Victor Escorcia, and Bernard Ghanem. Diagnosing error in temporal action detectors. In *ECCV*, 2018. 8

[3] Humam Alwassel, Fabian Caba Heilbron, and Bernard Ghanem. Action search: Spotting actions in videos and its application to temporal action localization. In *ECCV*, 2018. 1

[4] Yueran Bai, Yingying Wang, Yunhai Tong, Yang Yang, Qiyue Liu, and Junhui Liu. Boundary content graph neural network for temporal action proposal generation. In *ECCV*, 2020. 2

[5] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. In *NeurIPS*, 2020. 2

[6] Hao Chen, Ran Tao, Han Zhang, Yidong Wang, Wei Ye, Jindong Wang, Guosheng Hu, and Marios Savvides. Conv-adapter: Exploring parameter efficient transfer learning for convnets. *arXiv preprint arXiv:2208.07463*, 2022. 3

[7] Shoufa Chen, Chongjian Ge, Zhan Tong, Jiangliu Wang, Yibing Song, Jue Wang, and Ping Luo. Adaptformer: Adapting vision transformers for scalable visual recognition. In *NeurIPS*, 2022. 3

[8] Tianqi Chen, Bing Xu, Chiyuan Zhang, and Carlos Guestrin. Training deep nets with sublinear memory cost. *arXiv preprint arXiv:1604.06174*, 2016. 4

[9] 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

[10] Feng Cheng and Gedas Bertasius. Tallformer: Temporal action localization with long-memory transformer. In *ECCV*, 2022. 1, 2, 4, 6

[11] Feng Cheng, Mingze Xu, Yuanjun Xiong, Hao Chen, Xinyu Li, Wei Li, and Wei Xia. Stochastic backpropagation: A memory efficient strategy for training video models. In *CVPR*, 2023. 2

[12] MMAction2 Contributors. Openmmlab's next generation video understanding toolbox and benchmark. `https://github.com/open-mmlab/mmaction2`, 2020. 5

[13] Dima Damen, Hazel Doughty, Giovanni Maria Farinella, Sanja Fidler, Antonino Furnari, Evangelos Kazakos, Davide Moltisanti, Jonathan Munro, Toby Perrett, Will Price, et al. Scaling egocentric vision: The epic-kitchens dataset. In *ECCV*, 2018. 5, 8

[14] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *ICLR*, 2021. 1, 3

[15] Victor Escorcia, Fabian Caba Heilbron, Juan Carlos Niebles, and Bernard Ghanem. DAPs: Deep action proposals for action understanding. In *ECCV*, 2016. 1

[16] Haoqi Fan, Bo Xiong, Karttikeya Mangalam, Yanghao Li, Zhicheng Yan, Jitendra Malik, and Christoph Feichtenhofer. Multiscale vision transformers. In *ICCV*, 2021. 3

[17] Christoph Feichtenhofer, Haoqi Fan, Jitendra Malik, and Kaiming He. SlowFast networks for video recognition. In *ICCV*, 2019. 3, 4

[18] Kristen Grauman, Andrew Westbury, Eugene Byrne, Zachary Chavis, Antonino Furnari, Rohit Girdhar, Jackson Hamburger, Hao Jiang, Miao Liu, Xingyu Liu, et al. Ego4d: Around the world in 3,000 hours of egocentric video. In *CVPR*, 2022. 5

[19] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *CVPR*, 2022. 1

[20] Fabian Caba Heilbron, Victor Escorcia, Bernard Ghanem, and Juan Carlos Niebles. ActivityNet: A large-scale video benchmark for human activity understanding. In *CVPR*, 2015. 1, 5

[21] Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016. 4

[22] 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 *ICML*, 2019. 2, 3, 4, 8

[23] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. In *ICML*, 2021. 3

[24] Menglin Jia, Luming Tang, Bor-Chun Chen, Claire Cardie, Serge Belongie, Bharath Hariharan, and Ser-Nam Lim. Visual prompt tuning. In *ECCV*, 2022. 3

[25] YG Jiang, J Liu, A Roshan Zamir, G Toderici, I Laptev, M Shah, and R Sukthankar. Thumos challenge: Action recognition with a large number of classes, 2014. 1, 5

[26] Will Kay, Joao Carreira, Karen Simonyan, Brian Zhang, Chloe Hillier, Sudheendra Vijayanarasimhan, Fabio Viola, Tim Green, Trevor Back, Paul Natsev, et al. The kinetics human action video dataset. *arXiv preprint arXiv:1705.06950*, 2017. 8

[27] Brian Lester, Rami Al-Rfou, and Noah Constant. The power of scale for parameter-efficient prompt tuning. In *EMNLP*, 2021. 3

[28] Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation. In *ACL*, 2021. 3

[29] 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 *CVPR*, 2021. 1, 2, 4, 6

[30] Tianwei Lin, Xiao Liu, Xin Li, Errui Ding, and Shilei Wen. BMN: boundary-matching network for temporal action proposal generation. In *ICCV*, 2019. 1, 2, 6

[31] Qinying Liu and Zilei Wang. Progressive boundary refinement network for temporal action detection. In *AAAI*, 2020. 2

[32] Shuming Liu, Mengmeng Xu, Chen Zhao, Xu Zhao, and Bernard Ghanem. Etad: Training action detection end to end on a laptop. In *CVPRW*, 2023. 1, 2

[33] Xiaolong Liu, Song Bai, and Xiang Bai. An empirical study of end-to-end temporal action detection. In *CVPR*, 2022. 1, 2, 6

[34] 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

[35] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *ICCV*, 2021. 3

[36] Jinjie Mai, Abdullah Hamdi, Silvio Giancola, Chen Zhao, and Bernard Ghanem. Egoloc: Revisiting 3d object localization from egocentric videos with visual queries. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 45–57, 2023. 1

[37] Paulius Micikevicius, Sharan Narang, Jonah Alben, Gregory Diamos, Erich Elsen, David Garcia, Boris Ginsburg, Michael Houston, Oleksii Kuchaiev, Ganesh Venkatesh, et al. Mixed precision training. In *ICLR*, 2017. 4

[38] Jonghwan Mun, Minsu Cho, and Bohyung Han. Local-global video-text interactions for temporal grounding. In *CVPR*, 2020. 1

[39] OpenAI. Gpt-4 technical report, 2023. 1, 2

[40] Zhiwu Qing, Haisheng Su, Weihao Gan, Dongliang Wang, Wei Wu, Xiang Wang, Yu Qiao, Junjie Yan, Changxin Gao, and Nong Sang. Temporal context aggregation network for temporal action proposal refinement. In *CVPR*, 2021. 2

[41] Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. Improving language understanding by generative pre-training. *OpenAI*, 2018. 2

[42] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019. 2

[43] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *ICML*, 2021. 3

[44] Merey Ramazanova, Victor Escorcia, Fabian Caba Heilbron, Chen Zhao, and Bernard Ghanem. Owl (observe, watch, listen): Localizing actions in egocentric video via audiovisual temporal context. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshop (CVPRW)*, 2023. 1

[45] Jiayi Shao, Xiaohan Wang, Ruijie Quan, Junjun Zheng, Jiang Yang, and Yi Yang. Action sensitivity learning for temporal action localization. In *ICCV*, 2023. 2, 6

[46] Dingfeng Shi, Yujie Zhong, Qiong Cao, Jing Zhang, Lin Ma, Jia Li, and Dacheng Tao. React: Temporal action detection with relational queries. In *ECCV*, 2022. 2

[47] Dingfeng Shi, Yujie Zhong, Qiong Cao, Lin Ma, Jia Li, and Dacheng Tao. Tridet: Temporal action detection with relative boundary modeling. In *CVPR*, 2023. 2, 6, 7

[48] Mattia Soldan, Mengmeng Xu, Sisi Qu, Jesper Tegner, and Bernard Ghanem. VLG-Net: Video-language graph matching network for video grounding. In *ICCVW*, 2021. 1

[49] Yi-Lin Sung, Jaemin Cho, and Mohit Bansal. Lst: Ladder side-tuning for parameter and memory efficient transfer learning. In *NeurIPS*, 2022. 3, 5

[50] Yi-Lin Sung, Jaemin Cho, and Mohit Bansal. Vl-adapter: Parameter-efficient transfer learning for vision-and-language tasks. In *CVPR*, 2022. 2, 4

[51] Jing Tan, Jiaqi Tang, Limin Wang, and Gangshan Wu. Relaxed transformer decoders for direct action proposal generation. In *ICCV*, 2021. 2

[52] Tuan N Tang, Kwonyoung Kim, and Kwanghoon Sohn. Temporalmaxer: Maximize temporal context with only max pooling for temporal action localization. *arXiv preprint arXiv:2303.09055*, 2023. 7

[53] Zhan Tong, Yibing Song, Jue Wang, and Limin Wang. Videomae: Masked autoencoders are data-efficient learners for self-supervised video pre-training. In *NeurIPS*, 2022. 1, 4, 6

[54] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, 2017. 4

[55] Chenhao Wang, Hongxiang Cai, Yuxin Zou, and Yichao Xiong. Rgb stream is enough for temporal action detection. *arXiv preprint arXiv:2107.04362*, 2021. 2

[56] 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 *ICCV*, 2023. 1, 3, 6, 7

[57] Yi Wang, Kunchang Li, Yizhuo Li, Yinan He, Bingkun Huang, Zhiyu Zhao, Hongjie Zhang, Jilan Xu, Yi Liu, Zun Wang, et al. Internvideo: General video foundation models via generative and discriminative learning. *arXiv preprint arXiv:2212.03191*, 2022. 1, 6, 7

[58] Zhenzhi Wang, Ziteng Gao, Limin Wang, Zhifeng Li, and Gangshan Wu. Boundary-aware cascade networks for temporal action segmentation. In *ECCV*, 2020. 2

[59] Kun Xia, Le Wang, Sanping Zhou, Nanning Zheng, and Wei Tang. Learning to refactor action and co-occurrence features for temporal action localization. In *CVPR*, 2022. 2

[60] Mengmeng Xu, Chen Zhao, David S Rojas, Ali Thabet, and Bernard Ghanem. G-TAD: Sub-graph localization for temporal action detection. In *CVPR*, 2020. 1, 6

[61] Le Yang, Houwen Peng, Dingwen Zhang, Jianlong Fu, and Junwei Han. Revisiting anchor mechanisms for temporal action localization. *IEEE Transactions on Image Processing*, 29:8535–8548, 2020. 2

[62] 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. 1, 2, 6

[63] Taojiannan Yang, Yi Zhu, Yusheng Xie, Aston Zhang, Chen Chen, and Mu Li. Aim: Adapting image models for efficient video action recognition. In *ICLR*, 2023. 2, 3, 4

[64] Ting Yao, Tao Mei, and Yong Rui. Highlight detection with pairwise deep ranking for first-person video summarization. In *CVPR*, 2016. 1

[65] Dongshuo Yin, Xueting Han, Bin Li, Hao Feng, and Jing Bai. Parameter-efficient is not sufficient: Exploring parameter, memory, and time efficient adapter tuning for dense predictions. *arXiv preprint arXiv:2306.09729*, 2023. 3, 5

[66] Chenlin Zhang, Jianxin Wu, and Yin Li. Actionformer: Localizing moments of actions with transformers. In *ECCV*, 2022. 1, 2, 3, 5, 6, 7

[67] Chen Zhao, Ali K Thabet, and Bernard Ghanem. Video self-stitching graph network for temporal action localization. In *ICCV*, 2021. 1, 2

[68] Chen Zhao, Merey Ramazanova, Mengmeng Xu, and Bernard Ghanem. Segtad: Precise temporal action detection via semantic segmentation. In *ECCVW*, 2022. 2

[69] Chen Zhao, Shuming Liu, Karttikeya Mangalam, and Bernard Ghanem. Re^{2}TAL: Rewiring pretrained video backbones for reversible temporal action localization. In *CVPR*, 2023. 1, 2, 3, 4, 6

[70] Chen Zhao, Shuming Liu, Karttikeya Mangalam, Guocheng Qian, Fatimah Zohra, Abdulmohsen Alghannam, Jitendra Malik, and Bernard Ghanem. Dr^{2}Net: Dynamic reversible dual-residual networks for memory-efficient finetuning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024. 1, 2

[71] Hang Zhao, Zhicheng Yan, Lorenzo Torresani, and Antonio Torralba. HACS: Human action clips and segments dataset for recognition and temporal localization. *ICCV*, 2019. 1

[72] Y Zhao, B Zhang, Z Wu, S Yang, L Zhou, S Yan, L Wang, Y Xiong, D Lin, Y Qiao, et al. Cuhk & ethz & siat submission to activitynet challenge 2017. *CVPR ActivityNet Workshop*, 2017. 6

[73] Zixuan Zhao, Dongqi Wang, and Xu Zhao. Movement enhancement toward multi-scale video feature representation for temporal action detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 13555–13564, 2023. 2