End-to-End Action Segmentation Transformer

Tieqiao Wang Sinisa Todorovic
Oregon State University
{wangtie, sinisa}@oregonstate.edu
https://github.com/tqosu/EAST

Abstract

Most recent work on action segmentation relies on pre-computed frame features from models trained on other tasks and typically focuses on framewise encoding and labeling without explicitly modeling action segments. To overcome these limitations, we introduce the End-to-End Action Segmentation Transformer (EAST), which processes raw video frames directly – eliminating the need for pre-extracted features and enabling true end-to-end training. Our contributions are as follows: (1) a lightweight adapter design for effective fine-tuning of large backbones; (2) an efficient segmentation-by-detection framework for leveraging action proposals predicted over a coarsely downsampled video; and (3) a novel action-proposal-based data augmentation strategy. EAST achieves SOTA performance on standard benchmarks, including GTEA, 50Salads, Breakfast, and Assembly-101.

1. Introduction

Action segmentation is a basic vision problem that involves labeling frames of an untrimmed video with their corresponding action classes. The problem poses many challenges, including the inherent ambiguity of action boundaries, and significant computational demands of processing long video sequences.

Recent approaches typically address these challenges by using pre-computed frame features, e.g., I3D [6] or TSM [23]. Such features are known to be suboptimal [8], because they have been previously extracted by other methods trained on vision tasks that are different from action segmentation (e.g., action recognition). Furthermore, due to memory and computational constraints, most approaches focus on framewise representations [9, 25, 39] that lack explicit modeling of action instances (with few exceptions [27, 37] that increase complexity). Consequently, they neglect the bottom-up/top-down integration of frame and action representations, which was once essential in traditional frameworks [1, 5, 15, 29, 31]. Finally, as datasets for action segmentation are significantly smaller than for other tasks (e.g., action recognition), previous work resorts to data augmentation and self-supervised learning [2, 19, 28]. However, these methods typically augment only local frame features, and do not augment action instances.

To address these limitations, we propose the End-to-End Action Segmentation Transformer (EAST). As shown in Fig. 1, EAST consists of the following modules: (a) a large backbone, (b) a detector that predicts action proposals over coarsely sampled frames; (c) an aggregator that combines the proposals to infer class distributions for all frames at the original (unsampled) frame rate; and (d) a refinement module for predicting the final framewise labels. With EAST, we make the following three key contributions.

First, we enable efficient end-to-end training of EAST by introducing lightweight Contract-Expand Adapters (CEA) into a large backbone network. CEA reduces complexity by compressing and expanding features around depth-wise convolutions. This allows efficient fine-tuning of the large backbone to extract multiscale features directly from raw RGB frames, ensuring they are optimized for action segmentation rather than for unrelated vision tasks. While recent methods such as Bridge-Prompt [20] and FACT [27] also claim end-to-end training from RGB inputs, their training strategy consists of disjoint stages – first training the backbone, then freezing features to train the rest of their segmentation model. In contrast, EAST is trained truly end-to-end, without freezing features at any stage.

Second, unlike most recent work that focuses on framewise labeling without explicitly modeling the temporal extent of action instances, EAST performs action segmentation by detection. Specifically, EAST detects action proposals over a downsampled input by regressing action boundaries over the sampled frames. The proposals are then integrated and refined for the final segmentation. This provides two key advantages: improved efficiency by detecting action proposals on coarsely sampled frames (e.g., 1–6 fps), and enhanced framewise classification by explicitly considering the context of detected action segments. It is worth noting that temporal downsampling does not affect

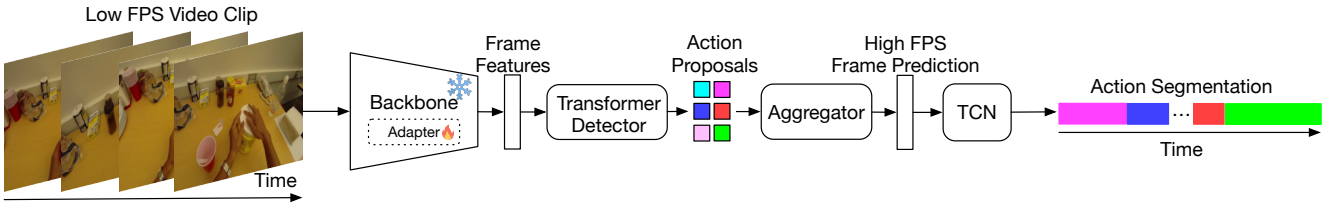

Figure 1. EAST consists of a frozen backbone with trainable adapters for efficient feature adaptation, a transformer-based detector for regressing action boundaries over coarsely sampled frames (low FPS), an aggregator for combining the action proposals to predict a class distribution of every frame at the original frame-rate (high FPS), and a refinement module to perform final framewise classification.

the ground truth or evaluation, as our boundary regression is specified relative to timestamps within the video. These boundary detections are mapped to the original full frame rate, and serve as useful constraints for final action segmentation over all frames.

Third, we introduce a novel proposal-based data augmentation method to enhance EAST training. In training, our experiments show that the high-confidence action proposals generally align well with the ground truth, creating an easier learning signal for the subsequent proposal aggregation and refinement stages. To encourage learning from more challenging cases, we augment training by selectively removing high-confidence proposals before passing the remaining proposals to the subsequent EAST modules.

With these contributions, EAST achieves state-of-the-art performance on standard benchmarks, including the GTEA, 50Salads, Breakfast, and Assembly-101 datasets. Our approach outperforms existing methods across all metrics.

2. Related Work

This section reviews closely related work.

Efficient Training. While end-to-end training offers known advantages, memory and computational constraints often make it impractical. Parameter-efficient fine-tuning (PEFT) methods, such as adapters [12], LoRA [14], and prefix-tuning [22], address these limitations by reducing the number of trainable parameters. However, PEFT's potential for video understanding, particularly action segmentation, remains largely unexplored. AdaTAD [26] introduces Temporal-Informative Adapters (TIA) for action detection, using depth-wise convolutions (DWConv) [13] to enhance temporal reasoning. While TIA improves performance over standard adapters [12], it also increases complexity and slows convergence. To address this, we propose the Contract-Expand Adapter (CEA), designed specifically for action segmentation. CEA applies feature compression and expansion around the DWConv. This reduces the computational load within the adapter, achieving both the performance gains of TIA and the benefits of standard adapters – lower complexity and faster convergence.

Temporal Action Segmentation has been tackled with multi-stage framewise networks like MS-TCN [9], ASFormer [39], and DiffAct [25]. However, these lack explicit action instance representations and require post-processing, hindering end-to-end training. More recent approaches (UVAST [4], FACT [27], BaFormer [37]) model actions using query tokens alongside frame tokens, but at the cost of significantly increased complexity.

Importantly, most recent methods operate on all frames, at the input frame rate, and do not use action boundaries to constrain framewise labeling [9, 39]. This limits their ability to handle downsampled videos, a critical requirement for efficient end-to-end training with long videos. In contrast, we perform efficient action-boundary regression on multi-scale frame features, followed by the integration of the action proposals "top-down" for final framewise classification. This enables competitive performance of EAST even with downsampled input, significantly reducing model and computational complexity, compared to methods requiring full, unsampled video sequences [9, 39].

Test-Time Post-Processing and Data Augmentation. To improve performance, some approaches resort to post-processing, such as, e.g., Viterbi decoding [4]. However, Viterbi decoding is computationally expensive and incompatible with end-to-end training. To enable robust training on relatively small action segmentation datasets, prior work uses data augmentation [2, 25], which are either simplistic – e.g., feature masking [25] – or overly complex – e.g., reinforcement learning based sequence generation [2]. The latter would be difficult to optimize within an end-to-end framework. In contrast, we introduce a new data augmentation method that manipulates action proposals to enforce EAST training under higher uncertainty conditions, seamlessly integrating within our end-to-end training. To our knowledge, this is the first work to apply proposal-based data augmentation for action segmentation.

3. Specification of EAST

EAST consists of a backbone, detector, integrator, and refinement module, as shown in Fig. 1. Given an untrimmed RGB video, $\mathbf{V} \in \mathbb{R}^{T \times H \times W \times 3}$, as input, the backbone takes a downsampled sequence, $\mathbf{V}' \in \mathbb{R}^{T' \times H \times W \times 3}$, where H and W are the frame height and width, and T' is the number of coarsely sampled frames, $T' \ll T$. Frames are uni-

formly sampled at an empirically optimized rate to facilitate efficient end-to-end training. The backbone output is passed to the detector to predict: (i) Initial frame labels, $\hat{\mathcal{Y}}_1 = \{(t_i, \hat{y}_i)\}_{i=1}^{T'}$, where t_i is the timestamp of frame i in the original (unsampled) video $\mathbf{V}$, and $\hat{y}_i$ is its predicted class; and (ii) Action proposals, $\hat{\mathcal{S}} = \{(\hat{t}_n^s, \hat{t}_n^e, \boldsymbol{\pi}_n)\}_{n=1}^{N}$, where N is the number of action proposals, $\hat{t}_n^s$ and $\hat{t}_n^e$ denote the predicted start and end timestamps in $\mathbf{V}$ of nth action proposal, and $\boldsymbol{\pi}_n$ is the predicted class distribution, $\boldsymbol{\pi}_n = \{\pi_n(a) : a \in \mathcal{A}\}$, over the set of action classes, $\mathcal{A}$, including a background class. For each sampled frame in $\mathbf{V}'$, the detector regresses timestamps of action boundaries in the unsampled $\mathbf{V}$, rather than their frame indices. This enables the use of variable downsampling frame rates based on available memory and computational resources.

EAST's integrator takes the action proposals in $\hat{\mathcal{S}}$ as input, and combines them to predict a class distribution of every frame of the unsampled $\mathbf{V}$. These predictions are then progressively refined through multiple stages of the standard Temporal Convolutional Network (TCN) [9] for final framewise classification, $\hat{\mathcal{Y}}_2$, over all T frames.

Our end-to-end training uses the proposed new data augmentation method, where $\hat{\mathcal{S}}$ is corrupted by randomly removing a subset of the most confident proposals, before producing $\hat{\mathcal{Y}}_2$. In the following, we provide a more detailed specification of EAST.

3.1. Contract-Expand Adapter

In training, EAST fine-tunes a pre-trained video foundation model on a given action segmentation dataset. As the backbone, we use ViT-G [40], pre-trained with VideoMAEv2 [36] on related vision tasks. To facilitate efficient end-to-end training within memory and computational constraints, we design a lightweight Contract-Expand Adapter (CEA), and integrate it into ViT-G. Building on the recent approaches to feature adaptation [12, 26], we insert CEA between the backbone's layers. As shown in Fig. 2, CAE adapts the features $\boldsymbol{x}$ of the previous layer with a residual, which results in the adapted features $\boldsymbol{x}'$ that are further passed to the following layer.

Fig. 2 illustrates key differences of CEA from previous approaches. The Standard Adapter [12] consists of down-projection and up-projection layers with a non-linear activation. However, the Standard Adapter does not explicitly model temporal context, making it unsuitable for action segmentation. The Temporal Interaction Adapter (TIA) [26] incorporates temporal depth-wise convolutional layers (DWConv) to aggregate temporal context. TIA first reshapes a given input feature of shape (B, C, T, H, W) to $(B \times H \times W, C, T)$, and then applies the same DWConv independently to each spatial location $(h, w) \in H \times W$. This results in a high computational cost, which would make TIA very challenging to incorporate in our end-to-end training. To meet our memory and computational constraints, we adopt a simpler adapter design as follows.

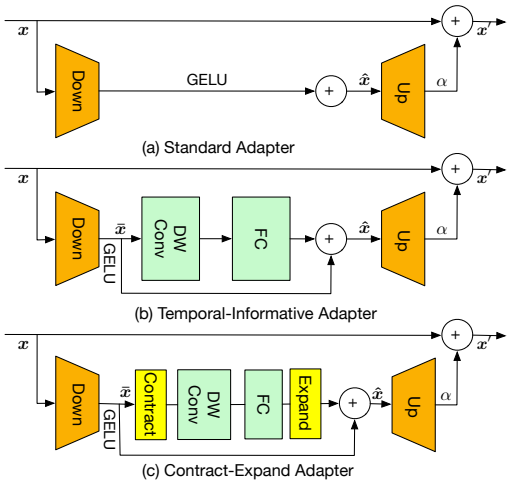

Figure 2. Adapters for efficient fine-tuning of a foundation backbone are typically inserted between its layers. (a) Standard Adapter [12] (orange). (b) Temporal Interaction Adapter (TIA) [26] (green). (c) Our Contract-Expand Adapter (CEA) (yellow). CEA consists of temporal depth-wise convolutions [13], and parameter-free contract/expand layers.

Our key idea is to use spatial average pooling directly within the adapter to reduce the number of spatial locations $|H \times W|$ that share the same DWConv. This is based on the hypothesis that, during backbone fine-tuning, spatial context is less crucial for feature adaptation than temporal context. Our spatial average pooling significantly reduces data flow and computational complexity compared to TIA. After applying DWConv to a few pooled spatial locations (h, w), the resulting features are then appropriately copied to the other $H \times W$ locations, spatially upscaling the enhanced features before they are passed to the next backbone layer.

As our results show, not only does our contract-and-expand strategy reduce GFLOPs, but it also improves both convergence speed and overall performance. Placing the spatial pooling outside the down- and up-projection layers of the adapter degrades both performance and convergence speed, highlighting the importance of its integration within the core structure of our adapter.

CEA's operations include the following:

$$
\begin{aligned}
\bar{\boldsymbol{x}} &= \boldsymbol{\sigma}(\boldsymbol{W}_{\text{down}}^{\top} \cdot \boldsymbol{x}) \\
\bar{\boldsymbol{x}}_c &= \text{contract}_{HW}(\bar{\boldsymbol{x}}) \\
\hat{\boldsymbol{x}}_m &= \boldsymbol{W}_{\text{mid}}^{\top} \cdot \textbf{DWConv}_k(\bar{\boldsymbol{x}}_c) \\
\bar{\boldsymbol{x}}_e &= \text{expand}_{HW}(\hat{\boldsymbol{x}}_m) \\
\hat{\boldsymbol{x}} &= \bar{\boldsymbol{x}}_e + \bar{\boldsymbol{x}} \\
\boldsymbol{x}' &= \alpha \cdot \boldsymbol{W}_{\text{up}}^{\top} \cdot \hat{\boldsymbol{x}} + \boldsymbol{x}
\end{aligned}
\tag{1}
$$

where $\boldsymbol{x}$ and $\boldsymbol{x}'$ are the input and output features, and $\bar{\boldsymbol{x}}$ and $\hat{\boldsymbol{x}}$ are intermediate features, as illustrated in Fig. 2. $\boldsymbol{W}_{\text{down}}$

and $\boldsymbol{W}_{\text{up}}$ are projection weights, $\boldsymbol{W}_{\text{mid}}$ are weights of an intermediate fully-connected layer, $\textbf{DWConv}_k$ is the depthwise convolution, α is a learnable scalar, and $\boldsymbol{\sigma}(\cdot)$ is the GELU activation function [11].

During fine-tuning, only the CEA modules inserted between the backbone layers are trained, while the backbone remains frozen. With a temporal kernel size of 3 and a channel downsampling ratio of 4, the CEA comprises just 4.7% of the backbone's parameters. The CEA's GFLOPs are almost identical to the Standard Adapter. Compared to the Standard Adapter, CEA increases GFLOPs by an additional 0.04, while TIA increases it by an additional 5.8 GFLOPs.

3.2. Low-Frame-Rate Action Detection

While accounting for temporal context is widely recognized as beneficial for action segmentation, memory and time complexity constraints often limit the video length that can be analyzed. Therefore, temporal downsampling seems like a critical strategy to manage computational resources, especially for our end-to-end training. However, state-of-the-art (SOTA) action segmentation models typically struggle to maintain the high accuracy achieved at high frame rates (FPS) when applied to low-FPS input.

To enable efficient end-to-end training with temporal downsampling, we adopt a segmentation-by-detection framework, departing from SOTA approaches. Action proposals are predicted on coarsely sampled frames and then integrated at the original high frame rate for framewise classification. Inspired by anchor-free detectors (e.g., FCOS [34] and ActionFormer [41]), we treat each sampled frame as a query for its corresponding action proposal. This generates high-quality action proposals from low-FPS input. Compared to SOTA methods that rely on separate frame and action branches [27] or learnable queries [4], our approach significantly simplifies training by directly predicting action instances from sampled frames.

We first feed the backbone output $\mathcal{X}$ to a transformer encoder, which produces a multiscale feature pyramid $\mathcal{Z} = \{(\mathbf{z}_i^1, \ldots, \mathbf{z}_i^L)\}_{i=1}^{T'}$, capturing long-range temporal dependencies. This encoder consists of a shallow convolutional projection followed by a Transformer network with a multi-head self-attention, which operates at varying temporal scales by downsampling with strided depthwise 1D convolutions. The transformer encoder output is passed to a convolutional decoder with classification and regression heads. The classification head uses a 1D convolution across the L pyramid levels in $\mathcal{Z}$ to predict the class distribution of every frame, $\boldsymbol{\pi}_i = \{\pi_i(a) : a \in \mathcal{A}\}$, $i = \{1, \ldots, T'\}$, where $\mathcal{A}$ is the set of action classes including a background class, $\pi_i(a) \in [0, 1]$, and $\sum_{a\in\mathcal{A}} \pi_i(a) = 1$. Simultaneously, for every frame i, the regression head convolves $\mathcal{Z}$ across the levels to predict time offsets $\hat{d}_i^s$ and $\hat{d}_i^e$ to the start and end timestamps of the action instances to which ith frame belongs. In this way, every frame i generates the corresponding action proposal with the start and end timestamps estimated as $\hat{t}_i^s = t_i - \hat{d}_i^s$ and $\hat{t}_i^e = t_i + \hat{d}_i^e$.

In summary, the detector of EAST performs structured prediction $\mathcal{X} \rightarrow \{(\boldsymbol{\pi}_i, \hat{t}_i^s, \hat{t}_i^e)\}_{i=1}^{T'}$, which is mapped to the initial frame labels $\hat{\mathcal{Y}}_1$ as $\hat{y}_i = \text{argmax}_{a\in\mathcal{A}}\ \pi_i(a)$, and the set of T' action proposals $\hat{\mathcal{S}} = \{(\hat{t}_n^s, \hat{t}_n^e, \boldsymbol{\pi}_n)\}_{n=1}^{N}$, $N = T'$.

3.3. High-Frame-Rate Aggregation and Refinement

After generating action proposals $\hat{\mathcal{S}}$ from the downsampled $\mathbf{V}'$, they are combined to estimate the class distributions, $\boldsymbol{p}_i = \{p_i(a) : a\in\mathcal{A}\}$, of all frames in the unsampled $\mathbf{V}$. To this end, each frame $i = 1, \ldots, T$ of $\mathbf{V}$ aggregates the class distributions $\{\boldsymbol{\pi}_n\}$ of all proposals whose temporal intervals cover the timestamp of ith frame, $\hat{t}_n^s \le t_i \le \hat{t}_n^e$:

$$p_i(a) \propto \sum_{n=1}^{N} \pi_n(a) \cdot \mathbb{1}(\hat{t}_n^s \le t_i \le \hat{t}_n^e), \tag{2}$$

where "$\propto$" denotes proportionality up to a normalizing constant, $\mathbb{1}(\cdot)$ is a binary indicator, and $\sum_{a\in\mathcal{A}} p_i(a)=1$.

The aggregated class distributions of all frames, $\{\boldsymbol{p}_i\}_{i=1}^{T}$, are passed to a 3-stage temporal convolutional network (TCN) [9] for final frame classification $\hat{\mathcal{Y}}_2$. Similar refinement strategies are used in MS-TCN [9] and ASFormer [39]. The aggregation of action proposals in (2) helps improve temporal smoothness of frame labels in $\hat{\mathcal{Y}}_2$ relative to the initial prediction $\hat{\mathcal{Y}}_1$.

3.4. Training Loss Functions

In training, predictions $\hat{\mathcal{Y}}_1$ and $\hat{\mathcal{S}}$ over sampled frames in $\boldsymbol{V}'$, $i = 1, \ldots, T'$, incur loss, $\mathcal{L}$, defined as

$$\mathcal{L} = \frac{1}{T_+} \sum_{i=1}^{T'} \mathcal{L}_{\text{c}}(y_i, \hat{y}_i) + \lambda_{\text{r}} \mathbb{1}(\hat{y}_i) \mathcal{L}_{\text{r}}([t^s_{n(i)}, t^e_{n(i)}], [\hat{t}_i^s, \hat{t}_i^e]), \tag{3}$$

where $\mathcal{L}_{\text{c}}$ denotes the focal loss [24]; y_i is the ground-truth class; λ_{r} is a weighting hyperparameter; $\mathbb{1}(\cdot)$ is a binary indicator that equals 0 if ith frame is classified as background, and 1, otherwise; T_+ is the total number of sampled frames classified as an action $T_+ = \sum_{i=1}^{T'} \mathbb{1}(\hat{y}_i)$; $\mathcal{L}_{\text{r}}$ is the DIoU loss [42] for regression; $[t^s_{n(i)}, t^e_{n(i)}]$ denotes the time interval of nth ground-truth action instance closest to the predicted interval $[\hat{t}_i^s, \hat{t}_i^e]$ of ith action proposal. Note that the regression loss is applied only to action proposals classified as an action excluding background.

The final frame classification $\hat{\mathcal{Y}}_2$ is supervised with the cross entropy loss and smoothness loss, as in [9].

3.5. Proposal-Based Data Augmentation

This section introduces a new proposal-based data augmentation, which is seamlessly integrated into our end-to-end

training. It enforces high uncertainty in the input to the aggregation and TCN modules, reflecting the likely conditions during testing. Integrating existing data augmentation methods that manipulate or generate frame sequences into end-to-end training is challenging due to memory and complexity constraints. In contrast, our method is both efficient and effective, as it operates on significantly fewer proposals than frames in the video.

Our method randomly removes $K < A$ out of the top A most confident action proposals from $\hat{\mathcal{S}}$, resulting in $\hat{\mathcal{S}}'$. In our experiments, we choose $A = 30$, because videos of the existing action-segmentation datasets typically have a maximum of 30 action instances. Confidence of a proposal, κ_n, is estimated as its maximum class score $\kappa_n = \max_{a \in \mathcal{A}} \pi_n(a)$. This reduces the average confidence of the remaining proposals in $\hat{\mathcal{S}}'$ passed to the aggregation module, where they "compete" under increased uncertainty to assign their class distributions to frames in (2). It is worth noting that due to the random removals, $\hat{\mathcal{S}}'$ may still include some of the top-scoring proposals from $\hat{\mathcal{S}}$, which facilitates prediction of $\hat{\mathcal{Y}}_2$. Multiple random proposal removals are applied to generate multiple versions of $\hat{\mathcal{S}}'$, and thus perform data augmentation.

4. Results

Datasets. For evaluation, we use the GTEA [10], 50Salads [33], Breakfast [17], and Assembly101 [30] datasets.

- **GTEA** [10] consists of 28 egocentric videos, annotated with 11 action classes. The videos span approximately 1 minute and include around 19 action instances.
- **50Salads** [33] is a collection of 50 top-view videos of salad preparation with 17 action classes. The average length of these videos is 6 minutes, with approximately 20 action instances per video.
- **Breakfast** [17] consists of 1712 videos showing 48 breakfast preparation actions from a third-person perspective. The videos have an average length of 2 minutes, but they may significantly vary in duration.
- **Assembly101** [30] consists of 4321 videos and 202 action classes based on 11 verbs and 61 objects. The dataset features people assembling and disassembling 101 toys. The videos have on average 24 action instances over a duration of 7.1 minutes.

Consistent with previous work, we perform five-fold cross-validation on 50Salads, and four-fold cross-validation on GTEA and Breakfast, using the standard splits [4, 21, 38, 39]. For Assembly101, we use the official training and validation splits specified in [30].

Metrics. As in SOTA action segmentation approaches, we use framewise classification accuracy (Acc), edit score (Edit), and F1-scores (F1@$\{10, 25, 50\}$) at overlap thresholds of 10%, 25%, and 50% [18]. Edit score measures similarity between the predicted and ground-truth action sequences. F1-scores evaluate localization of action instances. We also report the average precision at Intersection over Union (IoU) thresholds of $\{0.3, 0.4, 0.5, 0.6, 0.7\}$ and the mean average precision (mAP) of our action proposals predicted by EAST's detector.

Implementation Details. EAST is implemented using PyTorch 2.0.1 and MMAction2 [7] on H100 GPUs. EAST consists of a backbone, detector, aggregator and TCN. The backbone is VideoMAEv2 [36] with ViT-G [40], pre-trained as in [36]. Parameters of the backbone are frozen to their pre-trained values, and fine-tuned with our CEA adapters placed between ViT-G blocks of VideoMAEv2, with the adapter's learning rate set to 2e-4. The adapter's projection layer weights and α are initialized to 0 and 1, respectively.

During training, video clips are randomly cropped to 768 frames. The frame sampling rate (FPS) is treated as an empirically optimized hyperparameter tested in Tab. 9. We use the following FPS: 3 for GTEA, 1 for 50salads, 3 for Breakfast, 6 for Assembly101. Backbone frame features, $\mathcal{X}$, are extracted using non-overlapping temporal windows of size 16 (stride = 16 frames) and a spatial resolution of 160x160. For inference on videos exceeding 768 frames, a 0.25 overlap sliding window approach is employed. Predicted action boundary timestamps are regressed directly in seconds. These are then multiplied by the FPS to generate high-FPS frame-wise predictions, ensuring consistency with prior work for comparison. EAST is trained in two stages for stable convergence. First, the backbone and detector are trained end-to-end for 300 epochs on GTEA, 150 epochs on 50Salads, 30 epochs on Breakfast, and 15 epochs on Assembly101. Subsequently, the entire EAST is trained for 50 epochs using our data augmentation method. As in [25, 27], model selection is performed on the validation set based on the average metrics.

4.1. Features and Training Efficiency

In this section, we compare EAST with SOTA on the two largest datasets Breakfast and Assembly101 to evaluate: frame feature representation and training efficiency. As SOTA representatives, for comparison, we choose FACT [27], LTContext [3], ASFormer [39], and MSTCN [9].

The SOTA methods use precomputed I3D features [6] for Breakfast and TSM features [23] for Assembly101. To ensure consistency in the frame features across the SOTA methods and EAST, we extracted MAEv2 frame features for all the methods, including ours, using the ViT-G backbone pretrained with VideoMAEv2 [36]. Tab. 1 shows that the SOTA methods improve performance on Breakfast when using precomputed MAEv2 features, while LTContext with MAEv2 features fails to do so on Assembly101. On both datasets, end-to-end trained EAST using the same pretrained ViT-G achieves superior performance.

Tab. 4 compares per-epoch time, total training time, and

Method	Feature	F1@{10,25,50}			Edit	Acc
MSTCN [9]	I3D	52.6	48.1	37.9	61.7	66.3
('CVPR19)	MAEv2*	59.9	55.1	43.9	65.0	68.5
ASFormer [39]	I3D	76.0	70.6	57.4	75.0	73.5
('BMCV21)	MAEv2*	78.7	73.6	60.8	76.8	75.0
LTContext [3]	I3D	77.6	72.6	60.1	77.0	74.2
('ICCV23)	MAEv2*	80.6	75.7	64.1	75.2	76.6
FACT [27] ('CVPR24)	I3D	81.4	76.5	66.2	79.7	76.2
	I3D*	78.4	73.3	62.2	77.4	75.5
	MAEv2*	80.6	75.9	65.3	78.9	77.2
EAST	MAEv2	**85.6**	**81.5**	**71.6**	**83.5**	**82.2**

Table 1. Impact of feature representation on Breakfast. *: results from official code. SOTA methods use precomputed I3D features on Breakfast. To ensure a consistent feature representation, we compare SOTA and EAST using MAEv2 frame features extracted by the ViT-G backbone pretrained with VideoMAEv2 [36] across all the methods. Results are averaged across four standard splits.

Method	Feature	F1@{10,25,50}			Edit	Acc
LTContext [3]	TSM	33.9	30.0	22.6	30.4	41.2
('ICCV23)	MAEv2	31.3	27.9	21.1	27.8	40.3
EAST	MAEv2	**42.3**	**39.4**	**32.8**	**39.9**	**48.4**

Table 2. Impact of feature representation on Assembly101. LTContext [3] uses precomputed TSM features on Assembly101. For fair comparison, we evaluate LTContext and EAST when both methods use MAEv2 frame features extracted by the ViT-G backbone pretrained with VideoMAEv2 [36].

Methods	MSTCN++	ASFormer	LTContext	DiffAct	FACT	EAST
GFLOPS	4.5	7.6	7.8	63.0	5.5	9.2

Table 3. Computational cost in GFLOPS, measured on a 5-minute video using the setup from [27].

GPU memory usage of SOTA methods and EAST on Breakfast using NVIDIA H100s. Note that the SOTA methods in Tab. 4 rely on pre-computed features and in their reports exclude feature extraction time and memory usage. Therefore, the comparison of training efficiency is unfavorable to EAST, whose reported numbers in Tab. 4 account for the full pipeline, including training the large backbone. Tab. 4 shows EAST achieves total training time comparable to LTContext – the fastest feature-based Transformer – despite processing RGB frames.

For a fair comparison with prior work that does not support direct RGB input and relies on pre-computed features, we account for feature extraction costs in Tab. 14 in the supplement. As Tab. 14 demonstrates, SOTA methods require significantly more resources – e.g., FACT needs 300.3 hours for I3D feature extraction plus 122.3 hours for model training (total 422.6 hours), making it 36.4× longer than EAST's end-to-end training. EAST uses only 40GB of memory during training, which is highly efficient considering the ViT-G backbone alone requires 6GB for per-frame feature extraction. In contrast, prior work requires 25TB to extract I3D features for a 10-minute video. Our efficiency

Method	MSTCN	ASFormer	LTContext	FACT	EAST
Time per Epoch (min)	25.0	99.0	4.5	48.9	23.3
Number of Epochs	50	120	150	150	30
Training Duration (h)	20.8	198.0	11.3	122.3	11.6
Peak GPU Memory (MiB)	2464	5014	7358	18990	40804

Table 4. Comparison of training efficiency on Breakfast in terms of per-epoch time, total training time, total epochs, and memory usage on an H100 GPU. Our EAST matches MSTCN's per-epoch time, and reduces overall training time due to faster convergence of learning and capability to work with downsampled videos. Higher memory usage of EAST is due to taking the RGB input rather than pre-computed features.

Dataset	Method	0.3	0.4	0.5	0.6	0.7	mAP
GTEA	No adapter	89.4	87.1	83.0	74.0	66.1	79.9
	Standard [12]	94.9	92.9	90.3	83.4	75.6	87.4
	TIA [26]	94.7	92.9	88.2	83.1	76.4	87.1
	CEA	**95.2**	**93.8**	**91.1**	**84.9**	**77.9**	**88.6**
Breakfast	No adapter	64.7	61.9	58.3	52.0	42.8	55.9
	Standard [12]	73.8	71.3	68.0	62.2	53.9	65.9
	TIA [26]	74.9	72.5	69.2	63.7	54.8	67.0
	CEA	**75.8**	**73.2**	**70.1**	**63.9**	**55.7**	**67.7**

Table 5. Average precision at varying IoUs and mAP of EAST's detector on Breakfast for different adapters used in fine-tuning of the ViT-G backbone. The row denoted with "No adapter" reports the results when ViT-G is not fine-tuned, i.e., when EAST uses pre-computed MAEv2 frame features of the frozen ViT-G pretrained with VideoMAEv2.

stems from segmentation-by-detection enabling low frame-sampling rates (1-6 fps) versus high-FPS SOTA methods. For inference on a 10-minute 50Salads video, EAST uses only 4.3K GFLOPs—lower than all prior work when combining their feature extraction and model GFLOPs (Tab 3). While EAST has 152.58M parameters (mainly ViT-G), the added components are proportionally smaller than SOTA relative to their backbones (e.g., I3D's 24.57M parameters).

4.2. CEA Adapter and Backbones

This section evaluates the proposed CEA adapter for fine-tuning the backbone alongside different backbone networks. We analyze their impact on EAST's detector performance, as the accuracy of its output – i.e., action proposals – is critical for the effectiveness of the subsequent aggregation and refinement modules. To assess this, we report average precision across varying IoU thresholds and mAP.

Adapter. Table 5 compares the performance of EAST's detector on GTEA and Breakfast when using different adapters for fine-tuning the ViT-G backbone, including the Standard Adapter [12], TIA [26], and our proposed CEA. We also conduct an ablation study where ViT remains frozen during training, without any adapter-based fine-tuning. In this setting, EAST effectively operates with pre-computed MAEv2 frame features extracted from the ViT-G backbone pretrained with VideoMAEv2 [36]. Our CEA gives the best performance across all metrics. Regarding

Methods	Standard [12]	TIA [26]	CEA
GFLOPs	46.06	51.86	46.10

Table 6. Computational cost (GFLOPs) of different adapters used for fine-tuning the backbone on a video sequence of 768 frames.

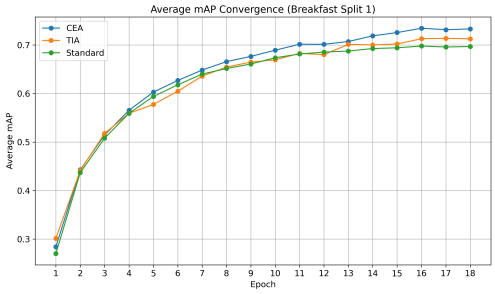

Figure 3. The plots of EAST's detector mAP vs. training epochs on Breakfast, for different adapters placed in the backbone, including the Standard Adapter[12], TIA [26], and our proposed CEA. The CEA adapter consistently achieves higher mAP across all epochs, and exhibits a faster convergence rate compared to the Standard and TIA adapters.

Backbone	Dim	Mem	0.3	0.4	0.5	0.6	0.7	mAP
ViT-S	384	3787	66.4	63.7	58.7	53.3	43.9	57.2
ViT-B	768	9434	68.9	66.6	62.8	55.3	47.3	60.2
ViT-L	1024	16920	73.1	69.4	65.6	58.3	49.3	63.1
ViT-G	1408	40804	78.8	76.3	72.4	65.0	56.7	69.8

Table 7. Evaluation of EAST's detector on Breakfast when using different backbone networks. The table reports the backbone's output feature dimension (Dim), GPU memory usage (Mem, in MiB) during training, average precision at varying IoUs, and mAP.

computational efficiency, Table 6 shows that our CEA requires only 0.04 more GFLOPs than the Standard Adapter, whereas TIA incurs an additional 5.8 GFLOPs, demonstrating CEA's significantly lower computational cost compared to TIA. Fig. 3 presents the plots of mAP vs. training epochs on Breakfast, for different adapters placed in the backbone, showing that CEA consistently achieves the best mAP in every epoch.

Backbone. Tab. 7 compares EAST's detector performance on Breakfast when using different ViT backbones, from the smallest ViT-S to the largest ViT-G, pre-trained with VideoMAE [35] and VideoMAEv2 [36]. ViT-G is pre-trained on the unlabeled hybrid dataset [36], while the other ViT variants are pre-trained on the Kinetics-400 dataset [43]. Tab. 7 shows that EAST with ViT-G outperforms the other alternatives across all metrics. Therefore, we select ViT-G as our default backbone, and set that its output fine-tuned frame features $\mathcal{X}$ have a dimensionality of 1408. In contrast, previous work uses pre-computed I3D [6] or TSM [23] features with a dimensionality of 2048.

Dataset	Method	F1@10	F1@25	F1@50	Edit	Acc
GTEA	Baseline	87.3	84.7	75.9	82.5	86.3
	MSTCN	93.2	93.0	88.7	92.1	83.8
	ASFormer	86.8	84.0	74.3	83.1	83.6
	Proposal Aug	95.8	95.4	91.7	95.4	87.1
Breakfast	Baseline	78.2	74.2	65.8	75.1	77.6
	MSTCN	85.6	81.3	71.5	83.4	80.3
	ASFormer	84.1	79.9	69.7	82.2	80.2
	Proposal Aug	86.2	82.2	71.8	84.5	82.8

Table 8. Action segmentation results using identical action proposals from our detector, comparing refinement strategies. "Baseline" denotes framewise classification via Eq. 2 without augmentation (Sec. 3.5). "MSTCN" and "ASFormer" apply their respective refinement networks to our generated proposals. "Proposal Aug" (EAST) incorporates our augmentation during training for improved refinement learning. The comparison between Baseline and Proposal Aug isolates our proposed augmentation' impact.

FPS	F1@{10,25,50}			Edit	Acc
1	84.1	79.8	69.6	81.7	80.4
3	86.2	82.2	71.8	84.5	82.8

Table 9. Sensitivity of EAST performance to the frame sampling rate (FPS) on Breakfast.

Backbone	Method	F1@{10,25,50}			Edit	Acc
Frozen	MSTCN('CVPR19) [9]	85.8	83.4	69.8	79.0	76.3
	ASFormer('BMCV21) [39]	90.1	88.8	79.2	84.6	79.7
	UVAST('ECCV22) [4]	92.7	91.3	81.0	92.1	80.2
	RTK('ICCV23) [16]	91.2	90.6	83.4	87.9	80.3
	DiffAct ('ICCV23) [25]	92.5	91.5	84.7	89.6	82.2
	FACT ('CVPR24 [27])	93.5	92.1	84.1	91.4	86.1
	BaFormer ('NIPS24 [37])	92.0	91.3	83.5	88.7	83.0
ViT-G fine-tuned	EAST	**95.8**	**95.4**	**91.7**	**95.4**	**87.1**

Table 10. Action segmentation on GTEA.

4.3. Aggregator and Sensitivity to FPS

Tab. 8 reports the action segmentation performance of EAST's aggregation and refinement modules on GTEA and Breakfast. It also includes ablations where we replace the aggregator and TCN module with the refinement stages of MS-TCN [9] and ASFormer [39], or omit refinement of EAST's detector output altogether. The results demonstrate that EAST's aggregation and refinement gives the best segmentation performance across all metrics on both datasets.

Tab. 9 evaluates EAST's sensitivity to the video downsampling rate on Breakfast. EAST maintains SOTA performance at low FPS rates and improves as the frame rate increases, subject to memory and compute constraints.

4.4. Comparison with SOTA

Tables 10–13 compare EAST with SOTA on four datasets. EAST consistently outperforms prior work across all datasets, achieving substantial improvements in all metrics. On Assembly101, EAST surpasses previous methods by 7.2 points in accuracy and 9.5 points in Edit score. Furthermore, EAST demonstrates F1@50 score gains of 7.0, 3.6,

Backbone	Method	F1@{10,25,50}			Edit	Acc
Frozen	MSTCN('CVPR19) [9]	76.3	74.0	64.5	67.9	80.7
	ASFormer('BMCV21) [39]	85.1	83.4	76.0	79.6	85.6
	UVAST('ECCV22)[4]	89.1	87.6	81.7	83.9	87.4
	RTK('ICCV23) [16]	87.4	86.1	79.5	81.4	85.9
	LTContext('ICCV23) [3]	89.4	87.7	82.0	83.2	87.7
	DiffAct ('ICCV23) [25]	90.1	89.2	83.7	85.0	88.9
	BaFormer ('NIPS24 [37])	89.3	88.4	83.9	84.2	89.5
ViT-G fine-tuned	EAST	**93.1**	**91.9**	**88.6**	**88.4**	**91.9**

Table 11. Action segmentation on 50salads.

Backbone	Method	F1@{10,25,50}			Edit	Acc
Frozen	MSTCN('CVPR19) [9]	52.6	48.1	37.9	61.7	66.3
	ASFormer('BMCV21) [39]	76.0	70.6	57.4	75.0	73.5
	UVAST('ECCV22)[4]	75.9	70.0	57.2	76.5	66.0
	RTK('ICCV23) [16]	76.9	72.4	60.5	76.1	73.3
	LTContext('ICCV23) [3]	77.6	72.6	60.1	77.0	74.2
	DiffAct ('ICCV23) [25]	80.3	75.9	64.6	78.4	76.4
	FACT ('CVPR24) [27]	81.4	76.5	66.2	79.7	76.2
	BaFormer ('NIPS24 [37])	79.2	74.9	63.2	77.3	76.6
ViT-G fine-tuned	EAST	**86.2**	**82.2**	**71.8**	**84.5**	**82.8**

Table 12. Action segmentation Breakfast.

Backbone	Method	F1@{10,25,50}			Edit	Acc
Frozen	MS-TCN++ (PAMI'20) [21]	31.6	27.8	20.6	30.7	37.1
	UVAST (ECCV'22) [4]	32.1	28.3	20.8	31.5	37.4
	ASFormer ('BMCV21) [39]	33.4	29.2	21.4	30.5	38.8
	C2F-TCN (TPAMI'23) [32]	33.3	29.0	21.3	32.4	39.2
	LTContext ('ICCV23) [3]	33.9	30.0	22.6	30.4	41.2
ViT-G fine-tuned	EAST	**42.3**	**39.4**	**32.8**	**39.9**	**48.4**

Table 13. Action segmentation on Assembly101.

5.2, and 10.2 percentage points on the GTEA, 50Salads, Breakfast, and Assembly101 datasets, respectively.

While maintaining comparable per-epoch processing times to prior work, EAST significantly reduces the total number of training epochs required for convergence. On Breakfast, EAST converges in 30 epochs, compared to 150 for FACT [27] and 1000 for DiffAct [25]. On Assembly101, EAST requires only 12 epochs, whereas LTContext [3] and C2F-TCN [32] require 120 and 200 epochs.

4.5. Qualitative Results

Fig. 4 compares of EAST segmentation results with the ground truth and SOTA on sample videos from GTEA, Breakfast, and Assembly101. In the top example video from GTEA, EAST successfully detects all actions, whereas the other methods miss at least one action. On the middle example video from Breakfast, the SOTA methods exhibit varying degrees of missed actions, spurious predictions, or oversegmentation. In contrast, EAST provides high-quality segmentation closely aligned with the ground truth. Finally, in the bottom video from Assembly101, EAST fails to detect a brief action instance and incorrectly identifies an action boundary too early – both of these are ambiguous edge cases that are challenging to discern even through visual inspection.

Figure 4. Segmentation results on sample videos from GTEA (top), Breakfast (middle), and Assembly101 (bottom). For each video sequence, the top row shows the color-coded ground-truth action instances, middle row shows EAST output, and bottom row shows SOTA results generated using publicly available models.

5. Conclusion

We have introduced EAST – the first fully end-to-end trainable action segmenter. EAST performs segmentation by detection, which enables temporal downsampling of input videos, significantly reducing computational costs. EAST takes RGB frames, sampled at a low frame rate, as input to a large-scale backbone. The backbone is fine-tuned using our Contract-Expand Adapter (CAE). CAE is especially effective in reducing the computational costs of end-to-end training by leveraging spatial pooling. The backbone features are passed to the detector to predict action proposals, which are then aggregated and refined to produce final framewise labeling at the original unsampled frame rate. We have also specified a novel proposal-based data augmentation that increases uncertainty of the detector's output during training, effectively simulating test-time conditions. EAST outperforms prior work across all metrics on the GTEA, 50Salads, Breakfast, and Assembly101 datasets, while maintaining comparable processing times, even with the additional step of extracting frame features by the backbone. We have conducted a comprehensive ablation study to evaluate EAST's performance under various configurations. Additional results are presented in the supplements.

Acknowledgement: This work has been supported by USDA NIFA award No.2021-67021-35344.

References

[1] Mohamed R. Amer, Sinisa Todorovic, Alan Fern, and Song-Chun Zhu. Monte carlo tree search for scheduling activity recognition. In *2013 IEEE International Conference on Computer Vision*, pages 1353–1360, 2013. 1

[2] Nicolas Aziere and Sinisa Todorovic. Markov game video augmentation for action segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 13505–13514, 2023. 1, 2

[3] Emad Bahrami, Gianpiero Francesca, and Juergen Gall. How much temporal long-term context is needed for action segmentation? In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10351–10361, 2023. 5, 6, 8, 11

[4] Nadine Behrmann, S Alireza Golestaneh, Zico Kolter, Jürgen Gall, and Mehdi Noroozi. Unified fully and timestamp supervised temporal action segmentation via sequence to sequence translation. In *ECCV*, 2022. 2, 4, 5, 7, 8

[5] William Brendel and Sinisa Todorovic. Learning spatiotemporal graphs of human activities. In *IEEE International Conference on Computer Vision, ICCV*, pages 778–785, 2011. 1

[6] Joao Carreira and Andrew Zisserman. Quo vadis, action recognition? a new model and the Kinetics dataset. In *CVPR*, 2017. 1, 5, 7

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

[8] Guodong Ding, Fadime Sener, and Angela Yao. Temporal action segmentation: An analysis of modern techniques. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2023. 1

[9] Yazan Abu Farha and Jurgen Gall. MS-TCN: Multi-stage temporal convolutional network for action segmentation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 3575–3584, 2019. 1, 2, 3, 4, 5, 6, 7, 8, 11

[10] Alireza Fathi, Xiaofeng Ren, and James M Rehg. Learning to recognize objects in egocentric activities. In *CVPR*, 2011. 5

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

[12] Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for nlp. In *ICML*, 2019. 2, 3, 6, 7

[13] Andrew G Howard. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017. 2, 3

[14] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. In *ICML*, 2021. 2

[15] Yifei Huang, Yusuke Sugano, and Yoichi Sato. Improving action segmentation via graph-based temporal reasoning. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 14021–14031, 2020. 1

[16] Borui Jiang, Yang Jin, Zhentao Tan, and Yadong Mu. Video action segmentation via contextually refined temporal keypoints. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 13836–13845, 2023. 7, 8

[17] Hilde Kuehne, Ali Arslan, and Thomas Serre. The language of actions: Recovering the syntax and semantics of goal-directed human activities. In *CVPR*, 2014. 5

[18] Colin Lea, Michael D. Flynn, Rene Vidal, Austin Reiter, and Gregory D. Hager. Temporal convolutional networks for action segmentation and detection, 2016. 5

[19] Jun Li and Sinisa Todorovic. Action shuffle alternating learning for unsupervised action segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021. 1

[20] Muheng Li, Lei Chen, Yueqi Duan, Zhilan Hu, Jianjiang Feng, Jie Zhou, and Jiwen Lu. Bridge-prompt: Towards ordinal action understanding in instructional videos. In *CVPR*, 2022. 1

[21] Shi-Jie Li, Yazan Abu Farha, Yun Liu, Ming-Ming Cheng, and Juergen Gall. MS-TCN++: Multi-stage temporal convolutional network for action segmentation. *IEEE TPAMI*, 2020. 5, 8

[22] Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation. In *ACL*, 2021. 2

[23] Ji Lin, Chuang Gan, and Song Han. Tsm: Temporal shift module for efficient video understanding. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 7083–7093, 2019. 1, 5, 7

[24] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 42(2):318–327, 2020. 4

[25] Daochang Liu, Qiyue Li, Anh-Dung Dinh, Tingting Jiang, Mubarak Shah, and Chang Xu. Diffusion action segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10139–10149, 2023. 1, 2, 5, 7, 8

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

[27] Zijia Lu and Ehsan Elhamifar. FACT: Frame-action cross-attention temporal modeling for efficient supervised action segmentation. In *Conference on Computer Vision and Pattern Recognition 2024*, 2024. 1, 2, 4, 5, 6, 7, 8, 11

[28] Ishan Misra, C. Lawrence Zitnick, and Martial Hebert. Shuffle and learn: Unsupervised learning using temporal order verification. In *14th European Conference on Computer Vision ECCV*, 2016. 1

[29] Mingtao Pei, Yunde Jia, and Song Zhu. Parsing video events with goal inference and intent prediction. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 487–494, 2011. 1

[30] Fadime Sener, Dibyadip Chatterjee, Daniel Shelepov, Kun He, Dipika Singhania, Robert Wang, and Angela Yao. As-

sembly101: A large-scale multi-view video dataset for understanding procedural activities. In *CVPR*, 2022. 5

[31] Zhangzhang Si, Mingtao Pei, Benjamin Z. Yao, and Song-Chun Zhu. Unsupervised learning of event AND-OR grammar and semantics from video. In *IEEE International Conference on Computer Vision, ICCV*, pages 41–48, 2011. 1

[32] Dipika Singhania, Rahul Rahaman, and Angela Yao. Coarse to fine multi-resolution temporal convolutional network. *arXiv preprint arXiv:2105.10859*, 2021. 8

[33] Sebastian Stein and Stephen J McKenna. Combining embedded accelerometers with computer vision for recognizing food preparation activities. In *Proceedings of the 2013 ACM international joint conference on Pervasive and ubiquitous computing*, 2013. 5

[34] Zhi Tian, Chunhua Shen, Hao Chen, and Tong He. Fcos: Fully convolutional one-stage object detection. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 9627–9636, 2019. 4

[35] Zhan Tong, Yibing Song, Jue Wang, and Limin Wang. Videomae: Masked autoencoders are data-efficient learners for self-supervised video pre-training. In *NeurIPS*, 2022. 7

[36] Limin Wang, Bingkun Huang, Zhiyu Zhao, Zhan Tong, Yinan He, Yi Wang, Yali Wang, and Yu Qiao. Videomae v2: Scaling video masked autoencoders with dual masking. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14549–14560, 2023. 3, 5, 6, 7

[37] Peiyao Wang, Yuewei Lin, Erik Blasch, Jie Wei, and Haibin Ling. Efficient temporal action segmentation via boundary-aware query voting. *arXiv preprint arXiv:2405.15995*, 2024. 1, 2, 7, 8

[38] Ziwei Xu, Yogesh S Rawat, Yongkang Wong, Mohan Kankanhalli, and Mubarak Shah. Don't pour cereal into coffee: Differentiable temporal logic for temporal action segmentation. In *NeurIPS*, 2022. 5

[39] Fangqiu Yi, Hongyu Wen, and Tingting Jiang. Asformer: Transformer for action segmentation. In *British Machine Vision Conference*, 2021. 1, 2, 4, 5, 6, 7, 8, 11

[40] Xiaohua Zhai, Alexander Kolesnikov, Neil Houlsby, and Lucas Beyer. Scaling vision transformers. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 12104–12113, 2022. 3, 5

[41] Chen-Lin Zhang, Jianxin Wu, and Yin Li. Actionformer: Localizing moments of actions with transformers. In *European Conference on Computer Vision*, pages 492–510. Springer, 2022. 4

[42] Zhaohui Zheng, Ping Wang, Wei Liu, Jinze Li, Rongguang Ye, and Dongwei Ren. Distance-iou loss: Faster and better learning for bounding box regression. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(07):12993–13000, 2020. 4

[43] Andrew Zisserman, Joao Carreira, Karen Simonyan, Will Kay, Brian Zhang, Chloe Hillier, Sudheendra Vijayanarasimhan, Fabio Viola, Tim Green, Trevor Back, et al. The kinetics human action video dataset. *arXiv preprint arXiv*, 1705, 2017. 7