An Empirical Study of End-to-End Temporal Action Detection

Xiaolong Liu[1] Song Bai[2] Xiang Bai[1*]
[1]Huazhong University of Science and Technology [2]ByteDance Inc.
{liuxl, xbai}@hust.edu.cn, songbai.site@gmail.com

Abstract

Temporal action detection (TAD) is an important yet challenging task in video understanding. It aims to simultaneously predict the semantic label and the temporal interval of every action instance in an untrimmed video. Rather than end-to-end learning, most existing methods adopt a head-only learning paradigm, where the video encoder is pre-trained for action classification, and only the detection head upon the encoder is optimized for TAD. The effect of end-to-end learning is not systematically evaluated. Besides, there lacks an in-depth study on the efficiency-accuracy trade-off in end-to-end TAD. In this paper, we present an empirical study of end-to-end temporal action detection. We validate the advantage of end-to-end learning over head-only learning and observe up to 11% performance improvement. Besides, we study the effects of multiple design choices that affect the TAD performance and speed, including detection head, video encoder, and resolution of input videos. Based on the findings, we build a mid-resolution baseline detector, which achieves the state-of-the-art performance of end-to-end methods while running more than 4× faster. We hope that this paper can serve as a guide for end-to-end learning and inspire future research in this field. Code and models are available at https://github.com/xlliu7/E2E-TAD.

1. Introduction

With the development of information technology, the numbers of videos generated and accessed are rapidly increasing, underscoring the need for automatic video understanding, such as human action recognition and temporal action detection (TAD)[1]. Action recognition aims to predict the action label (*e.g.*, basketball dunk) of a short, trimmed video. Differently, TAD aims to determine the label, as well as the temporal interval of every action instance in a long untrimmed video. It is more challenging and also practical in real-world actions, such as security surveillance, sports analysis, and smart video editing.

Owing to the strong discriminative power of neural networks, deep learning methods have dominated the field of temporal action detection [20, 48, 49, 53]. As depicted in Fig. 1, a temporal action detector typically consists of a **video encoder** and a **detection head**, similar to the

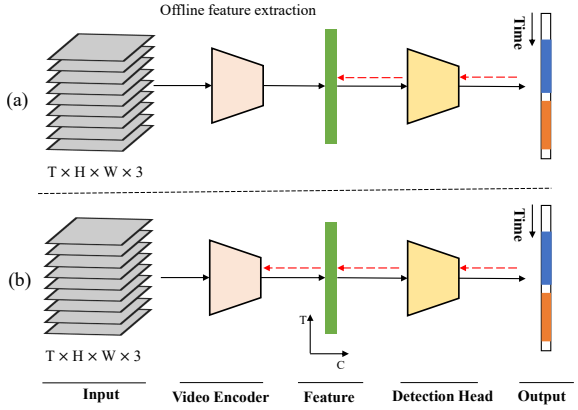

Figure 1. Head-only learning (a) *vs*. end-to-end learning (b) for temporal action detection. Solid arrows and dashed arrows represent forward pass and the gradient flow of back propagation.

Method	E2E	Flow	FLOPs	Latency	mAP
THUMOS14					
MUSES [25]		✓	17.5T	72s*+2.1s	53.4
AFSD [18]	✓	✓	2780G	2472ms	52.0
Ours	✓		475G	587ms	**54.2**
ActivityNet					
AFSD [18]	✓	✓	499G	291ms	34.39
Ours	✓		62G	63ms	**35.10**

Table 1. Comparison between the baseline detector built in this work (ours) with state-of-the-art methods. The latency and FLOPs are measured at the video level. The time of optical flow extraction is not included in latency. *The time cost of I3D [6] feature extraction. E2E: end-to-end.

*Corresponding author
[1]Also known as temporal action localization (TAL).

backbone-head structure in object detection [12, 33, 39]. Different from modern object detectors that are trained end-to-end[2], most TAD methods adopt a **head-only learning** paradigm. They first pre-train the video encoder on a large action recognition dataset (*e.g.*, Kinetics [6]) then freeze it for offline feature extraction. After that, only the detection head upon the features is trained for the TAD task on the target datasets. This leaves the video features sub-optimal and restricts the performance.

Although a few works [18, 28, 45] have adopted end-to-end learning, there lacks an in-depth analysis of it. The actual benefit of end-to-end learning is still unclear. Besides, the effects of many factors in end-to-end TAD, such as the video encoder, the detection head, the image and temporal resolution of input videos, are not systematically studied. In a way, lack of such a study blocks the research of end-to-end TAD. Moreover, existing works more or less neglect the efficiency, which is an important factor in real-world applications. For example, in large-scale systems, such as online video platforms, running time determines computational expenses. Unfortunately, most methods do not discuss the computation cost. A few works discuss the running time of certain parts of the full model, *e.g.*, the detection head [20,26,30,54] or report inference speed (FPS) [18,45]. But they do not explore the efficiency-accuracy trade-off. This paper aims to address the above issues.

We conduct an empirical study of end-to-end temporal action detection. Four video encoders and three detection heads with different high-level designs are evaluated on two standard TAD datasets, *i.e.*, THUMOS14 and ActivityNet. **Firstly**, we uncover the benefit of end-to-end learning. It is shown that end-to-end trained video encoders with a medium image resolution (96^2) can match or surpass pre-trained ones with standard image resolution (224^2) in terms of TAD performance. **Secondly**, we evaluate the effect of a series of design choices on performance and efficiency, including detection head, video encoder, image resolution and temporal resolution. It may serve as a guide for seeking the efficiency-accuracy trade-off. **Lastly**, we build a baseline detector based on our study. It achieves state-of-the-art performance of end-to-end TAD while running more than 4× faster (see Tab. 1). Specifically, it can process a 4-minute video in only 0.6 seconds. We hope that our work can facilitate future research in temporal action detection.

2. Related Works

Temporal Action Detection Methods. Current temporal action detection methods can be roughly categorized into three groups. **Anchor-based methods** [7, 17, 18, 30, 37, 53, 57, 59] first generate a dense set of anchors, *i.e.*, temporal segments that may contain an action, then leverage a classifier to classify them into background or one action class. In these methods, anchors are generated by uniform sampling [3, 7, 8, 11, 35, 45], grouping potential action boundaries [22, 34, 56, 57], or a combination of the them [10, 27]. **Anchor-free methods** [2, 18, 21, 34, 52] directly predict the action class for each frame in the video. Then they group frames with the same class into temporal segments. Some methods [18, 50] additionally regress the distance to action boundaries. **Query-based methods** [26, 38] draw inspiration from the DETR object detection framework [5]. They take as input a small set of learnable embeddings called action queries and video features, and map each query to an action prediction. This is achieved via Transformer attention [42] that models the relations between query embeddings and video features. Owing to a one-to-one matching mechanism between ground truth actions and queries, they generate spare and unique action predictions. Different from previous methods that mostly focus on the design of network architecture or framework, we focus on the learning paradigm and efficiency-accuracy trade-off.

Video Encoders. The video encoders in TAD are adapted from action recognition networks by dropping the classification heads. In previous methods, two-stream networks (*e.g.*, TSN [43]) and 3D Convolutional Neural Networks (*e.g.*, C3D [40], I3D [6]) are commonly used video encoders. Two-stream networks, firstly proposed in [36], consist of two 2D Convolutional Neural Network (CNN) streams that operate on RGB frames and optical flow frames separately and their outputs are fused. In two-stream methods, optical flow is crucial for high performance as they explicitly capture motion cues. However, the calculation of optical flow is very expensive. Differently, 3D networks can capture motion information from a sequence of frames, at the cost of more parameters and computation than 2D networks. I3D [6], a representative of this kind, is widely used in previous TAD methods. To mitigate the above issues of 3D networks, recent methods [9, 19, 32, 41, 44] use different ways to approximate 3D convolution. For example, decomposing 3D convolution into 1D and 2D convolution, or combining a temporal shift operation [19] with 2D convolution. In this paper, we evaluate various video encoders to examine their performance and efficiency in temporal action detection. Their effects have not been systematically studied before.

Learning Paradigms of TAD. Most TAD methods first extract features with video encoders pre-trained on action recognition (classification) datasets (*e.g.* Kinetics-400 [6], similar to the role of ImageNet in image recognition). Then they train and evaluate the detection head with the extracted features. In this way, the experimental period can be greatly shortened. Therefore, it is adopted by most existing works. However, there are two issues in this learning paradigm,

[2]"End-to-end" has diverse meanings in literature. Here we mean joint learning of the video encoder and the detection head in a detector.

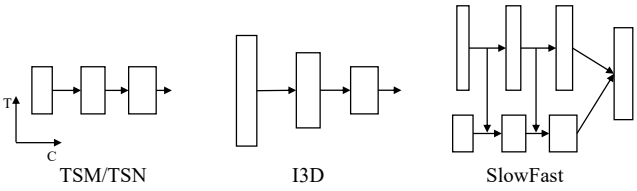

Figure 2. A high-level diagram of the video encoders studied in this work. For simplicity, we do not show the spatial dimension.

task inconsistency and data inconsistency between the pre-training stage and the downstream TAD stage. To deal with the task inconsistency issue, [46] designs a pre-training task that classifies synthesized video clips with different kinds of boundaries. To cope with the data inconsistency issue, some works [22, 28, 29] pre-train the video encoder for action recognition on the target TAD dataset. This paper explores an alternative way of end-to-end training on the TAD datasets. The goal of this paper is not to compare end-to-end training with other pre-training techniques. Instead, we aim to explore the effects of a series of factors on speed and accuracy and seek a trade-off between them.

3. Experimental Setup

In this section, we review the video encoders and temporal action detection heads that we study in this paper. The datasets for performance evaluation and the implementation details are also described here.

3.1. Video Encoders

We mainly study four kinds of video encoders, TSN [43], TSM [19], I3D [6] and SlowFast [9]. Fig. 2 illustrates the network structures of these video encoders.

TSN is a pure 2D CNN encoder. It processes each frame independently.

TSM combines a temporal shift operation with 2D convolution as a basic building block of video encoders. The shift operation moves a small fraction of channels of the input feature map forward and another fraction backward in the temporal axis. It is equivalent to temporal 1D convolution with constant parameters but introduces no computation cost. Spatiotemporal features from multiple frames are then captured with 2D convolution on the shifted features.

I3D follows the design of the Inception network [14] for image recognition but inflates all convolutional and pooling layers into 3D counterparts. As temporal pooling is involved, it outputs feature maps with different resolution in different stages of the network.

SlowFast (SF) consists of a slow pathway and fast pathway that operate on sparsely and densely sampled video frames respectively. The fast pathway has fewer channels than the slow pathway. Therefore it can efficiently capture motion information, which is fused to the slow pathway stage by stage. It follows recent works [32, 41] to apply 1D and 2D convolution iteratively.

3.2. Temporal Action Detection Heads

We study three kinds of temporal action detection heads (methods), anchor-based, anchor-free, and query-based. G-TAD [48], AFSD [18], and TadTR [26] are selected as the representative of each kind for their state-of-the-art performances. Here we briefly describe their frameworks.

G-TAD views a video as a graph and all snippets in the video as its nodes. With such a formulation, the context information in the video can be captured by graph convolution on these nodes. These nodes are sampled as potential action boundaries and paired nodes become anchors. Similar to RoIAlign [13], an SGAlign operation is designed to extract aligned features within the temporal region of each anchor. These anchors are then classified by several fully connected layers upon the aligned features.

AFSD is an anchor-free detector. Inspired by the anchor-free methods [31, 39] in object detection, it detects actions by predicting the action class and the distances to action boundaries for each frame. Using this formulation, it first generates coarse action predictions with pyramid features from the video encoder. To enhance the detection performance, a saliency-based refinement module is designed. It extracts the salient features around the boundaries of each predicted action via a boundary pooling operation. These features are utilized to generate refined predictions.

TadTR views TAD as a direct set prediction problem. Based on Transformer [42], it maps a small set of learned action query embeddings to corresponding action predictions with a Transformer encoder-decoder architecture. The Transformer encoder takes as input the features from the video encoder. It models the long-range dependency in the temporal dimension with a sparse attention mechanism and captures the global context. The decoder looks up global context related to each query via cross-attention and predicts the boundaries and the action class thereon. In pursuit of more accurate boundaries and confidence scores, it utilizes a segment refinement mechanism that iteratively refines the boundaries in each decoder layer and an actions regression head that re-computes a confidence score according to the final predicted boundaries.

3.3. End-to-end Learning

We drop the classifier in the original network of each video encoder and modify the last global pooling layer to only perform spatial pooling. Then the detection head is attached to the last layer of the encoder, resulting in a unified network. The network directly takes video frames as

input and is trained with the loss functions defined by each detector. During training, gradients flow backward to both the head and the video encoder. In this way, they can be optimized simultaneously towards stronger temporal action detection performance.

3.4. Datasets

We conduct evaluations on two datasets, THUMOS14 [15] and ActivityNet [4] (v1.3). **THUMOS14** collects sports videos from 20 classes. It contains 200 and 212 untrimmed videos for training and testing. The actions are densely distributed and very short. The average length of videos and actions is 4.4 minutes and 5 seconds respectively. **ActivityNet** consists of 19994 videos in 200 action classes of daily activities. It contains 10024, 4926, and 5044 videos in the training, validation, and testing sets. Following previous work, we use the validation set for evaluation, as the annotations on the testing set are reserved by the organizers. The average length of videos and actions is 2 minutes and 48 seconds respectively.

Evaluation Metrics. For both datasets, we use mean Average Precision (mAP) at different temporal IoU thresholds as the evaluation metric. On THUMOS14, the IoU thresholds are $\{0.3, 0.4, 0.5, 0.6, 0.7\}$. On ActivityNet, we choose 10 values ranging from 0.5 to 0.95 with a step of 0.05. We also report the average of the mAP at all thresholds, which is the primary metric for performance comparison.

3.5. Implementation Details

Video Encoders. The SlowFast encoder has several variants. We choose the "SlowFast 4x16, R50" variant for its efficiency. Given an input clip of N frames, the fast and the slow pathway sample N and N/8 frames respectively. We resize the output features of the two pathways to the same length and concatenate them into one. The length is set to N/4. In other words, the temporal output stride is 4. I3D extract features of multiple temporal resolutions. A **feature fusion** strategy is applied to better utilize these features. We temporally up-sample the features from the fifth stage by $2\times$ and fuse it with the features from the fourth stage. In this way, the temporal output stride is also 4. As a reference, the temporal output stride of TSM and TSN is 1.

Clip Sampling. We use video clips for training and evaluation. On THUMOS14, we uniformly sample clips of 25.6 seconds, which is longer than 99.6% of all action instances. The sampling stride between adjacent clips is set to 25% and 75% of the clip length during training and evaluation, respectively. Unless specially noted, TSM and TSN sample video frames at 3.75 FPS on THUMOS14. SlowFast and I3D sample frames at 10 FPS. On ActivityNet, as the ratio of action length to video length is much larger, we follow [18] to treat each full video as a single clip and sample a fixed number of frames as the input to video encoders. According to [18], this strategy is better than sampling with a fixed frame rate. This number is set to 384 for SlowFast and I3D and 96 for TSM and TSN. In this way, the output features of these encoders have the same length of 96 (an average of 0.8 FPS). By default, we set the image size of the input video to 96×96, which has $5.4\times$ fewer pixels than the commonly used 224×224 resolution.

Training. The models are trained with Adam [16] optimizer, setting weight decay to 10^{-4}. The base learning rate is set to 10^{-4} and 5×10^{-5} on THUMOS14 and ActivityNet empirically. The learning rate of the video encoder is multiplied by a factor of 0.1, which helps to stabilize training. We divide the learning rate by 10 after τ_1 epochs and the total number of epochs is τ_2. We set $\tau_1 = 10$ and $\tau_2 = 12$ on THUMOS14. On ActivityNet, they are set to 8 and 10, respectively. We set the batch size to 4 for all models and freeze the batch normalization layers in the video encoders. With this configuration, most models can be trained using **a single GPU with 12 GB of memory**. We analyze the effect of batch size in the supplementary and observe that varying batch size from 4 to 16 gives similar performance. We use cropping, horizontal flipping, rotation and photometric distortion for image augmentation. The angle range of random rotation is (-45, 45) degree. The settings of photometric distortion follow [24]. The probability of the latter three transformations is 0.5.

Inference. We follow the details of each detection head in their original implementation. On ActivityNet, we follow previous works [18, 20, 22, 28, 47, 48, 54] to perform class-agnostic localization and use the video-level classification labels from [58]. Latency is measured on a single TITAN Xp GPU, with the batch size set to 1. We take the average time of 100 runs after 10 warm-up runs. Unless specially noted, the computation costs on THUMOS14 are measured for video clips of 25.6 seconds.

4. Results and Analyses

4.1. The Effect of End-to-end Learning

Head-only *vs*. E2E. In Tab. 2, we compare the performance of traditional head-only learning and end-to-end learning using the TadTR detector. When studying the performance gain of end-to-end learning, we keep the same mid-resolution (96×96) setting. We also list the performance of head-only learning with 224×224 resolution. We see that:

(I) End-to-end learning consistently improves performance on multiple datasets and backbones. On THUMOS14, end-to-end learning improves the average mAP by 9.41% and 11.21% with TSM ResNet-18 and TSM ResNet-50 encoders respectively. On ActivityNet, it achieves an im-

provement of 1.30% and 1.38% average mAP with the two encoders respectively. We show that this also generalizes to other video encoders (I3D and SlowFast) and detection heads (AFSD and G-TAD) **in the supplementary**.

(II) The performance of mid-resolution (96^2) end-to-end models can match or surpass that of standard-resolution (224^2) models trained in the head-only paradigm. On THUMOS14, the former outperforms the latter by 7.52% (45.25% *vs*. 37.77%) on the TSM ResNet-50 encoder. A similar observation is drawn on TSM ResNet-18. On ActivityNet, the performance of the above two settings is close. It indicates that end-to-end training is an effective way of enhancing efficient mid-resolution models.

(III) The performance gains of end-to-end learning on ActivityNet are smaller than those on THUMOS14. There are two reasons. 1) The performance gain on ActivityNet only reflects **the effect of end-to-end learning on the localization sub-task**, as the detectors only perform class-agnostic localization on this dataset. To verify this, we evaluate the effect of end-to-end training on class-aware detection on ActivityNet. Compared with head-only learning, end-to-end learning enjoys a gain of 5.70% mAP (19.38% to 25.08%, with TSM ResNet50), which is larger than the gain on the localization sub-task. It means the classification sub-task also benefits from E2E learning. 2) **ActivityNet and THUMOS14 have different characteristics**. THUMOS14 poses a great challenge to temporal localization, as the actions are shorter and each video has a large amount of background (71%) on average. Differently, on ActivityNet, actions are much longer and each video has only 36% background on average. To verify the effect of different characteristics, we conduct a comparison of E2E and head-only learning on HACS Segments [55], which shares the same classes and has a similar distribution as ActivityNet. We observe that E2E learning results in an improvement of 6.28% mAP (19.28% to 25.70%, with TSM ResNet-50), similar to the observation on class-aware detection on ActivityNet.

Image Augmentations. One particular benefit of end-to-end learning is the feasibility of image augmentations. Except for the commonly used random cropping and random horizontal flipping augmentations, we also study stronger augmentations, including random rotation and random photometric distortion. The effect of these augmentations is depicted in Tab. 3. On both datasets, they result in large performance gains. On THUMOS14, random cropping brings a 3.32% improvement. Random flipping further improves the performance by 1.09%. Using stronger augmentations, the average mAP is boosted by 1.35%. In total, the improvement is 5.76%. This is reasonable as THUMOS14 is a relatively smaller dataset. On ActivityNet, the average mAP improves from 31.98% to 33.42% (+1.44%). We find that stronger data augmentations do not provide a clear performance gain, as ActivityNet is already a large-scale dataset.

Paradigm	Img. Res.	ResNet-18	ResNet-50
THUMOS14			
Head-only	224^2	33.79	37.77
Head-only	96^2	28.90	34.04
E2E	96^2	**38.31**	**45.25**
Gain	-	+9.41	+11.21
ActivityNet			
Head-only	224^2	**33.43**	**34.21**
Head-only	96^2	32.12	32.76
E2E	96^2	33.42	34.14
Gain	-	+1.30	+1.38

Table 2. **Head-only learning *vs*. end-to-end (E2E) learning**. Average mAP is reported. Head: TadTR. Video encoder: TSM.

Augmentation	Average mAP			
Cropping		✓	✓	✓
Horizontal Flipping			✓	✓
Rotation				✓
Distortion				✓
THUMOS14	39.49	42.81	43.90	**45.25**
ActivityNet	31.98	33.24	33.40	**33.42**

Table 3. The effect of **image augmentations**. Head: TadTR. Video encoder: TSM ResNet-50 on THUMOS14 and TSM ResNet-18 on ActivityNet.

It is worth noting that end-to-end learning without image augmentation performs worse than head-only learning, possibly due to overfitting.

4.2. Evaluation of Design Choices

Detection Heads. Tab. 4 and Tab. 5 and compare different heads on ActivityNet and THUMOS14 respectively. Note that we use the labels from the external video-level action classifiers for G-TAD following the original paper [48], as this head is designed to generate class-agnostic proposals. Although the detection head only contributes to a small fraction of the computation cost of a detector, there are still differences between detectors in performance, computation cost, and model size. To be specific:

(I) Performance: On both datasets, the query-based detector TadTR achieves the best performance. Its advantage is large in mAP at high IoU thresholds. Specifically, it outperforms G-TAD by 5.19% at the strict IoU threshold 0.95 on ActivityNet. On THUMOS14, it outperforms AFSD [18] by 4.5% in terms of mAP@0.7 on THUMOS14 using the I3D encoder. We observe that G-TAD achieves much lower

Head	FLOPs/G	Latency/ms	Params	0.5	0.75	0.95	Avg.
AFSD*	249.4/3.3	145.5/26.9	**30M**	-	-	-	32.90
G-TAD	169.2/44.6	99.5/31.0	38M	49.22	34.55	4.74	33.17
TadTR	**125.6/0.9**	**78.4/9.7**	45M	**49.56**	**35.24**	**9.93**	**34.35**

Table 4. Comparison of end-to-end trained detectors with different **heads** on ActivityNet. Encoder: **I3D**. All methods use 384 frames inputs (except * uses 768 frames). The values before and after each slash are measured for the full network and the head respectively.

Head	0.3	0.4	0.5	0.6	0.7	Avg.
I3D with a frame rate of 10 FPS						
AFSD*	57.7	52.8	45.4	34.9	22.0	43.6
G-TAD	52.5	45.9	37.6	28.5	19.1	36.7
TadTR	**59.6**	**54.5**	**47.0**	**37.8**	**26.5**	**45.1**
TSM ResNet-50 with a frame rate of 2.5 FPS						
AFSD	56.0	50.0	42.2	32.8	20.5	40.3
G-TAD	51.5	43.4	33.8	23.5	13.6	33.2
TadTR	**58.1**	**52.9**	**44.6**	**36.2**	**24.1**	**43.2**

Table 5. Comparison of end-to-end trained detectors with different **heads** on THUMOS14. * Results from [18].

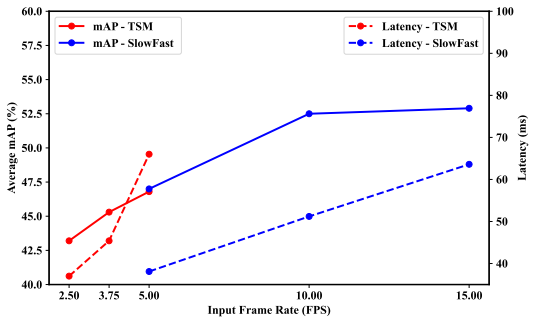

Figure 3. The effect of **input frame rate** on TAD performance (left Y-axis, solid lines) and latency (right Y-aixs, dashed lines) on THUMOS14. Red lines and blue lines are with TSM ResNet-50 encoder and SlowFast ResNet-50 encoder respectively.

performance on THUMOS14, as the external action classifier restricts the classification accuracy. Making class-aware predictions like the other two heads is likely to boost its performance.

(II) Computation cost: G-TAD has much higher FLOPs than the other two heads, as it generates dense anchors. It acounts for around 1/3 of the full network's latency. TadTR has the lowest latency as it outputs very sparse detections. Therefore, reducing the number of detections is a promising direction for building efficient detectors.

(III) Model size: AFSD has the smallest model size, only 66.7% that of TadTR. Therefore it is a better choice when a small model size is desired.

Video Encoders. Tab. 6 compares different encoders on THUMOS14 and ActivityNet. We observe that:

(I) While using a smaller backbone reduces the computation cost, it may severely downgrade the detection performance. For example, the performance of TSM with ResNet-18 is 7% lower than that with ResNet-50.

(II) **Motion information is important for temporal action detection**. The commonly used TSN encoder falls far behind the others, for lack of motion information modeling. It is even weaker than TSM ResNet-18, which models motion information but has a smaller backbone.

(III) TSM performs on par with I3D, another typical video encoder in TAD. Meanwhile, its latency is around half of I3D. We observe that the advantage of I3D lies in mAP at high IoU thresholds, as it uses a higher sampling frame rate. Therefore TSM is a desirable replacement for I3D when there is no strict demand on localization accuracy.

(IV) SlowFast achieves the best performance on both datasets. This is reasonable, as SlowFast is a state-of-the-art action recognition model. Its advantage is particularly large on THUMOS14, as the fast pathway can effectively model fast-changing motion, which helps to localize short actions on this dataset. Meanwhile, it is also efficient. It has lower FLOPs than TSM R50, TSN R50, and I3D. The incosistency between FLOPs and latency might be due to the low GPU utilization at low the video resolution.

Temporal Resolution. Fig. 3 compares the performance of TadTR using different input frame rates. We use temporal linear interpolation to ensure the output feature sequence has the same length. It is observed that increasing the input frame rate from 2.5 to 5 steadily improves the detection performance of TSM [19] on THUMOS14, where most actions instances are very short. Therefore, we switch the encoder to SlowFast [9], which performs as well as TSM at 5 FPS but runs much faster, owing to the efficiency of its fast pathway. The performance improves by a sizable margin as the frame rate increases to 10 FPS. We show in Fig. 4 that the increase is mainly from short actions. It indicates that a high frame rate is important for detecting short actions. Further increasing the frame rate does not bring a clear performance gain.

Image Resolution. Fig. 5 compares the performance with

Encoder	FLOPs	Latency	Param	THUMOS14						ActivityNet			
				0.3	0.4	0.5	0.6	0.7	Avg.	0.5	0.75	0.95	Avg.
TSM R18	**32.3G**	**25.7ms**	**24M**	52.8	47.9	39.8	30.7	20.3	38.3	49.12	34.00	9.74	33.42
TSM R50	73.2G	41.4ms	36M	60.5	55.5	47.5	37.6	25.3	45.3	49.59	34.74	9.72	34.14
TSN R50	73.2G	41.4ms	36M	44.2	39.6	31.9	22.9	13.7	30.5	48.97	33.26	7.84	32.65
I3D	125.6G	78.4ms	45M	59.6	54.5	47.0	37.8	26.5	45.1	49.56	35.24	9.93	34.35
SF R50	62.1G	63.5ms	46M	**69.4**	**64.3**	**56.0**	**46.4**	**34.9**	**54.2**	**50.13**	**35.78**	**10.52**	**35.10**

Table 6. Comparison of end-to-end trained detectors with different **video encoders**. FLOPs and latency are measured on ActivityNet.

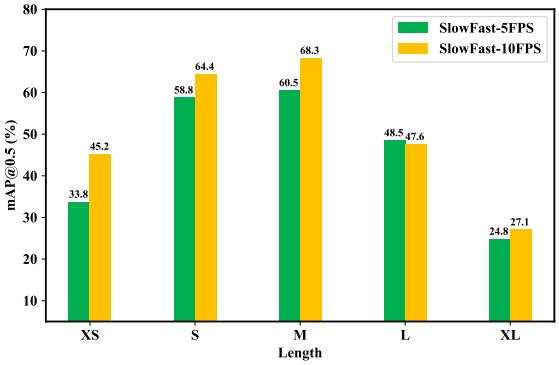

Figure 4. Increasing the input frame rate (from 5 FPS to 10 FPS) helps to detect short actions. Actions are divided into five groups according to their length (in seconds): XS (0, 3], S (3, 6], M (6, 12], L (12, 18], XL (18, inf). Detector: TadTR.

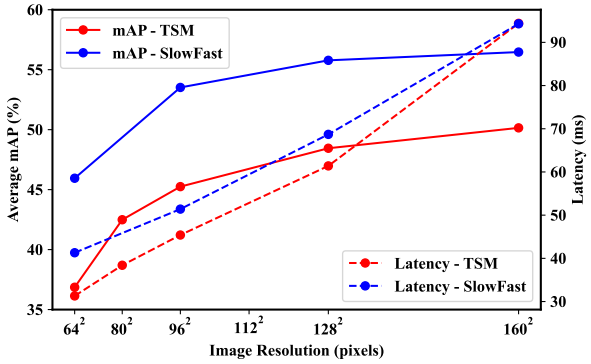

Figure 5. The effect of **image resolution** on THUMOS14. The input frame rate is set to 3.75 and 10 for TSM and SlowFast respectively. Detector: TadTR.

different input image resolution on THUMOS14. The slop of each line segment roughly represents the average performance gain per pixel. We observe that:

(I) Increasing image resolution boosts TAD performance, at the expense of efficiency. The improvement is especially large when the resolution increases from small (64^2) to medium (96^2). It indicates that a sufficient image resolution is critical for good performance. After that, the average performance gain per pixel gradually decreases. Therefore we choose the 64^2 resolution for a balance between performance and efficiency.

(II) Increasing image resolution is less important than switching to a more suitable video encoder. We find that SlowFast ResNet-50 encoder with 96^2 resolution outperforms TSM ResNet-50 encoder with 160^2 resolution.

Due to space limit, we put the analyses of the the effect of video resolution on ActivityNet **in the supplementary**. We also analyze the effects of the other two design choices, feature fusion and the frame sampling manner in it.

4.3. Comparison with State-of-the-art Methods

In the above study, we identify that SlowFast well balances between performance and accuracy and that TadTR is a strong and efficient action detection head. Here we combine them as a baseline detector for comparison with state-of-the-art methods. The default resolution is used.

Detection Performance. Tab. 7 compares the detection performance of different methods on THUMOS14 and ActivityNet. We divide them into two groups according to whether end-to-end training is used. Alghough S-CNN [35], CDC [34], and SSN [57] are multi-stage methods, we still regard them as end-to-end methods as the encoder and the head are jointly optimized in each stage. We observe that:

(I) On both datasets, the baseline detector achieves the best performance among end-to-end methods. This is a result of the better video encoder and the stronger detection head.

(II) Without optical flow, this detector surpasses those two-stream methods that are based on pre-trained features, such as MUSES [25] and VSGN [54]. Similarly, AFSD-RGB also outperforms many two-stream methods. It means that **optical flow is not necessary for TAD**, as the video encoders learn to capture cues of action boundaries from RGB frames via end-to-end training.

Computation Cost. In Tab. 1, we already compare the computation cost with of the state-of-the-art methods. Our presented detector has lower computation cost than previous end-to-end detector, as a result of the more efficient

Method	Encoder	Flow	THUMOS14						ActivityNet			
			0.3	0.4	0.5	0.6	0.7	Avg.	0.5	0.75	0.95	Avg.
Yeung *et al*. [51]	VGG16		36.0	26.4	17.1	-	-	-	-	-	-	-
TAL-Net [7]	I3D	✓	53.2	48.5	42.8	33.8	20.8	39.8	38.23	18.30	1.30	20.22
BSN [22]	TSN	✓	53.5	45	36.9	28.4	20	-	46.45	29.96	8.02	30.03
BMN [20]	TSN	✓	56.0	47.4	38.8	29.7	20.5	38.5	50.07	34.7	8.29	33.85
G-TAD [48]	TSN	✓	54.5	47.6	40.2	30.8	23.4	39.3	50.36	34.60	9.02	34.09
BC-GNN [1]	TSN	✓	57.1	49.1	40.4	31.2	23.1	40.2	50.56	34.75	9.37	34.26
A2Net [50]	I3D	✓	58.6	54.1	45.5	32.5	17.2	41.6	43.55	28.69	3.70	27.75
P-GCN [53]	I3D	✓	63.6	57.8	49.1	-	-	-	48.26	33.16	3.27	31.11
MUSES [25]	I3D	✓	68.9	64.0	**56.9**	46.3	31.0	53.4	50.02	34.97	6.57	33.99
VSGN [54]	TSN	✓	66.7	60.4	52.4	41.0	30.4	50.2	52.38	**36.01**	8.37	35.07
S-CNN [35]	C3D		36.3	28.7	19	10.3	5.3	-	-	-	-	-
R-C3D [45]	C3D		44.8	35.6	28.9	-	-	-	26.80	-	-	-
SS-TAD [2]	C3D		45.7	–	29.2	–	9.6	-	-	-	-	-
CDC [34]	C3D		40.1	29.4	23.3	13.1	7.9	22.8	45.3	26.0	0.2	23.8
SSN [58]	TSN	✓	51.9	41.0	29.8	-	-	-	-	-	-	-
GTAN [28]	P3D		57.8	47.2	38.8	-	-	-	52.61	34.14	8.91	34.31
PBRNet [23]	I3D	✓	58.5	54.6	51.3	41.8	29.5	47.1	**53.96**	34.97	8.98	35.01
AFSD [18]	I3D	✓	67.3	62.4	55.5	43.7	31.1	52.0	52.38	35.27	6.47	34.39
AFSD-RGB [18]	I3D		57.7	52.8	45.4	34.9	22.0	43.6	-	-	-	32.90
Ours	SF R50		**69.4**	**64.3**	56.0	**46.4**	**34.9**	**54.2**	50.47	35.99	**10.83**	**35.10**

Table 7. State-of-the-art comparison in terms mAP at different thresholds. Only the methods in the second group are end-to-end trained.

video encoder and detection head. Compared with the state-of-the-art method [25] that is based on pre-trained features, the baseline runs 126× faster. We analyze the reason for the huge difference between their computation costs **in the supplementary**.

Besides, we compare the inference speed in terms of inference FPS in Tab. 8. Note that this metric has a bias. It is more favorable for methods that use a high input frame rate (*e.g*., 25 in R-C3D [45]). Therefore we also report the speedup ratio, *i.e*. the ratio of inference FPS to the input frame rate. Our detector runs at 5076 FPS and has a speedup ratio of 508, which is much faster than the other end-to-end methods.

Model	GPU	Infer. FPS	Speedup
S-CNN [35]	-	60	-
CDC [34]	TITAN Xm	500	-
SS-TAD [2]	TITAN Xm	701	23
R-C3D [45]	TITAN Xm	569	23
R-C3D [45]	TITAN Xp	1030	45
AFSD [18]	TITAN Xp	3403*	340*
Ours	TITAN Xp	5076	508

Table 8. Comparison of the inference speed, measure by the number of processed frames per second (FPS) and the speedup ratio. *Only measure the RGB network.

5. Conclusion

We conduct an empirical study of end-to-end temporal action detection. We show that end-to-end training gives rise to much better performance than the traditional head-only learning paradigm, where the video encoder is only optimized for action recognition. We also study multiple factors that affect the performance and accuracy of end-to-end temporal action detection to seek a efficiency-accuracy trade-off. Based on our findings, we build a mid-resolution detector that outperforms previous end-to-end methods while running more than 4× faster. It is also encouraging that the detector surpasses the previous two-stream models without optical flow. The results show that end-to-end learning is a promising direction for building strong and efficient TAD models. Hopefully, this work can serve as a useful reference guide for end-to-end training and inspire future research.

Limitation. End-to-end learning may still restrict the use of stronger video encoders, higher video resolution due to the constraint of GPU memory. In the future, we plan to explore the complementarity of end-to-end learned features with pre-trained features to address this limitation.

Acknowledgement. This work was supported by National Key R&D Program of China (No. 2018YFB1004600).

References

[1] 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*, pages 121–137, 2020. 8

[2] Shyamal Buch, Victor Escorcia, Bernard Ghanem, Li Fei-Fei, and Juan Carlos Niebles. End-to-end, single-stream temporal action detection in untrimmed videos. In *BMVC*, 2017. 2, 8

[3] Shyamal Buch, Victor Escorcia, Chuanqi Shen, Bernard Ghanem, and Juan Carlos Niebles. Sst: Single-stream temporal action proposals. In *CVPR*, pages 6373–6382, 2017. 2

[4] Fabian Caba Heilbron, Victor Escorcia, Bernard Ghanem, and Juan Carlos Niebles. Activitynet: A large-scale video benchmark for human activity understanding. In *CVPR*, pages 961–970, 2015. 4

[5] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *ECCV*, pages 213–229, 2020. 2

[6] Joao Carreira and Andrew Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *CVPR*, pages 4724–4733, 2017. 1, 2, 3

[7] Yu-Wei Chao, Sudheendra Vijayanarasimhan, Bryan Seybold, David A Ross, Jia Deng, and Rahul Sukthankar. Rethinking the faster r-cnn architecture for temporal action localization. In *CVPR*, pages 1130–1139, 2018. 2, 8

[8] Victor Escorcia, Fabian Caba Heilbron, Juan Carlos Niebles, and Bernard Ghanem. Daps: Deep action proposals for action understanding. In *ECCV*, pages 768–784, 2016. 2

[9] Christoph Feichtenhofer, Haoqi Fan, Jitendra Malik, and Kaiming He. Slowfast networks for video recognition. In *ICCV*, pages 6202–6211, 2019. 2, 3, 6

[10] Jiyang Gao, Kan Chen, and Ram Nevatia. Ctap: Complementary temporal action proposal generation. In *ECCV*, September 2018. 2

[11] Jiyang Gao, Zhenheng Yang, Chen Sun, Kan Chen, and Ram Nevatia. Turn tap: Temporal unit regression network for temporal action proposals. In *ICCV*, pages 3648–3656, 2017. 2

[12] Ross Girshick. Fast r-cnn. In *ICCV*, December 2015. 2

[13] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *ICCV*, pages 2961–2969, 2017. 3

[14] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, pages 448–456, 2015. 3

[15] YG Jiang, Jingen Liu, A Roshan Zamir, G Toderici, I Laptev, Mubarak Shah, and Rahul Sukthankar. Thumos challenge: Action recognition with a large number of classes, 2014. 4

[16] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015. 4

[17] Zhihui Li and Lina Yao. Three birds with one stone: Multi-task temporal action detection via recycling temporal annotations. In *CVPR*, pages 4751–4760, 2021. 2

[18] 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*, pages 3320–3329, 2021. 1, 2, 3, 4, 5, 6, 8

[19] Ji Lin, Chuang Gan, and Song Han. Tsm: Temporal shift module for efficient video understanding. In *ICCV*, pages 7083–7093, 2019. 2, 3, 6

[20] Tianwei Lin, Xiao Liu, Xin Li, Errui Ding, and Shilei Wen. Bmn: Boundary-matching network for temporal action proposal generation. In *ICCV*, pages 3889–3898, 2019. 1, 2, 4, 8

[21] Tianwei Lin, Xu Zhao, and Zheng Shou. Single shot temporal action detection. In *ACM MM*, pages 988–996, 2017. 2

[22] Tianwei Lin, Xu Zhao, Haisheng Su, Chongjing Wang, and Ming Yang. Bsn: Boundary sensitive network for temporal action proposal generation. In *ECCV*, September 2018. 2, 3, 4, 8

[23] Qinying Liu and Zilei Wang. Progressive boundary refinement network for temporal action detection. In *AAAI*, volume 34, pages 11612–11619, 2020. 8

[24] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *ECCV*, pages 21–37. Springer, 2016. 4

[25] Xiaolong Liu, Yao Hu, Song Bai, Fei Ding, Xiang Bai, and Philip H. S. Torr. Multi-shot temporal event localization: A benchmark. In *CVPR*, pages 12596–12606, June 2021. 1, 7, 8

[26] Xiaolong Liu, Qimeng Wang, Yao Hu, Xu Tang, Song Bai, and Xiang Bai. End-to-end temporal action detection with transformer. *arXiv preprint arXiv:2106.10271*, 2021. 2, 3

[27] Yuan Liu, Lin Ma, Yifeng Zhang, Wei Liu, and Shih-Fu Chang. Multi-granularity generator for temporal action proposal. In *CVPR*, pages 3604–3613, 2019. 2

[28] Fuchen Long, Ting Yao, Zhaofan Qiu, Xinmei Tian, Jiebo Luo, and Tao Mei. Gaussian temporal awareness networks for action localization. In *CVPR*, pages 344–353, 2019. 2, 3, 4, 8

[29] AJ Piergiovanni and Michael S. Ryoo. Temporal gaussian mixture layer for videos. In *ICML*, 2019. 3

[30] 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*, pages 485–494, 2021. 2

[31] Han Qiu, Yuchen Ma, Zeming Li, Songtao Liu, and Jian Sun. Borderdet: Border feature for dense object detection. In *ECCV*, pages 549–564. Springer, 2020. 3

[32] Zhaofan Qiu, Ting Yao, and Tao Mei. Learning spatio-temporal representation with pseudo-3d residual networks. In *ICCV*, pages 5533–5541, 2017. 2, 3

[33] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *NIPS*, pages 91–99, 2015. 2

[34] Zheng Shou, Jonathan Chan, Alireza Zareian, Kazuyuki Miyazawa, and Shih-Fu Chang. Cdc: Convolutional-de-convolutional networks for precise temporal action localization in untrimmed videos. In *ICCV*, pages 1417–1426, 2017. 2, 7, 8

[35] Zheng Shou, Dongang Wang, and Shih-Fu Chang. Temporal action localization in untrimmed videos via multi-stage cnns. In *CVPR*, pages 1049–1058, 2016. 2, 7, 8

[36] Karen Simonyan and Andrew Zisserman. Two-stream convolutional networks for action recognition in videos. In *NIPS*, pages 568–576, 2014. 2

[37] Deepak Sridhar, Niamul Quader, Srikanth Muralidharan, Yaoxin Li, Peng Dai, and Juwei Lu. Class semantics-based attention for action detection. In *ICCV*, pages 13739–13748, 2021. 2

[38] Jing Tan, Jiaqi Tang, Limin Wang, and Gangshan Wu. Relaxed transformer decoders for direct action proposal generation. In *ICCV*, pages 13526–13535, October 2021. 2

[39] Zhi Tian, Chunhua Shen, Hao Chen, and Tong He. Fcos: Fully convolutional one-stage object detection. In *CVPR*, pages 9627–9636, 2019. 2, 3

[40] Du Tran, Lubomir Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. Learning spatiotemporal features with 3d convolutional networks. In *ICCV*, pages 4489–4497, 2015. 2

[41] Du Tran, Heng Wang, Lorenzo Torresani, Jamie Ray, Yann LeCun, and Manohar Paluri. A closer look at spatiotemporal convolutions for action recognition. In *CVPR*, pages 6450–6459, 2018. 2, 3

[42] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NIPS*, pages 5998–6008, 2017. 2, 3

[43] Limin Wang, Yuanjun Xiong, Zhe Wang, Yu Qiao, Dahua Lin, Xiaoou Tang, and Luc Van Gool. Temporal segment networks: Towards good practices for deep action recognition. In *ECCV*, pages 20–36, 2016. 2, 3

[44] Saining Xie, Chen Sun, Jonathan Huang, Zhuowen Tu, and Kevin Murphy. Rethinking spatiotemporal feature learning: Speed-accuracy trade-offs in video classification. In *ECCV*, pages 305–321, 2018. 2

[45] Huijuan Xu, Abir Das, and Kate Saenko. R-c3d: region convolutional 3d network for temporal activity detection. In *ICCV*, pages 5794–5803, 2017. 2, 8

[46] Mengmeng Xu, Juan-Manuel Pérez-Rúa, Victor Escorcia, Brais Martinez, Xiatian Zhu, Li Zhang, Bernard Ghanem, and Tao Xiang. Boundary-sensitive pre-training for temporal localization in videos. In *ICCV*, pages 7220–7230, 2021. 3

[47] Mengmeng Xu, Juan-Manuel Perez-Rua, Xiatian Zhu, Bernard Ghanem, and Brais Martinez. Low-fidelity video encoder optimization for temporal action localization. In *NeurIPS*, 2021. 4

[48] Mengmeng Xu, Chen Zhao, David S Rojas, Ali Thabet, and Bernard Ghanem. G-TAD: Sub-graph localization for temporal action detection. In *CVPR*, pages 10156–10165, 2020. 1, 3, 4, 5, 8

[49] Le Yang, Junwei Han, Tao Zhao, Tianwei Lin, Dingwen Zhang, and Jianxin Chen. Background-click supervision for temporal action localization. *TPAMI*, 2021. 1

[50] 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, 8

[51] Serena Yeung, Olga Russakovsky, Greg Mori, and Li Fei-Fei. End-to-end learning of action detection from frame glimpses in videos. In *CVPR*, pages 2678–2687, 2016. 8

[52] Ze-Huan Yuan, Jonathan C Stroud, Tong Lu, and Jia Deng. Temporal action localization by structured maximal sums. In *CVPR*, volume 2, page 7, 2017. 2

[53] Runhao Zeng, Wenbing Huang, Mingkui Tan, Yu Rong, Peilin Zhao, Junzhou Huang, and Chuang Gan. Graph convolutional networks for temporal action localization. In *ICCV*, pages 7094–7103, 2019. 1, 2, 8

[54] Chen Zhao, Ali K Thabet, and Bernard Ghanem. Video self-stitching graph network for temporal action localization. In *ICCV*, pages 13658–13667, 2021. 2, 4, 7, 8

[55] Hang Zhao, Antonio Torralba, Lorenzo Torresani, and Zhicheng Yan. HACS: human action clips and segments dataset for recognition and temporal localization. In *ICCV*, pages 8667–8677, 2019. 5

[56] Peisen Zhao, Lingxi Xie, Chen Ju, Ya Zhang, Yanfeng Wang, and Qi Tian. Bottom-up temporal action localization with mutual regularization. In *ECCV*, 2020. 2

[57] Yue Zhao, Yuanjun Xiong, Limin Wang, Zhirong Wu, Xiaoou Tang, and Dahua Lin. Temporal action detection with structured segment networks. *ICCV*, pages 2914–2923, 2017. 2, 7

[58] Yue Zhao, Bowen Zhang, Zhirong Wu, Shuo Yang, Lei Zhou, Sijie Yan, Limin Wang, Yuanjun Xiong, Wang Yali, Dahua Lin, Yu Qiao, and Xiaoou Tang. CUHK & ETHZ & SIAT submission to ActivityNet challenge 2017. *arXiv preprint arXiv:1710.08011*, pages 20–24, 2017. 4, 8

[59] Zixin Zhu, Wei Tang, Le Wang, Nanning Zheng, and Gang Hua. Enriching local and global contexts for temporal action localization. In *ICCV*, pages 13516–13525, 2021. 2