ActionFormer: Localizing Moments of Actions with Transformers

Chen-Lin Zhang[⋆,1,2], Jianxin Wu[1], and Yin Li[3]

[1] State Key Laboratory for Novel Software Technology, Nanjing University, China
[2] 4Paradigm Inc, Beijing, China
[3] University of Wisconsin-Madison, USA
{zclnjucs, wujx2001}@gmail.com   yin.li@wisc.edu

Abstract. Self-attention based Transformer models have demonstrated impressive results for image classification and object detection, and more recently for video understanding. Inspired by this success, we investigate the application of Transformer networks for temporal action localization in videos. To this end, we present ActionFormer—a simple yet powerful model to identify actions in time and recognize their categories in a single shot, without using action proposals or relying on pre-defined anchor windows. ActionFormer combines a multiscale feature representation with local self-attention, and uses a light-weighted decoder to classify every moment in time and estimate the corresponding action boundaries. We show that this orchestrated design results in major improvements upon prior works. Without bells and whistles, ActionFormer achieves 71.0% mAP at tIoU=0.5 on THUMOS14, outperforming the best prior model by 14.1 absolute percentage points. Further, ActionFormer demonstrates strong results on ActivityNet 1.3 (36.6% average mAP) and EPIC-Kitchens 100 (+13.5% average mAP over prior works). Our code is available at https://github.com/happyharrycn/actionformer_release.

Keywords: temporal action localization; action recognition; egocentric vision; vision transformers; video understanding

1 Introduction

Identifying action instances in time and recognizing their categories, known as temporal action localization (TAL), remains a challenging problem in video understanding. Significant progress has been made in developing deep models for TAL. Most previous works have considered using action proposals [36] or anchor windows [50], and developed convolutional [85,56], recurrent [7], and graph [4,75,78] neural networks for TAL. Despite a steady progress on major benchmarks, the accuracy of existing methods usually comes at a price of modeling complexity, with increasingly sophisticated proposal generation, anchor design, loss function, network architecture, and output decoding process.

⋆ Work was done when visiting UW Madison.
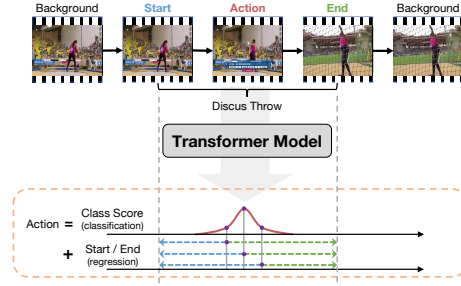
Fig. 1. An illustration of our Action-Former. We propose a Transformer based model to localize action instances in time (*top*) by (1) classifying every moment into action categories and (2) estimating their distances to action boundaries (*bottom*).

In this paper, we adopt a minimalist design and develop a Transformer based model for TAL, inspired by the recent success of Transformers in NLP [64,19] and vision [20,48,11]. Originally developed for sequence data, Transformers use self-attention to model long-range dependencies, and thus are a natural fit for TAL in untrimmed videos. Our method, illustrated in Fig. 1, adapts local self-attention to model temporal context in an input untrimmed videos, classifies every moment, and regresses their corresponding action boundaries. The result is a deep model trained using standard classification and regression loss, and can localize moments of actions in a single shot, without using action proposals or pre-defined anchor windows.

Specifically, our model, dubbed ActionFormer, integrates local self-attention to extract a feature pyramid from an input video. Each location in the output pyramid represents a moment in the video, and is treated as an action candidate. A lightweight convolutional decoder is further employed on the feature pyramid to classify these candidates into foreground action categories, and to regress the distance between a foreground candidate and its action onset and offset. The results can be easily decoded into actions with their labels and temporal boundaries. Our method thus provides a *single-stage anchor-free* model for TAL.

We show that such a simple model, with proper design, can be surprisingly powerful for TAL. In particular, ActionFormer establishes a new state of the art across several major TAL benchmarks, surpassing previous works by a significant margin. For example, ActionFormer achieves 71.0% mAP at tIoU=0.5 on THU-MOS14, outperforming the best prior model by 14.1 absolute percentage points. Further, ActionFormer reaches an average mAP of 36.6% on ActivityNet 1.3. More importantly, ActionFormer shows impressive results on EPIC-Kitchens 100 for egocentric action localization, with a boost of over 13.5 absolute percentage points in average mAP.

Our work is based on simple techniques, supported by favourable empirical results, and validated by extensive ablation experiments, at our best. Our main contributions are summarized as follows. First, we are among the first to propose a Transformer based model for single-stage anchor-free TAL. Second, we study key design choices of developing Transformer models for TAL, and demonstrate a simple model that works surprisingly well. Finally, our model achieves state-of-the-art results across major benchmarks and offers a solid baseline for TAL.

2   Related Works

Two-stage TAL. These approaches first generate candidate video segments as action proposals, and further classify the proposals into actions and refine their temporal boundaries. Previous works focused on action proposal generation, by either classifying anchor windows [9,22,8] or detecting action boundaries [38,36,47,26,84], and more recently using a graph representation [4,75] or Transformers [59,13,66]. Others have integrated proposal generation and classification into a single model [57,85,56,14]. More recent effort investigates the modeling of temporal context among proposals using graph neural networks [80,75,83] or attention and self-attention mechanisms [89,52,58]. Similar to previous approaches, our method considers the modeling of long-term temporal context, yet uses a self-attention within a Transformer model. Different from previous approaches, our model detects actions without using proposals.

Single-stage TAL. Several recent works focused on single-stage TAL, seeking to localize actions in a single shot without using action proposals. Many of them are anchor-based (*e.g.*, using anchor windows sampled from sliding windows). Lin *et al.* [37] presented the first single-stage TAL using convolutional networks, borrowing ideas from a single-stage object detector [44]. Buch *et al.* [7] presented a recurrent memory module for single-stage TAL. Long *et al.* [50] proposed to use Gaussian kernels to dynamically optimize the scale of each anchor, based on a 1D convolutional network. Yang *et al.* [77] explored the combination of anchor-based and anchor-free models for single-stage TAL, again using convolutional networks. More recently, Lin *et al.* [35] proposed an anchor-free single-stage model by designing a saliency-based refinement module incorporated in convolutional network. Similar ideas were also explored in video grounding [81].

Our model falls into the category of single-stage TAL. Indeed, our formulation follows a minimalist design of sequence labeling by classifying every moment and regressing their action boundaries, previously discussed in [77,35]. The key difference is that we design a Transformer network for action localization. The result is a single stage anchor-free model that outperforms all previous methods. A concurrent work from Liu *et al.* [46] also used Transformer for TAL, yet considered a set prediction problem similar to DETR [11].

Spatial-temporal Action Localization. A related yet different task, known as spatial-temporal action localization, is to detect the actions both temporally and spatially, in the form of moving bounding boxes of an actor. It is possible that TAL might be used as a first step for spatial-temporal localization. Girdhar *et al.* [25] proposed to use Transformer for spatial-temporal action localization. While both our work and [25] use Transformer, the two models differ significantly. We consider a sequence of video frames as the inputs, while [25] used a set of 2D object proposals. Moreover, our work addresses a different task of TAL.

Object Detection. TAL models have been heavily influenced by the developments of object detection models. Some of our model design, including the multiscale feature representation and convolutional decoder, is inspired by feature

pyramid network [39] and RetinaNet [40]. Our training using center sampling also stems from recent single-stage object detectors [21,60,82].

Vision Transformer. Transformer models were originally developed for NLP tasks [64], and has demonstrated recent success for many vision tasks. ViT [20] presented the first pure Transformer-based model that can achieve state-of-the-art performances on image classification. Subsequent works, including DeiT [61], T2T-ViT [79], Swin Transformer [48], Focal Transformer [76] and PVT [69], have further pushed the envelope, resulting in vision Transformer backbones with impressive results on classification, segmentation, and detection tasks. Transformer have also been explored in object detection [11,88,17,68], semantic segmentation [70,72,15], and video representation learning [3,49,23]. Our model builds on these developments and presents one of the first Transformer models for TAL.

3   ActionFormer: A Simple Transformer Model for Temporal Action Localization

Given an input video $\mathbf{X}$, we assume that $\mathbf{X}$ can be represented using a set of feature vectors $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_T\}$ defined on discretized time steps $t = \{1, 2, \ldots, T\}$, where the total duration T varies across videos. For example, $\mathbf{x}_t$ can be the feature vector of a video clip at moment t extracted from a 3D convolutional network. The goal of temporal action localization is to predict the action label $\mathbf{Y} = \{\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_N\}$ based on the input video sequence $\mathbf{X}$. $\mathbf{Y}$ consists of N action instances $\mathbf{y}_i$, where N also varies across videos. Each instance $\mathbf{y}_i = (s_i, e_i, a_i)$ is defined by its starting time s_i (onset), ending time e_i (offset) and its action label a_i, where $s_i \in [1, T]$, $e_i \in [1, T]$, $a_i \in \{1, .., C\}$ (C pre-defined categories) and $s_i < e_i$. The task of TAL is thus a challenging problem of structured output prediction.

A Simple Representation for Action Localization. Our method builds on an anchor-free representation for action localization, inspired by [77,35]. The key idea is to classify each moment as either one of the action categories or the background, and further regress the distance between this time step and the action's onset and offset. In doing so, we convert the structured output prediction problem ($\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_T\} \rightarrow \mathbf{Y} = \{\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_N\}$) into a more approachable sequence labeling problem

$$\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_T\} \rightarrow \hat{\mathbf{Y}} = \{\hat{\mathbf{y}}_1, \hat{\mathbf{y}}_2, ..., \hat{\mathbf{y}}_T\}. \tag{1}$$

The output $\hat{\mathbf{y}}_t = (p(a_t), d_t^s, d_t^e)$ at time t is defined as

- $p(a_t)$ consists of C values, with each representing a binomial variable indicating the probability of action category a_t ($\in \{1, 2, \ldots, C\}$) at time t. This can be considered as the outputs of C binary classification.
- $d_t^s > 0$ and $d_t^e > 0$ are the distance between the current time t to the action's onset and offset, respectively. d_t^s and d_t^e are not defined if the time step t lies on the background.

Transformer模型最早是用在NLP任务中，后来被证明在视觉任务中也有良好的表现。纯基于Transformer的模型在图像分类上已经取得了最先进得效果。随后Transformer被用作视觉任务的主干网络，在分类、分割、检测上取得了成功。Transformer已经被用在目标检测，语义分割，视频表示学习。
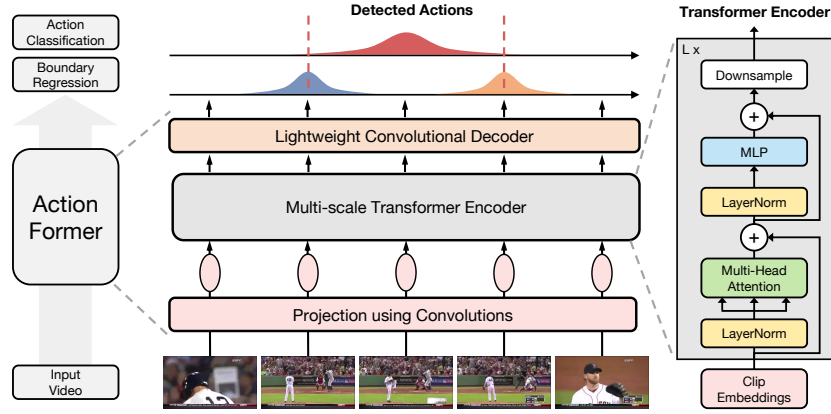
Fig. 2. Overview of our ActionFormer. Our method builds a Transformer based model to detect an action instance by classifying every moment and estimating action boundaries. Specifically, ActionFormer first extracts a sequence of video clip features, and embeds each of these features. The embedded features are further encoded into a feature pyramid using a multi-scale Transformer (right). The feature pyramid is then examined by shared classification and regression heads, producing an action candidate at every time step. Our method provides a single-stage anchor-free model for temporal action localization with strong performance across several datasets.

Intuitively, this formulation considers *every moment t* in the video $\mathbf{X}$ as an action candidate, recognizes the action's category a_t, and estimates the distances between current step and the action boundaries (d_t^s and d_t^e) if an action presents. Action localization results can be readily decoded from $\hat{\mathbf{y}}_t = (p(a_t), d_t^s, d_t^e)$ by

$$a_t = \arg\max \; p(a_t), \quad s_t = t - d_t^s, \quad e_t = t + d_t^e. \tag{2}$$

Method Overview. Our model — ActionFormer learns to label an input video sequence $f(\mathbf{X}) \to \hat{\mathbf{Y}}$. Specifically, f is realized using a deep model. ActionFormer follows an encoder-decoder architecture proven successful in many vision tasks, and decomposes f as $h \circ g$. Here $g\colon \mathbf{X} \to \mathbf{Z}$ encodes the input into a latent vector $\mathbf{Z}$, and $h\colon \mathbf{Z} \to \hat{\mathbf{Y}}$ subsequently decodes $\mathbf{Z}$ into the sequence label $\hat{\mathbf{Y}}$.

Fig. 2 presents an overview of our model. Importantly, our encoder g is parameterized by a Transformer network [64]. Our decoder h adopts a lightweight convolutional network. To capture actions at various temporal scales, we design a multi-scale feature representation $\mathbf{Z} = \{\mathbf{Z}^1, \mathbf{Z}^2, \ldots, \mathbf{Z}^L\}$ forming a feature pyramid with varying resolutions. Note that our model operates on a temporal axis defined by feature grids rather than the absolute time, allowing it to adapt to videos with different frame rates. We now describe the details of our model.

3.1 Encode Videos with Transformer

Our model first encodes an input video $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_T\}$ into a multiscale feature representation $\mathbf{Z} = \{\mathbf{Z}^1, \mathbf{Z}^2, \ldots, \mathbf{Z}^L\}$ using an encoder g. The encoder g

consists of (1) a projection function using a convolutional network that embeds each feature $(\mathbf{x}_t)$ into a D-dimensional space; and (2) a Transformer network that maps the embedded features to the output feature pyramid $\mathbf{Z}$.

Projection. Our projection $\mathbf{E}$ is a shallow convolutional network with ReLU as the activation function, defined as

$$\mathbf{Z}^0 = [\mathbf{E}(\mathbf{x}_1), \mathbf{E}(\mathbf{x}_2), \ldots, \mathbf{E}(\mathbf{x}_T)]^T, \tag{3}$$

where $\mathbf{E}(\mathbf{x}_i) \in \mathbb{R}^D$ is the embedded feature of $\mathbf{x}_i$. Adding convolutions before a Transformer network was recently found helpful to better incorporate local context for time series data [32] and to stabilize the training of vision Transformers [71]. An position embedding [64] $\mathbf{E}_{pos} \in \mathbb{R}^{T \times D}$ can be optionally added. However, we find that doing so will decrease the performance of the model, and have thus removed position embeddings in our model by default.

Local Self-Attention. The Transformer network further takes $\mathbf{Z}^0$ as input. The core of a Transformer is self-attention [64]. We briefly introduce the key idea to make the paper self-contained. Concretely, self-attention computes a weighted average of features with the weight proportional to a similarity score between pairs of input features. Given $\mathbf{Z}^0 \in \mathbb{R}^{T \times D}$ with T time steps of D dimensional features, $\mathbf{Z}^0$ is projected using $\mathbf{W}_Q \in \mathbb{R}^{D \times D_q}$, $\mathbf{W}_K \in \mathbb{R}^{D \times D_k}$, and $\mathbf{W}_V \in \mathbb{R}^{D \times D_v}$ to extract feature representations $\mathbf{Q}$, $\mathbf{K}$, and $\mathbf{V}$, referred to as query, key and value respectively with $D_k = D_q$. The outputs $\mathbf{Q}$, $\mathbf{K}$, $\mathbf{V}$ are computed as

$$\mathbf{Q} = \mathbf{Z^0}\mathbf{W}_Q, \quad \mathbf{K} = \mathbf{Z^0}\mathbf{W}_K, \quad \mathbf{V} = \mathbf{Z^0}\mathbf{W}_V. \tag{4}$$

The output of self-attention is given by

$$\mathbf{S} = \text{softmax}\left(\mathbf{Q}\mathbf{K}^T / \sqrt{D_q}\right)\mathbf{V}, \tag{5}$$

where $\mathbf{S} \in \mathbb{R}^{T \times D}$ and softmax is performed *row-wise*. A multiheaded self-attention (MSA) further adds several self-attention operations in parallel.

A main advantage of MSA is the ability to integrate temporal context across the full sequence, yet such a benefit comes at the cost of computation. A vanilla MSA has a complexity of $O(T^2D + D^2T)$ in both memory and time, and is thus highly inefficient for long videos. There has been several recent work on efficient self-attention [73,5,67,16]. Here we adapt the local self-attention from [16] by limiting the attention within a local window. Our intuition is the temporal context beyond a certain range is less helpful for action localization. Such a local self-attention significantly reduces the complexity to $O(W^2TD + D^2T)$ with W the local window size ($\ll T$). Importantly, local self-attention is used in tandem with the multiscale feature representation $\mathbf{Z} = \{\mathbf{Z}^1, \mathbf{Z}^2, \ldots, \mathbf{Z}^L\}$, using the same window size on each pyramid level. With this design, a small window size (19) on a downsampled feature map (16x) will cover a large temporal range (304).

Multiscale Transformer. We now present the design of our Transformer encoder. Our Transformer has L Transformer layers with each layer consisting of alternating layers of local multiheaded self-attention (MSA) and MLP blocks.

Moreover, LayerNorm (LN) is applied before every MSA or MLP block, and residual connection is added after every block. GELU is used for the MLP. To capture actions at different temporal scales, a downsampling operator $\downarrow(\cdot)$ is optionally attached. This is given by

$$\bar{\mathbf{Z}}^\ell = \alpha^\ell \, \mathrm{MSA}(\mathrm{LN}(\mathbf{Z}^{\ell-1})) + \mathbf{Z}^{\ell-1}, \quad \hat{\mathbf{Z}}^\ell = \bar{\alpha}^\ell \, \mathrm{MLP}(\mathrm{LN}(\bar{\mathbf{Z}}^\ell)) + \bar{\mathbf{Z}}^\ell,$$
$$\mathbf{Z}^\ell = \downarrow(\hat{\mathbf{Z}}^\ell), \quad \ell = 1 \ldots L, \tag{6}$$

where $\mathbf{Z}^{\ell-1}, \bar{\mathbf{Z}}^\ell, \hat{\mathbf{Z}}^\ell \in \mathbb{R}^{T^{\ell-1} \times D}$ and $\mathbf{Z}^\ell \in \mathbb{R}^{T^\ell \times D}$. $T^{\ell-1}/T^\ell$ is the downsampling ratio. α^ℓ and $\bar{\alpha}^\ell$ are learnable per-channel scaling factors as in [62].

The downsampling operator $\downarrow$ is implemented using a strided depthwise 1D convolution due to its efficiency. We use 2x downsampling for our model. Our Transformer block is shown in Fig. 2 (right). Our model further combines several Transformer blocks with downsampling in between, resulting in a feature pyramid $\mathbf{Z} = \{\mathbf{Z}^1, \mathbf{Z}^2, \ldots, \mathbf{Z}^L\}$.

3.2   Decoding Actions in Time

Next, our model decodes the feature pyramid $\mathbf{Z}$ from the encoder g into the sequence label $\hat{\mathbf{Y}} = \{\hat{\mathbf{y}}_1, \hat{\mathbf{y}}_2, \ldots, \hat{\mathbf{y}}_T\}$ using the decoder h. Our decoder is a lightweight convolutional network with a classification and a regression head.

Classification Head. Given the feature pyramid Z, our classification head examines each moment t across all L levels on the pyramid, and predicts the probability of action $p(a_t)$ at every moment t.[4] This is realized using a lightweight 1D convolutional network attached to each pyramid level with its parameters shared across all levels. Our classification network is implemented using 3 layers of 1D convolutions with kernel size=3, layer normalization (for the first 2 layers), and ReLU activation. A sigmoid function is attached to each output dimension to predict the probability of C action categories. Adding layer normalization slightly boosts the performance as we will demonstrate in our ablation.

Regression Head. Similar to our classification head, our regression head examines every moment t across all L levels on the pyramid. The difference is that the regression head predicts the distances to the onset and offset of an action (d_t^s, d_t^e), only if the current time step t lies in an action. An output regression range is pre-specified for each pyramid level. The regression head, again, is implemented using a 1D convolutional network following the same design of the classification network, except that a ReLU is attached at the end for distance estimation.

3.3   ActionFormer: Model Design

Putting things together, ActionFormer is conceptually simple: each feature on the feature pyramid $\mathbf{Z}$ outputs an action score $p(a)$ and the corresponding temporal boundaries (s, e), which are then used to decode an action candidate.

---

[4] Without loss of clarity, we drop the index of the pyramid ℓ.

Notwithstanding the simplicity, we find that several key architecture designs are important to ensure a strong performance. We discuss these design choices here.

Design of the Feature Pyramid. A critical component of our model is the design of the temporal feature pyramid $\mathbf{Z} = \{\mathbf{Z}^1, \mathbf{Z}^2, \dots, \mathbf{Z}^L\}$. The design choices include (1) the number of levels within the pyramid; (2) the downsampling ratio between successive feature maps; and (3) the output regression range of each pyramid level. Inspired by the design of feature pyramid in modern object detectors (FPN [39] and FCOS [60]), we simplify our design choices by using a 2x downsampling of the feature maps, and roughly enlarging the output regression range by 2 accordingly. We explore different design choices in our ablation.

Loss Function. Our model outputs $(p(a_t), d_t^s, d_t^e)$ for every moment t, including the probability of action categories $p(a_t)$ and the distances to action boundaries (d_t^s, d_t^e). Our loss function, again following minimalist design, only has two terms: (1) $\mathcal{L}_{cls}$ a focal loss [40] for C way binary classification; and (2) $\mathcal{L}_{reg}$ a DIoU loss [87] for distance regression. The loss is defined for each video X as

$$\mathcal{L} = \sum_t \left(\mathcal{L}_{cls} + \lambda_{reg} \mathbb{1}_{c_t} \mathcal{L}_{reg} \right) / T_+, \tag{7}$$

where T_+ is the total number of positive samples. $\mathbb{1}_{c_t}$ is an indicator function that denotes if a time step t is within an action, *i.e.*, a positive sample. $\mathcal{L}$ is applied to all levels on the output pyramid, and averaged across all video samples during training. λ_{reg} is a coefficient balancing the classification and regression loss. We set $\lambda_{reg}{=}1$ by default and study the choice of λ_{reg} in our ablation.

Importantly, $\mathcal{L}_{cls}$ uses Focal loss [60] to recognize C action categories. Focal loss naturally handles imbalanced samples — there are much more negative samples than positive ones. Moreover, $\mathcal{L}_{reg}$ adopts a differentiable IoU loss [55]. $\mathcal{L}_{reg}$ is only enabled when the current time step contains a positive sample.

Center Sampling. During training, we find it helpful to adapt a center sampling strategy similar to [60,82], as we will show in our ablation study. Specifically, when determining the positive samples, only time steps within an interval around the center of an action are considered positive, where the duration of interval is proportional to the feature stride of the current pyramid level ℓ. More precisely, given an action centered at c, any time step $t \in [c - \alpha T/T^\ell, c + \alpha T/T^\ell]$ at the pyramid level ℓ is considered as positive, where $\alpha = 1.5$. Center sampling does not impact model inference, yet encourages higher scores around action centers.

3.4 Implementation Details

Training. Following [25], we use Adam [31] with warm-up for training. The warm-up stage is critical for model convergence and good performance, as also pointed out by [42]. When training with variable length input, we fix the maximum input sequence length, pad or cropped the input sequences accordingly, and add proper masking for operations in the model. This is equal to training with sliding windows as in [77]. Varying the maximum input sequence length during training has little impact to the performance, as shown in our ablation.

Inference. At inference time, we feed the full sequences into the model, as no position embeddings are used in the model. Our model takes the input video $\mathbf{X}$, and outputs $\{(p(a_t), d_t^s, d_t^e))\}$ for every time step t across all pyramid levels. Each time step t further decodes an action instance $(e_t = t - d_t^s, s_t = t + d_t^e, p(a_t))$. e_t and s_t are the onset and offset of the action, and $p(a_t)$ is an action confidence score. The result action candidates are further processed using Soft-NMS [6] to remove highly overlapping instances, leading to the final outputs of actions.

Network Architecture. We used 2 convolutions for projection, 7 Transformer blocks for the encoder (all using local attention and with 2x downsampling for the last 5), and separate classification and regression heads as the decoder. The regression range on each pyramid level was normalized by the stride of the features. More details are presented in the appendix C.

4   Experiments and Results

We now present our experiments and results. Our main results include benchmarks on THUMOS14 [29], ActivityNet-1.3 [10] and EPIC-Kitchens 100 [18]. Moreover, we provide extensive ablation studies of our model.

Evaluation Metric. For all datasets, we report the standard mean average precision (mAP) at different temporal intersection over union (tIoU) thresholds, widely used to evaluate TAL methods. tIoU is defined as the intersection over union between two temporal windows, *i.e.*, the 1D Jaccard index. Given a tIoU threshold, mAP computes the mean of average prevision across all action categories. An average mAP is also reported by averaging across several tIoUs.

Baseline and Comparison. For our main results on THUMOS14 [29] and ActivityNet-1.3 [10]. We compare to a strong set of baselines, including both two-stage (*e.g.*, G-TAD [75], BC-GNN [4], TAL-MR [84]) and single-stage (*e.g.*, A2Net [77], GTAN [50], AFSD [35], TadTR [46]) methods for TAL. Our close competitors are those single-stage methods. Despite our best attempt for a fair comparison, we recognize some of our baselines used different setups (*e.g.*, video features). Our experiment setup follows previous works [41,84]. And our intention here is to compare our results to the best results previously reported.

4.1   Results on THUMOS14

Dataset. THUMOS14 [29] dataset contains 413 untrimmed videos with 20 categories of actions. The dataset is divided into two subsets: validation set and test set. The validation set contains 200 videos and the test set contains 213 videos. Following the common practice [75,84,38,36], we use the validation set for training and report results on the test set.

Experiment Setup. We used two-stream I3D [12] pretrained on Kinetics to extract the video features on THUMOS14, following [41,84]. mAP@[0.3:0.1:0.7] was used to evaluate our model. A window size of 19 was used for local self-attention based on our ablation. Further details are described in the appendix C.

Table 1. Results on THUMOS14 and ActivityNet1.3. We report mAP at different tIoU thresholds. Average mAP in [0.3:0.1:0.7] is reported on THUMOS14 and [0.5:0.05:0.95] on ActivityNet1.3. Best results are in **bold** and second best <u>underlined</u>. Our method outperforms previous methods on THUMOS14 by a large margin, and beats previous methods when using the same features on ActivityNet1.3.

Type	Model	Feature	THUMOS14						ActivityNet1.3			
			0.3	0.4	0.5	0.6	0.7	Avg.	0.5	0.75	0.95	Avg.
Two-Stage	BMN [36]	TSN [65]	56.0	47.4	38.8	29.7	20.5	38.5	50.1	34.8	8.3	33.9
	DBG [34]	TSN [65]	57.8	49.4	39.8	30.2	21.7	39.8	—	—	—	—
	G-TAD [75]	TSN [65]	54.5	47.6	40.3	30.8	23.4	39.3	50.4	34.6	9.0	34.1
	BC-GNN [4]	TSN [65]	57.1	49.1	40.4	31.2	23.1	40.2	50.6	34.8	**9.4**	34.3
	TAL-MR [84]	I3D [12]	53.9	50.7	45.4	38.0	28.5	43.3	43.5	33.9	<u>9.2</u>	30.2
	P-GCN [80]	I3D [12]	63.6	57.8	49.1	—	—	—	48.3	33.2	3.3	31.1
	P-GCN [80]+TSP [2]	R(2+1)D [63]	69.1	63.3	53.5	40.4	26.0	50.5	—	—	—	—
	TSA-Net [26]	P3D [53]	61.2	55.9	46.9	36.1	25.2	45.1	48.7	32.0	9.0	31.9
	MUSES [45]	I3D [12]	68.9	64.0	56.9	46.3	31.0	—	50.0	35.0	6.6	34.0
	TCANet [52]	TSN [65]	60.6	53.2	44.6	36.8	26.7	44.3	52.3	36.7	6.9	35.5
	TCANet [52]	SlowFast [24]	—	—	—	—	—	—	54.3	**39.1**	8.4	**37.6**
	BMN-CSA [58]	TSN [65]	64.4	58.0	49.2	38.2	27.8	47.7	52.4	36.2	5.2	35.4
	ContextLoc [89]	I3D [12]	68.3	63.8	54.3	41.8	26.2	50.9	**56.0**	35.2	3.6	34.2
	VSGN [83]	TSN [65]	66.7	60.4	52.4	41.0	30.4	50.2	52.4	36.0	8.4	35.1
	VSGN [83]	I3D [12]	—	—	—	—	—	—	52.3	35.2	8.3	34.7
	VSGN [83]+TSP [2]	R(2+1)D [63]	—	—	—	—	—	—	53.3	36.8	8.1	35.9
	RTD-Net [59]	I3D [12]	68.3	62.3	51.9	38.8	23.7	49.0	47.2	30.7	8.6	30.8
Single-Stage	A^{2}Net [77]	I3D [12]	58.6	54.1	45.5	32.5	17.2	41.6	43.6	28.7	3.7	27.8
	GTAN [50]	P3D [53]	57.8	47.2	38.8	—	—	—	52.6	34.1	8.9	34.3
	PBRNet [43]	I3D [12]	58.5	54.6	51.3	41.8	29.5	—	54.0	35.0	9.0	35.0
	AFSD [35]	I3D [12]	67.3	62.4	55.5	43.7	31.1	52.0	52.4	35.3	6.5	34.4
	TadTR [46]	I3D [12]	62.4	57.4	49.2	37.8	26.3	46.6	49.1	32.6	8.5	32.3
	Ours	I3D [12]	**82.1**	**77.8**	**71.0**	**59.4**	**43.9**	**66.8**	53.5	36.2	8.2	35.6
	Ours+TSP [2]	R(2+1)D [63]	<u>73.4</u>	<u>67.4</u>	<u>59.1</u>	<u>46.7</u>	<u>31.5</u>	<u>55.6</u>	<u>54.7</u>	<u>37.8</u>	8.4	<u>36.6</u>

To show that our method can adapt to different video features, we also consider the pre-training method from [2] using an R(2+1)D network [63].

Results. Table 1 (left) summarizes the results. Our method achieves an average mAP of 66.8% ([0.3 : 0.1 : 0.7]), with an mAP of 71.0% at tIoU=0.5 and an mAP of 43.9% at tIoU=0.7, outperforming all previous methods by a large margin (+14.1% mAP at tIoU=0.5 and +12.8% mAP at tIoU=0.7). Our results stay on top of all single-stage methods, and also beats all previous two-stage methods, including the latest ones from [84,52,33,58]. Note that our method significantly outperforms the concurrent work of TadTR [46], which also designed a Transformer model for TAL. With the combination of a simple design and a strong Transformer model, our method establishes new state of the art on THUMOS14, crossing the 65% average mAP for the first time.

4.2 Results on ActivityNet-1.3

Dataset. ActivityNet-1.3 [10] is a large-scale action dataset which contains 200 activity classes and around 20,000 videos with more than 600 hours. The dataset is divided into three subsets: 10,024 videos for training, 4,926 for validation, and 5,044 for testing. Following the common practice in [38,36,75], we train our model on the training set and report the performance on the validation set.

Table 2. Results on EPIC-Kitchens 100 validation set. We report mAP at different tIoU thresholds and the average mAP in [0.1:0.1:0.5]. All methods used the same SlowFast features. Our method outperforms all baselines by a large margin.

Task	Method	0.1	0.2	0.3	0.4	0.5	Avg
	BMN [36,18]	10.8	9.8	8.4	7.1	5.6	8.4
Verb	G-TAD [75]	12.1	11.0	9.4	8.1	6.5	9.4
	Ours	**26.6**	**25.4**	**24.2**	**22.3**	**19.1**	**23.5**
	BMN [36,18]	10.3	8.3	6.2	4.5	3.4	6.5
Noun	G-TAD [75]	11.0	10.0	8.6	7.0	5.4	8.4
	Ours	**25.2**	**24.1**	**22.7**	**20.5**	**17.0**	**21.9**

Experiment Setup. We used two-stream I3D [12] for feature extraction.Following [38,36,75], the extracted features were downsampled into a fixed length of 160 using linear interpolation. For evaluation, we used mAP@[0.5:0.05:0.95] and also reported the average mAP. A window size of 11 was used for local self-attention. Further implementation details can be found in the appendix C. Moreover, we combined external classification results from [86] following [84,75,4,80]. Similarly, we consider the pre-training method from [2].

Results. Table 1 (right) shows the results. With I3D features, our method reaches an average mAP of 35.6% ([0.5 : 0.05 : 0.95]), outperforming all previous methods using the same features by at least 0.6%. This boost is significant as the result is averaged across many tIoU thresholds, including those tight ones *e.g.*, 0.95. Using the pre-training method from TSP [2] largely improves our results (36.6% average mAP). Our model thus outperforms the best method with the same features [83] by a major margin (+0.7%). Again, our method outperforms TadTR [46]. Our results are worse than TCANet [52]—a latest two-stage method using stronger SlowFast features [24] that are not publicly available. We conjecture our method will also benefit from better features. Nonetheless, our model clearly demonstrates state-of-the-art results on this challenging dataset.

4.3   Results on EPIC-Kitchens 100

Dataset. EPIC-Kitchens 100 is the largest egocentric action dataset. The dataset contains 100 hours of videos from 700 sessions capturing cooking activities in different kitchens. In comparison to ActivityNet-1.3, EPIC-Kitchens 100 has less number of videos, yet many more instances per video (average 128 vs. 1.5 on ActivityNet-1.3). In comparison to THUMOS14, EPIC-Kitchens is 3 times larger in terms of video hours and more than 10 times larger in terms of action instances. These egocentric videos also include significant camera motion. This dataset thus poses new challenges for TAL.

Experiment Setup. We used a SlowFast network [24] pre-trained on EPIC-Kitchens for feature extraction. This model is provided by [18]. Our model was trained on the training set and evaluated on the validation set. A window size of 9 was used for local self-attention. For evaluation, we used mAP@[0.1:0.1:0.5] and report the average mAP following [18]. In this dataset, an action is defined as a combination of a verb (action) and a noun (object). As this dataset was

recently released, we are only able to compare our methods to BMN [36] and G-TAD [75], both using the same SlowFast features provided by [18]. Again, implementation details are described in the appendix C.

Results. Table 2 presents the results. Our method achieves an average mAP ([0.1:0.1:0.5]) of 23.5% and 21.9% for verb and noun, respectively. Our results again largely outperform the strong baselines of BMN [36] and G-TAD [75] by over 13.5% in absolute percentage points. An interesting observation is that the gaps between our results and BMN / G-TAD are much larger on EPIC-Kitchens 100. A possible reason is that ActivityNet has a small number of actions per video (1.5), leading to imbalanced classification for our model; only a few moments (around the action center) are labeled positive while all rest are negative.

We adapt ActionFormer for EPIC-Kitchens 100 2022 Action Detection challenge. By combining features from SlowFast [24] and ViViT [3], ActionFormer achieves 21.36% / 20.95% average mAP for actions on the validation / test set. Our results ranked 2nd with a gap of 0.32 average mAP to the top solution.

4.4   Ablation Experiments

We conduct extensive ablations on THUMOS14 to understand our model design. Results are reported using I3D features with a fixed random seed for training. Further ablations on loss weight, maximum input length during training, input temporal feature resolution and error analysis can be found in the appendix A.

Baseline: A Convolutional Network. Our ablation starts by re-implementing a baseline anchor-free method (AF Base) as described in [77,35] (Table 3a row 1-2). This baseline shares the same action representation as our model, yet uses a 1D convolutional network as the encoder. We roughly match the number of layers and parameters of this baseline to our model. See the appendix C for more details. This baseline achieves an average mAP of 46.6% on THUMOS14 (Table 3a row 3), outperforms the numbers reported in [35] by 6.2%. We attribute the difference to variations in architectures and training schemes. This baseline, when using score fusion, reaches 52.9% average mAP (Table 3a row 4).

Transformer Network. Our next step is to simply replace the 1D convolutional network with our Transformer model using vanilla self-attention. This model achieves an average mAP of 62.7% (Table 3a row 5) — a major boost of 16.1%. We note that this model already outperforms the best reported results (56.9% mAP at tIoU=0.5 from [45]). This result shows that our Transformer model is very powerful for TAL, and serves as the main course of performance gain.

Layer Norm, Center Sampling, Position Encoding, & Score Fusion. We further add layer norm in the classification and regression heads, apply center sampling during training, and explore position encoding as well as score fusion (Table 3a row 6-9). Adding layer norm boosts the average mAP by 2.7%, and using center sampling further improves the performance by 1.4%. The commonly used position encoding, however, does not bring performance gain. We postulate that our projection using convolutions as well as the depthwise convolutions in our Transformer blocks already leak the location information, as also pointed

Table 3. Ablation studies on THUMOS14. We report mAP at tIoU=0.5 and 0.7, and the average mAP in $[0.3:0.1:0.7]$. Results are without score fusion unless specified.

Method	Backbone	LN	CTR	PE	Fusion	0.5	0.7	Avg
AF Base [77]	Conv					36.6	15.0	34.2
AF Base [35]	Conv					31.0	19.0	40.4
AF Base (Our Impl)	Conv					48.0	29.4	46.6
AF Base (Our Impl)	Conv				✓	54.6	33.4	52.9
Ours	Trans					66.8	38.4	62.7
Ours	Trans	✓				69.0	43.0	65.4
Ours	Trans	✓	✓	✓		70.4	43.6	66.7
Ours	Trans	✓	✓		✓	66.0	41.7	62.1
Ours	Trans	✓	✓			**71.0**	**43.9**	**66.8**
Ours (win size=19)	Trans	✓	✓			**71.0**	**43.9**	**66.8**

(a) **Model Design**: We start from a baseline 1D convolutional network (AF Base), and gradually replace convolutions with our Transformer model, add layer norm to heads (LN), enable center sampling during training (CTR), explore position encoding (PE), and fuse classification scores (Fusion).

Method	Win Size	0.5	0.7	Avg	GMACs	Time
AF Base	N/A	48.0	29.4	46.6	45.6	1.0x
Ours	9	70.5	42.7	66.5	45.2	2.0x
Ours	19	**71.0**	**43.9**	**66.8**	45.3	2.0x
Ours	25	70.3	43.9	66.4	45.4	2.0x
Ours	37	**71.0**	43.1	66.7	45.5	2.0x
Ours	Full	**71.0**	**43.9**	**66.8**	57.8	2.2x

(b) **Local Window Size**: We additionally report MACs and normalized run time by varying the local window size for self-attention in our model, using an input of 2304 time steps (5 minutes on THUMOS14). Time is normalized by setting AF Base to 1.0x.

Method	# Levels	Init Range	0.5	0.7	Avg
Ours	1	$[0, +\infty)$	51.8	15.8	47.6
Ours	3	$[0, 4)$	64.4	31.5	60.1
Ours	3	$[0, 8)$	61.4	30.0	57.6
Ours	3	$[0, 16)$	54.2	19.2	50.2
Ours	4	$[0, 4)$	67.4	39.7	63.7
Ours	5	$[0, 4)$	70.2	42.2	65.5
Ours	6	$[0, 4)$	**71.0**	**43.9**	**66.8**
Ours	7	$[0, 4)$	70.6	43.2	66.2

(c) **Design of Feature Pyramid**: We vary (1) the number of pyramid levels and (2) the initial regression range, and report mAP and the average mAP.

out in [72]. Further fusing the classification scores will decrease the largely performance. As a reference, when replacing the vanilla self-attention with a local version (window size=19), the average mAP remains the same.

Window Size for Local Self-Attention. Next, we study the effects of window size for local self-attention in our model. We vary the window size, re-train the model, and present both model accuracy, complexity (in GMACs), and run time in Table 3b. All results are reported without score fusion. Due to our design of a multiscale feature pyramid, even using the global self-attention only leads to 26% increase in MACs when compared to the baseline convolutional network. Reducing the window size cuts down the MACs yet maintains a similar accuracy. In addition to MACs, we also evaluate the normalized run time of these models on GPUs, where the base convolutional model is set to 1.0x. In spite of similar MACs, Transformer-based models are roughly 2x slower in run time compared to a convolution-based model (AF Base). It is known that self-attention is not easily parallelizable on GPUs. Also, our current implementation of Transformer used PyTorch primitives [51] without leveraging customized CUDA kernels.

Feature Pyramid. Further, we study the design of the feature pyramid. As discussed in Sec. 3.3, our design space is specified by (1) the number of pyramid
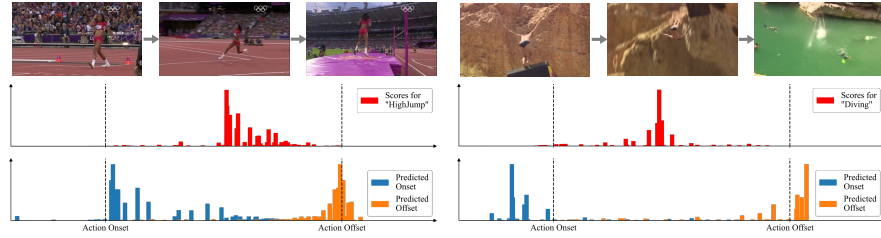
Fig. 3. Visualization of our results. From *top* to *bottom*: (1) input video frames; (2) action scores at each time step; (3) histogram of action onsets and offsets computed by weighting the regression outputs using action scores. See more in the appendix D

levels and (2) an initial regression range for the first pyramid level. We vary these parameters and report results in Table 3c. We follow our best design with a local window size=19 and layer norm, and center sampling.

First, we disable the feature pyramid and attach the heads to the feature map with the highest resolution. This is done by setting the number of pyramid to 1 with an initial regression range of $[0, +\infty)$, Removing the feature pyramid results in a major performance drop (-19.3% in average mAP), suggesting that using feature pyramid is critical for our model. Next, we set the number of pyramid levels to 3 and experiment with different initial regression ranges. The best results are achieved with the range of $[0, 4)$. Further increase of the range decreases the mAP scores. Finally, we fix the initial regression range to $[0, 4)$ and increase the number of pyramid levels. The performance of our method generally increases with more pyramid levels, yet is saturated when using 6 levels.

Result Visualization. Finally, we visualize the outputs of our model (before Soft-NMS) in Fig. 3, including the action scores, and the regression outputs weighted by the action scores (as a weighted histogram). Our model outputs a strong peak near the center of an action, potentially due to the employment of center sampling during training. The regression of action boundaries seems less accurate. We conjecture that our regression heads can be further improved.

5   Conclusion and Discussion

In this paper, we presented ActionFormer—a Transformer-based method for temporal action localization. ActionFormer has a simple design, falls into the category of single-stage anchor-free method, yet achieves impressive results across several major TAL benchmarks including THUMOS14, ActivityNet-1.3, and the more recent EPIC-Kitchens 100 (egocentric videos). Through our experiments, we showed that the power of ActionFormer lies in our orchestrated design, in particular the combination of local self-attention and a multiscale feature representation to model longer range temporal context in videos. We hope that our model, notwithstanding its simplicity, can shed light on the task of temporal action localization, as well as the broader field of video understanding.

6   Appendix

In the appendix, we describe (1) additional ablation experiments (Sec. A); (2) further error analysis of our results (Sec. B); (3) implementation details and how to reproduce our results (Sec. C); (4) additional visualizations of our results (Sec. D); and (5) limitation of our approach and furture directions (Sec. E). For sections, figures, tables, and equations, we use numbers (*e.g.*,, Sec. 1) to refer to the main paper and capital letters (*e.g.*,, Sec. A) to refer to this appendix.

A   Additional Ablation Experiments 额外的消融实验

Here we present additional ablation experiments, as mentioned in Sec. 4.4 of the main paper. These are omitted from the main paper due to lack of space. All experiments are reported on THUMOS14, consistent with our ablation experiments in the main paper. We follow our best design and use a local window size=19 with layer norm, center sampling, and score fusion enabled.

Loss Weight. We provide additional ablation on the loss weight λ_{reg} in Eq. 7. Specifically, we varied the loss weight $\lambda_{reg} \in [0.2, 0.5, 1, 2, 5]$, retrained the model, and reported the mAP scores. The results are presented in Table A. For a large range of λ_{reg}, our model has quite stable results with a maximum gap of 1.4% in average mAP. $\lambda_{reg} = 1$ yields the best results, as we used in all our experiments.

Table A. Ablation study on loss weight. We report mAP at tIoU=0.5 and 0.7, and the average mAP in $[0.3 : 0.1 : 0.7]$ on THUMOS14 by varying the loss weight λ_{reg} in Eq. 7.

Method	λ_{reg}	0.5	0.7	Avg
Ours	0.2	69.7	40.9	65.6
Ours	0.5	**71.3**	42.4	66.7
Ours	1	71.0	**43.9**	**66.8**
Ours	2	69.5	**43.9**	66.2
Ours	5	69.1	43.0	65.4

最大输入序列长度

Maximum Input Sequence Length during Training. A possible explanation of our superior results is that our model might benefit from training using a long sequence (2304 time steps as in our previous experiments). Here we examine the effects of maximum input sequence length during training. Table B reports mAP scores for different training sequence lengths. The results of our model remain fairly consistent even with much shorter input sequence length. Note that when truncating an input sequence, our training scheme is equal to training with sliding windows as in [77]. The differences are (1) the windows are dynamically sampled rather than pre-generated; (2) windows without foreground actions are

removed. When using a input sequence length of 512, similar to what was considered in [77] (512), our method only has a minor drop in average mAP (-1.1%) and significantly outperforms [77].

Table B. Ablation study on maximum input sequence length during training. We report mAP at tIoU=0.5 and 0.7, and the average mAP in $[0.3 : 0.1 : 0.7]$ on THUMOS14 by varying the maximum input length T_{max} for training.

Method	T_{max}	0.5	0.7	Avg
Ours	576	69.6	42.5	65.7
Ours	1152	**71.0**	42.7	66.3
Ours	2304	**71.0**	**43.9**	**66.8**

时序特征分辨率

Temporal Feature Resolution. Some of the previous works considered video features with lower temporal resolution. For example, a feature stride of 8 was used by PGCN [80] and ContextLoc [89]. To understand the effects of temporal feature resolution, we downsample our input I3D features and study the performance variation when using different feature strides. Table C report the results. When using a lower resolution (stride=8), the results of our model only drop slightly (-0.5% in average mAP). Further reducing the resolution (*e.g.*,, stride=16) leads to larger performance degradation, yet our results remains favourable.

Table C. Ablation study on temporal feature resolution. We report mAP at tIoU=0.5 and 0.7, and the average mAP in $[0.3 : 0.1 : 0.7]$ on THUMOS14 by varying the feature stride.

| Method	stride	0.5	0.7	Avg
Ours	4	**71.0**	**43.9**	**66.8**
Ours	8	69.8	**43.9**	66.3
Ours	16	65.8	38.4	61.9

B   Further Error Analyses 进一步的误差分析

We present further analyses of our results on THUMOS14 using the tool provided by [1]. We refer the readers to [1] for more details.

将动作以覆盖率，绝对长度和实例个数分类

Metrics. In [1], several characteristic metrics were defined given a dataset (*e.g.*, THUMOS14), including coverage, length, and the number of instances. Specifically, coverage presents the relative length of the actions (compared to the whole video), categorized into five bins: Extra Small (XS: (0, 0.02]), Small (S: (0.02, 0.04]), Medium (M: (0.04, 0.06]), Large (L:(0.06, 0.08]), and Extra Large (XL: (0.08, 1.0]). Length denotes the absolute length (in seconds) of actions, organized
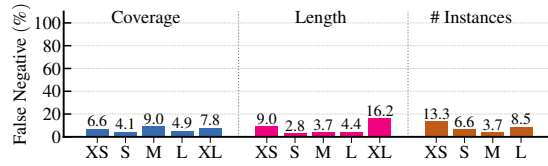
Fig. A. False negative (FN) profiling of our results on THUMOS14 using [1]. This figure shows the FN rates under different video contents. From this figure we can find that our model will suffer from extra short or extra long instances. Also, our model will suffer from video inputs which have a large number of action instances.
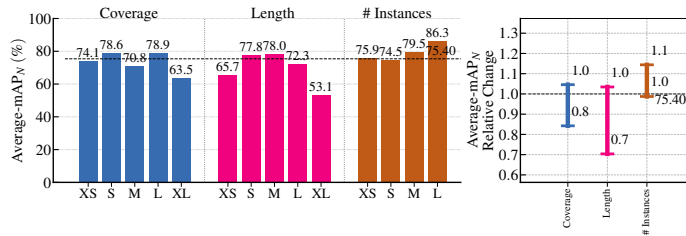

Fig. B. Sensitive analysis of our results on THUMOS14 using [1]. *Left*: normalized mAP at tIoU=0.5 under different video contents. *Right*: The relative normalized mAP change at tIoU=0.5 with respect to different characteristics of the ground truth instances.

into five length groups: Extra Small (XS: (0, 3]), Small (S: (3, 6]), Medium (M: (6, 12]), Long (L: (12, 18]), and Extra Long (XL: > 18). Moreover, number of instances refers to the total count of instances (from the same class) in a video. This number is further divided into four parts, including Extra Small (XS: 1); Small (S: [2, 40]); Medium (M: [40, 80]); Large (L: > 80).

Results and Analyses. Fig. A presents the false negative profiling. In Fig. A, we breakdown the false negative rates under the different coverage, length, and the number of instances. Our results have similar false negative rates across different coverage categories, yet have much higher false negative rates on action instances that are either very shot or very long (length), and on videos that contains many action instances (#instances). These action instances and videos are naturally more challenging.

Fig. B presents the sensitivity analysis of our results, *i.e.,*, normalized mAP at tIoU=0.5 under different characteristic metrics (left) and the variance of mAP across categories (right). Our model performs better on simple context scenarios, including XS/S/M/L coverage, S/M length and XS #instances, and worse on more complicated scenarios. The trend is similar to the false negative profiling in Fig. A. Moreover, our model is robust across different categories in coverage, length and #instances with small variances.
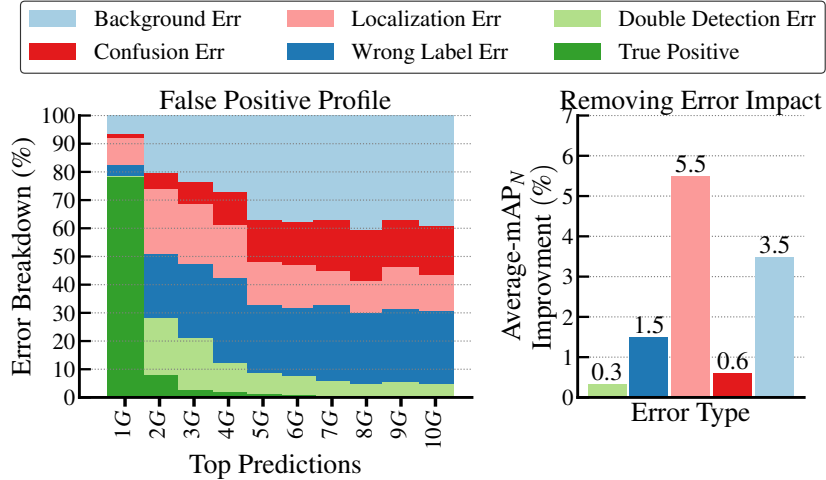
Fig. C. False positive (FP) profiling of our results on THUMOS14 using [1]. *Left*: FP error breakdown when considering the predictions for the top-10 ground-truth (G) instances. *Right*: The impact of error types. Localization error and background confusion are the top two error types.

C   Implementation Details实施细节

We now present implementation details including the network architecture, training and inference. Further details can be found in our code.

Network Architecture. We present our network architecture in Table D, as described in Sec. 3. In the ablation study (Sec. 4), we also considered a baseline that replaces the Transformer Units in Table D with convolution blocks, following the design of a bottleneck block in ResNet [27]. Specifically, a stack of three 1D convolutional layers were used. The kernel size of three convolutional layers were 1, 3 and 1, respectively. The expansion factor of the bottleneck block was 2. We added an extra strided convolutional layer with kernel size=1 and stride=2 to perform downsampling when necessary.

Training Details. For training, we considered both fixed length inputs (ActivityNet) and variable length inputs (THUMOS14, ActivityNet, and EPIC-Kitchens 100). For variable length inputs, we capped the input length to 2304 (around 5 minutes on THUMOS14 and around 20 minutes on EPIC-Kitchens 100), and randomly selected a subset of consecutive clips from an input video. Position embedding was disabled by default except for ActivityNet. Model EMA [28] and gradient clipping were also implemented to further stabilize the training. Hyperparameters were slightly different across datasets and discussed later in our experiment details.

Inference Details. For fixed length inputs (ActivityNet-1.3), we fed the full sequence into our model. For variable length inputs (THUMOS14 and EPIC-

Table D. The architecture of our model. Our network consists of (1) a Transformer encoder (first row block) and (2) a lightweight convolutional decoder with the classification / regression heads (last row block). For each layer, we list the layer name, layer parameters, the input to the layer, and the output feature size. We also include its regression range (in seconds for THUMOS14 and EPIC-Kitchens 100 and in number of time steps for ActivityNet-1.3). For convolutional layers, k is the kernel size of 1D convolutions and s is the stride, and c_i, c_o is the input and output feature channel, respectively. For Transformer Unit, ds is the downsampling ratio. T is the temporal length of input sequence and D is the input feature dimension. For classification head, the output dimension is the number of action categories. For regression head, the output dimension is 2, *i.e.*,, distances to action onset and offset.

	Name	Layer	Input	Output Size (T × D)	Regression Range
	input clip	-	-	T×D	-
	projection1	conv k=3, s=1 (c_i = D, c_o = 512)	input clip	T × 512	-
	projection2	conv k=3, s=1 (c_i = 512, c_o = 512)	projection1	T × 512	-
	transformer0	Transformer Unit, ds=1	projection2	T × 512	-
	transformer1	Transformer Unit, ds=1	transformer0	T × 512	[0, 4)
	transformer2	Transformer Unit, ds=2	transformer1	T/2 ×512	[4, 8)
encoder	transformer3	Transformer Unit, ds=2	transformer2	T/4 ×512	[8, 16)
	transformer4	Transformer Unit, ds=2	transformer3	T/8 ×512	[16, 32)
	transformer5	Transformer Unit, ds=2	transformer4	T/16 ×512	[32, 64)
	transformer6	Transformer Unit, ds=2	transformer5	T/32 ×512	[64, +∞)
decoder (heads)	cls / reg nets	conv k=3, s=1 (c_i = 512, c_o = 512)	transformer1,...,transformer6	[T/32×512,..., T×512]	-
		conv k=3, s=1 (c_i = 512, c_o = 512)	transformer1,...,transformer6	[T/32×512,..., T×512]	-
		conv k=3, s=1 (c_i = 512, c_o = output)	transformer1,...,transformer6	[T/32×output,..., T×output]	-

Kitchens 100), we sent the full sequence into the model. When using position embeddings in our ablation study, we adopted the technique from [20]. Specifically, for input sequences shorter than the training sequence length (2304), we fed the full sequence into our model and clipped the position embedding using the actual length of the video. For input sequences longer than the training sequence length, we again fed the full sequence into our model, yet used linear interpolation to upsample the position embeddings.

Score Fusion. For our experiments on THUMOS14 and ActivityNet-1.3, we sometimes consider score fusion using external classification scores. Specifically, given an input video, the top-2 video-level classes given by external classification scores were assigned to all detected action instances in this video, where the action scores from our model were multiplied with the external classification scores. Each detected action instance from our model thus creates two action instances. We refer the readers to [80] (Appendix E) for a more detailed description of the score fusion strategy.

Experiment Details. Our experiment details vary across datasets, as each dataset includes videos of different resolution and frame rate, and considers different types of features. We now describe our experiment details for THUMOS14, ActivityNet-1.3, and EPIC-Kitchens 100.

- **THUMOS14**: We used two-stream I3D [12] pretrained on Kinetics to extract the video features on THUMOS14, following [41,84]. We fed 16 consecutive frames as the input to I3D, used a sliding window with stride 4 and extracted 1024-D features before the last fully connected layer. The

two-stream features were further concatenated (2048-D) as the input to our model. mAP@[0.3:0.1:0.7] was used to evaluate our model. Our model was trained for 50 epochs with a linear warmup of 5 epochs. The initial learning rate was 1e-4 and a cosine learning rate decay is used. The mini-batch size was 2, and a weight decay of 1e-4 was used.

- **ActivityNet-1.3**: We used two-stream I3D [12] and TSP [2] for feature extraction, and increased the stride of the sliding window to 16. Following [38,36,75], the extracted features were downsampled into a fixed length of 160 and 192 using linear interpolation for I3D and TSP features, respectively. For evaluation, we used mAP@[0.5:0.05:0.95] and also reported the average mAP. Our model was trained for 15 epochs with a linear warmup of 5 epochs. The learning rate was 1e-3, the mini-batch size was 16, and the weight decay was 1e-4. For ActivityNet, we find it is helpful to train our model to generate proposals by considering all actions from a single category, and then use external classification scores for the recognition. This strategy was also used in previous single-stage TAL methods [35].

- **EPIC-Kitchens 100**: We used a SlowFast network [24] pre-trained on EPIC-Kitchens for feature extraction. This model is provided by [18]. We fed 32 frame window with a stride of 16 to extract 2304-D features. Our model was trained on the training set and evaluated on the validation set. A window size of 9 was used for local self-attention. For evaluation, we used mAP@[0.1:0.1:0.5] and report the average mAP following [18]. Our model was trained for 30 epochs with learning rate 1e-4, mini-batch size 2, and weight decay of 1e-4.

Reproducibility of Our Results. All results reported in the paper were obtained with the same random seed using PyTorch 1.10, CUDA 10.2 and CUDNN 7.6.5 on an NVIDIA Titan Xp GPU, using deterministic GPU computing routines. On the same machine, our code will always produce the same results when using the same random seed. Across machines/GPUs and computing environments, we have observed minor variation of average mAP scores (up to 0.5% average mAP on THUMOS, less than 0.2% average mAP on ActivityNet, and under 0.8% average mAP on EPIC Kitchens), yet those minor variations do not erode the clear performance gains of our method. Our code is made publicly available.

D   Additional Visualizations

Further, we present more visualizations of our results in Fig. D, extending Fig. 3 of the main paper. Our model is able to detect the occurrence of actions and estimate their temporal boundaries for the most of the cases (see the first column of Fig. D). The major failure modes of our model, as demonstrated in the second column of Fig. D, include (1) incorrect classification of action centers, *i.e.*, background confusion (classification errors); (2) inaccurate regression of the action's onset and offset (localization errors). We plan to address these issues in our future work.
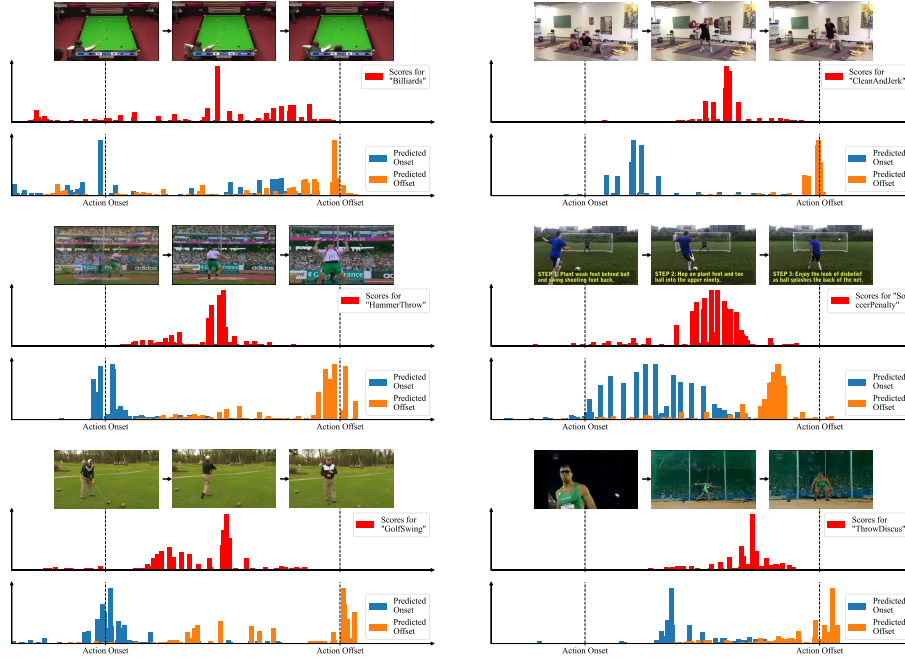
Fig. D. More visualization of our outputs. For each item from *top* to *bottom*: (1) the input video frames; (2) action scores at each time step; (3) histogram of action onsets and offsets computed by weighting the regression outputs using the action scores. *Left*: successful cases; *Right*: failure cases. This figure is best viewed in color and when zoomed in.

E   Limitations and Future Work

A main limitation of our method is the use of pre-extracted video features, also faced by many previous approaches. Another limitation is the need for many human labeled videos for training and the constraint of a pre-defined vocabulary of actions. Interesting future directions include pre-training for action localization [2,74], and learning from videos and text corpus [54,30] without human labels.

References

1. Alwassel, H., Caba Heilbron, F., Escorcia, V., Ghanem, B.: Diagnosing error in temporal action detectors. In: Eur. Conf. Comput. Vis. LNCS, vol. 11207, pp. 256–272 (2018)
2. Alwassel, H., Giancola, S., Ghanem, B.: TSP: Temporally-sensitive pretraining of video encoders for localization tasks. In: Int. Conf. Comput. Vis. Workshops. pp. 1–11 (2021)

3. Arnab, A., Dehghani, M., Heigold, G., Sun, C., Lučić, M., Schmid, C.: ViViT: A video vision transformer. In: Int. Conf. Comput. Vis. (2021)
4. Bai, Y., Wang, Y., Tong, Y., Yang, Y., Liu, Q., Liu, J.: Boundary content graph neural network for temporal action proposal generation. In: Eur. Conf. Comput. Vis. LNCS, vol. 12373, pp. 121–137 (2020)
5. Beltagy, I., Peters, M.E., Cohan, A.: Longformer: The long-document transformer. arXiv:2004.05150 (2020)
6. Bodla, N., Singh, B., Chellappa, R., Davis, L.S.: Soft-NMS–improving object detection with one line of code. In: Int. Conf. Comput. Vis. pp. 5561–5569 (2017)
7. Buch, S., Escorcia, V., Ghanem, B., Niebles Carlos, J.: End-to-end, single-stream temporal action detection in untrimmed videos. In: Brit. Mach. Vis. Conf. pp. 93.1–93.12 (2017)
8. Buch, S., Escorcia, V., Shen, C., Ghanem, B., Carlos Niebles, J.: SST: Single-stream temporal action proposals. In: IEEE Conf. Comput. Vis. Pattern Recog. pp. 2911–2920 (2017)
9. Caba Heilbron, F., Carlos Niebles, J., Ghanem, B.: Fast temporal activity proposals for efficient detection of human actions in untrimmed videos. In: IEEE Conf. Comput. Vis. Pattern Recog. pp. 1914–1923 (2016)
10. Caba Heilbron, F., Escorcia, V., Ghanem, B., Carlos Niebles, J.: ActivityNet: A large-scale video benchmark for human activity understanding. In: IEEE Conf. Comput. Vis. Pattern Recog. pp. 961–970 (2015)
11. Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., Zagoruyko, S.: End-to-end object detection with transformers. In: Eur. Conf. Comput. Vis. LNCS, vol. 12346, pp. 213–229 (2020)
12. Carreira, J., Zisserman, A.: Quo vadis, action recognition? a new model and the Kinetics dataset. In: IEEE Conf. Comput. Vis. Pattern Recog. pp. 4724–4733 (2017)
13. Chang, S., Wang, P., Wang, F., Li, H., Feng, J.: Augmented transformer with adaptive graph for temporal action proposal generation. arXiv preprint arXiv:2103.16024 (2021)
14. Chao, Y.W., Vijayanarasimhan, S., Seybold, B., Ross, D.A., Deng, J., Sukthankar, R.: Rethinking the Faster-RCNN architecture for temporal action localization. In: IEEE Conf. Comput. Vis. Pattern Recog. pp. 1130–1139 (2018)
15. Cheng, B., Schwing, A.G., Kirillov, A.: Per-pixel classification is not all you need for semantic segmentation. In: Adv. Neural Inform. Process. Syst. (2021)
16. Choromanski, K., Likhosherstov, V., Dohan, D., Song, X., Gane, A., Sarlos, T., Hawkins, P., Davis, J., Mohiuddin, A., Kaiser, L., et al.: Rethinking attention with performers. Int. Conf. Learn. Represent. (2021)
17. Dai, X., Chen, Y., Yang, J., Zhang, P., Yuan, L., Zhang, L.: Dynamic DETR: End-to-end object detection with dynamic attention. In: Int. Conf. Comput. Vis. pp. 2988–2997 (2021)
18. Damen, D., Doughty, H., Farinella, G.M., Furnari, A., Kazakos, E., Ma, J., Moltisanti, D., Munro, J., Perrett, T., Price, W., et al.: Rescaling egocentric vision. arXiv preprint arXiv:2006.13256 (2020)
19. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: BERT: Pre-training of deep bidirectional transformers for language understanding. In: North American Asso. Comput. Lin. pp. 4171–4186 (2019)
20. Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al.: An image is worth 16x16 words: Transformers for image recognition at scale. In: Int. Conf. Learn. Represent. (2021)

21. Duan, K., Bai, S., Xie, L., Qi, H., Huang, Q., Tian, Q.: CenterNet: Keypoint triplets for object detection. In: Int. Conf. Comput. Vis. pp. 6569–6578 (2019)
22. Escorcia, V., Heilbron, F.C., Niebles, J.C., Ghanem, B.: DAPs: Deep action proposals for action understanding. In: Eur. Conf. Comput. Vis. LNCS, vol. 9907, pp. 768–784 (2016)
23. Fan, H., Xiong, B., Mangalam, K., Li, Y., Yan, Z., Malik, J., Feichtenhofer, C.: Multiscale vision transformers. In: Int. Conf. Comput. Vis. (2021)
24. Feichtenhofer, C., Fan, H., Malik, J., He, K.: SlowFast networks for video recognition. In: Int. Conf. Comput. Vis. pp. 6202–6211 (2019)
25. Girdhar, R., Carreira, J., Doersch, C., Zisserman, A.: Video action transformer network. In: IEEE Conf. Comput. Vis. Pattern Recog. pp. 244–253 (2019)
26. Gong, G., Zheng, L., Mu, Y.: Scale matters: Temporal scale aggregation network for precise action localization in untrimmed videos. In: Int. Conf. Multimedia and Expo. pp. 1–6. IEEE (2020)
27. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: IEEE Conf. Comput. Vis. Pattern Recog. pp. 770–778 (2016)
28. Huang, G., Li, Y., Pleiss, G., Liu, Z., Hopcroft, J.E., Weinberger, K.Q.: Snapshot ensembles: Train 1, get m for free. In: Int. Conf. Learn. Represent. (2017)
29. Idrees, H., Zamir, A.R., Jiang, Y.G., Gorban, A., Laptev, I., Sukthankar, R., Shah, M.: The THUMOS challenge on action recognition for videos "in the wild". Comput. Vis. and Image Under. **155**, 1–23 (2017)
30. Jia, C., Yang, Y., Xia, Y., Chen, Y.T., Parekh, Z., Pham, H., Le, Q.V., Sung, Y., Li, Z., Duerig, T.: Scaling up visual and vision-language representation learning with noisy text supervision. In: Int. Conf. Mach. Learn. (2021)
31. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. In: Int. Conf. Learn. Represent. pp. 1–11 (2015)
32. Li, S., Jin, X., Xuan, Y., Zhou, X., Chen, W., Wang, Y.X., Yan, X.: Enhancing the locality and breaking the memory bottleneck of transformer on time series forecasting. In: Adv. Neural Inform. Process. Syst. vol. 32 (2019)
33. Li, X., Lin, T., Liu, X., Zuo, W., Li, C., Long, X., He, D., Li, F., Wen, S., Gan, C.: Deep concept-wise temporal convolutional networks for action localization. In: Proceedings of the 28th ACM International Conference on Multimedia. pp. 4004–4012 (2020)
34. Lin, C., Li, J., Wang, Y., Tai, Y., Luo, D., Cui, Z., Wang, C., Li, J., Huang, F., Ji, R.: Fast learning of temporal action proposal via dense boundary generator. In: AAAI. pp. 11499–11506 (2020)
35. Lin, C., Xu, C., Luo, D., Wang, Y., Tai, Y., Wang, C., Li, J., Huang, F., Fu, Y.: Learning salient boundary feature for anchor-free temporal action localization. In: IEEE Conf. Comput. Vis. Pattern Recog. pp. 3320–3329 (2021)
36. Lin, T., Liu, X., Li, X., Ding, E., Wen, S.: BMN: Boundary-matching network for temporal action proposal generation. In: Int. Conf. Comput. Vis. pp. 3889–3898 (2019)
37. Lin, T., Zhao, X., Shou, Z.: Single shot temporal action detection. In: ACM Int. Conf. Multimedia. pp. 988–996 (2017)
38. Lin, T., Zhao, X., Su, H., Wang, C., Yang, M.: BSN: Boundary sensitive network for temporal action proposal generation. In: Eur. Conf. Comput. Vis. LNCS, vol. 11208, pp. 3–19 (2018)
39. Lin, T.Y., Dollár, P., Girshick, R., He, K., Hariharan, B., Belongie, S.: Feature pyramid networks for object detection. In: IEEE Conf. Comput. Vis. Pattern Recog. pp. 2117–2125 (2017)

40. Lin, T.Y., Goyal, P., Girshick, R., He, K., Dollár, P.: Focal loss for dense object detection. In: Int. Conf. Comput. Vis. pp. 2980–2988 (2017)
41. Liu, D., Jiang, T., Wang, Y.: Completeness modeling and context separation for weakly supervised temporal action localization. In: IEEE Conf. Comput. Vis. Pattern Recog. pp. 1298–1307 (2019)
42. Liu, L., Liu, X., Gao, J., Chen, W., Han, J.: Understanding the difficulty of training transformers. In: Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP). pp. 5747–5763 (2020)
43. Liu, Q., Wang, Z.: Progressive boundary refinement network for temporal action detection. In: AAAI. vol. 34, pp. 11612–11619 (2020)
44. Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.Y., Berg, A.C.: SSD: Single shot multibox detector. In: Eur. Conf. Comput. Vis. pp. 21–37 (2016)
45. Liu, X., Hu, Y., Bai, S., Ding, F., Bai, X., Torr, P.H.: Multi-shot temporal event localization: a benchmark. In: IEEE Conf. Comput. Vis. Pattern Recog. pp. 12596–12606 (2021)
46. Liu, X., Wang, Q., Hu, Y., Tang, X., Bai, S., Bai, X.: End-to-end temporal action detection with transformer. arXiv preprint arXiv:2106.10271 (2021)
47. Liu, Y., Ma, L., Zhang, Y., Liu, W., Chang, S.F.: Multi-granularity generator for temporal action proposal. In: IEEE Conf. Comput. Vis. Pattern Recog. pp. 3604–3613 (2019)
48. Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., Lin, S., Guo, B.: Swin transformer: Hierarchical vision transformer using shifted windows. In: Int. Conf. Comput. Vis. (2021)
49. Liu, Z., Ning, J., Cao, Y., Wei, Y., Zhang, Z., Lin, S., Hu, H.: Video swin transformer. arXiv preprint arXiv:2106.13230 (2021)
50. Long, F., Yao, T., Qiu, Z., Tian, X., Luo, J., Mei, T.: Gaussian temporal awareness networks for action localization. In: IEEE Conf. Comput. Vis. Pattern Recog. pp. 344–353 (2019)
51. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al.: PyTorch: An imperative style, high-performance deep learning library. In: Adv. Neural Inform. Process. Syst. vol. 32 (2019)
52. Qing, Z., Su, H., Gan, W., Wang, D., Wu, W., Wang, X., Qiao, Y., Yan, J., Gao, C., Sang, N.: Temporal context aggregation network for temporal action proposal refinement. In: IEEE Conf. Comput. Vis. Pattern Recog. pp. 485–494 (2021)
53. Qiu, Z., Yao, T., Mei, T.: Learning spatio-temporal representation with pseudo-3d residual networks. In: Int. Conf. Comput. Vis. pp. 5533–5541 (2017)
54. Radford, A., Kim, J.W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., et al.: Learning transferable visual models from natural language supervision. In: Int. Conf. Mach. Learn. (2021)
55. Rezatofighi, H., Tsoi, N., Gwak, J., Sadeghian, A., Reid, I., Savarese, S.: Generalized intersection over union: A metric and a loss for bounding box regression. In: IEEE Conf. Comput. Vis. Pattern Recog. pp. 658–666 (2019)
56. Shou, Z., Chan, J., Zareian, A., Miyazawa, K., Chang, S.F.: CDC: Convolutional-de-convolutional networks for precise temporal action localization in untrimmed videos. In: IEEE Conf. Comput. Vis. Pattern Recog. pp. 5734–5743 (2017)
57. Shou, Z., Wang, D., Chang, S.F.: Temporal action localization in untrimmed videos via multi-stage CNNs. In: IEEE Conf. Comput. Vis. Pattern Recog. pp. 1049–1058 (2016)

58. Sridhar, D., Quader, N., Muralidharan, S., Li, Y., Dai, P., Lu, J.: Class semantics-based attention for action detection. In: Int. Conf. Comput. Vis. pp. 13739–13748 (2021)
59. Tan, J., Tang, J., Wang, L., Wu, G.: Relaxed transformer decoders for direct action proposal generation. In: Int. Conf. Comput. Vis. pp. 13526–13535 (2021)
60. Tian, Z., Shen, C., Chen, H., He, T.: FCOS: Fully convolutional one-stage object detection. In: Int. Conf. Comput. Vis. pp. 9627–9636 (2019)
61. Touvron, H., Cord, M., Douze, M., Massa, F., Sablayrolles, A., Jégou, H.: Training data-efficient image transformers & distillation through attention. In: Int. Conf. Mach. Learn. pp. 10347–10357 (2021)
62. Touvron, H., Cord, M., Sablayrolles, A., Synnaeve, G., Jégou, H.: Going deeper with image transformers. In: Int. Conf. Comput. Vis. (2021)
63. Tran, D., Wang, H., Torresani, L., Ray, J., LeCun, Y., Paluri, M.: A closer look at spatiotemporal convolutions for action recognition. In: IEEE Conf. Comput. Vis. Pattern Recog. pp. 6450–6459 (2018)
64. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. In: Adv. Neural Inform. Process. Syst. pp. 5998–6008 (2017)
65. Wang, L., Xiong, Y., Wang, Z., Qiao, Y., Lin, D., Tang, X., Van Gool, L.: Temporal segment networks: Towards good practices for deep action recognition. In: Eur. Conf. Comput. Vis. LNCS, vol. 11205, pp. 20–36 (2016)
66. Wang, L., Yang, H., Wu, W., Yao, H., Huang, H.: Temporal action proposal generation with transformers. arXiv preprint arXiv:2105.12043 (2021)
67. Wang, S., Li, B., Khabsa, M., Fang, H., Ma, H.: Linformer: Self-attention with linear complexity. arXiv preprint arXiv:2006.04768 (2020)
68. Wang, T., Yuan, L., Chen, Y., Feng, J., Yan, S.: PnP-DETR: Towards efficient visual analysis with transformers. In: Int. Conf. Comput. Vis. pp. 4661–4670 (2021)
69. Wang, W., Xie, E., Li, X., Fan, D.P., Song, K., Liang, D., Lu, T., Luo, P., Shao, L.: Pyramid vision transformer: A versatile backbone for dense prediction without convolutions. In: Int. Conf. Comput. Vis. (2021)
70. Wang, Y., Xu, Z., Wang, X., Shen, C., Cheng, B., Shen, H., Xia, H.: End-to-end video instance segmentation with Transformers. In: IEEE Conf. Comput. Vis. Pattern Recog. pp. 8741–8750 (2021)
71. Xiao, T., Singh, M., Mintun, E., Darrell, T., Dollár, P., Girshick, R.: Early convolutions help transformers see better. In: Adv. Neural Inform. Process. Syst. (2021)
72. Xie, E., Wang, W., Yu, Z., Anandkumar, A., Alvarez, J.M., Luo, P.: Segformer: Simple and efficient design for semantic segmentation with transformers. In: Adv. Neural Inform. Process. Syst. (2021)
73. Xiong, Y., Zeng, Z., Chakraborty, R., Tan, M., Fung, G., Li, Y., Singh, V.: Nyströmformer: A nyström-based algorithm for approximating self-attention. In: AAAI. vol. 35, pp. 14138–14148 (2021)
74. Xu, M., Pérez-Rúa, J.M., Escorcia, V., Martinez, B., Zhu, X., Zhang, L., Ghanem, B., Xiang, T.: Boundary-sensitive pre-training for temporal localization in videos. In: Int. Conf. Comput. Vis. pp. 7220–7230 (2021)
75. Xu, M., Zhao, C., Rojas, D.S., Thabet, A., Ghanem, B.: G-TAD: Sub-graph localization for temporal action detection. In: IEEE Conf. Comput. Vis. Pattern Recog. pp. 10156–10165 (2020)
76. Yang, J., Li, C., Zhang, P., Dai, X., Xiao, B., Yuan, L., Gao, J.: Focal self-attention for local-global interactions in vision transformers. In: Adv. Neural Inform. Process. Syst. (2021)

77. Yang, L., Peng, H., Zhang, D., Fu, J., Han, J.: Revisiting anchor mechanisms for temporal action localization. IEEE Trans. Image Process. **29**, 8535–8548 (2020)
78. Yang, Z., Qin, J., Huang, D.: Acgnet: Action complement graph network for weakly-supervised temporal action localization. In: AAAI. vol. 36-3, pp. 3090–3098 (2022)
79. Yuan, L., Chen, Y., Wang, T., Yu, W., Shi, Y., Jiang, Z., Tay, F.E., Feng, J., Yan, S.: Tokens-to-token ViT: Training vision transformers from scratch on ImageNet. In: Int. Conf. Comput. Vis. (2021)
80. Zeng, R., Huang, W., Tan, M., Rong, Y., Zhao, P., Huang, J., Gan, C.: Graph convolutional networks for temporal action localization. In: Int. Conf. Comput. Vis. pp. 7094–7103 (2019)
81. Zeng, R., Xu, H., Huang, W., Chen, P., Tan, M., Gan, C.: Dense regression network for video grounding. In: IEEE Conf. Comput. Vis. Pattern Recog. pp. 10287–10296 (2020)
82. Zhang, S., Chi, C., Yao, Y., Lei, Z., Li, S.Z.: Bridging the gap between anchor-based and anchor-free detection via adaptive training sample selection. In: IEEE Conf. Comput. Vis. Pattern Recog. pp. 9759–9768 (2020)
83. Zhao, C., Thabet, A.K., Ghanem, B.: Video self-stitching graph network for temporal action localization. In: Int. Conf. Comput. Vis. pp. 13658–13667 (2021)
84. Zhao, P., Xie, L., Ju, C., Zhang, Y., Wang, Y., Tian, Q.: Bottom-up temporal action localization with mutual regularization. In: Eur. Conf. Comput. Vis. LNCS, vol. 12353, pp. 539–555 (2020)
85. Zhao, Y., Xiong, Y., Wang, L., Wu, Z., Tang, X., Lin, D.: Temporal action detection with structured segment networks. In: Int. Conf. Comput. Vis. pp. 2914–2923 (2017)
86. Zhao, Y., Zhang, B.w., Wu, Z., Yang, S., Zhou, L., Yan, S., Wang, L., Xiong, Y., Lin, D., Qiao, Y., Tang, X.: CUHK & ETHZ & SIAT submission to ActivityNet challenge 2017. arXiv preprint arXiv:1710.08011 (2017)
87. Zheng, Z., Wang, P., Liu, W., Li, J., Ye, R., Ren, D.: Distance-IoU loss: Faster and better learning for bounding box regression. In: AAAI (2020)
88. Zhu, X., Su, W., Lu, L., Li, B., Wang, X., Dai, J.: Deformable DETR: Deformable transformers for end-to-end object detection. In: Int. Conf. Learn. Represent. pp. 1–11 (2021)
89. Zhu, Z., Tang, W., Wang, L., Zheng, N., Hua, G.: Enriching local and global contexts for temporal action localization. In: Int. Conf. Comput. Vis. pp. 13516–13525 (2021)