FACT: Frame-Action Cross-Attention Temporal Modeling for Efficient Action Segmentation

Zijia Lu
Northeastern Univeristy
lu.zij@northeastern.edu

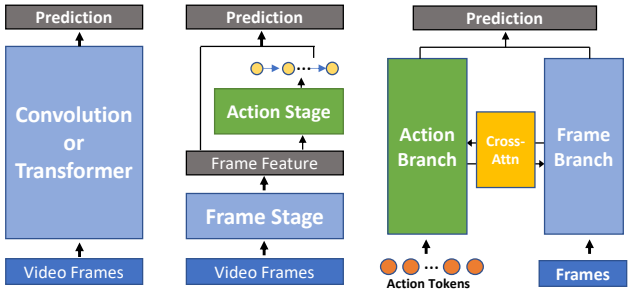
Ehsan Elhamifar
Northeastern University
e.elhamifar@northeastern.edu

Abstract

We study supervised action segmentation, whose goal is to predict framewise action labels of a video. To capture temporal dependencies over long horizons, prior works either improve framewise features with transformer or refine framewise predictions with learned action features. However, they are computationally costly and ignore that frame and action features contain complimentary information, which can be leveraged to enhance both features and improve temporal modeling. Therefore, we propose an efficient Frame-Action Cross-attention Temporal modeling (FACT) framework that performs temporal modeling with frame and action features in parallel and leverage this parallelism to achieve iterative bidirectional information transfer between the features and refine them. FACT network contains (i) a frame branch to learn frame-level information with convolutions and frame features, (ii) an action branch to learn action-level dependencies with transformers and action tokens and (iii) cross-attentions to allow communication between the two branches. We also propose a new matching loss to ensure each action token uniquely encodes an action segment, thus better captures its semantics. Thanks to our architecture, we can also leverage textual transcripts of videos to help action segmentation. We evaluate FACT on four video datasets (two egocentric and two third-person) for action segmentation with and without transcripts, showing that it significantly improves the state-of-the-art accuracy while enjoys lower computational cost (3 times faster) than existing transformer-based methods.[1]

1. Introduction

Automatic understanding of human actions in videos is crucial for many applications, such as assistive technologies [13, 31, 51], healthcare, robotics and security. Many real-world applications, however, require understanding of long and untrimmed videos, which has motivated a large body of recent research on action segmentation [1, 7, 16, 22, 24, 37, 40, 55, 56, 61, 62]. The goal of action segmentation is to assign a label for every video frame, hence partition a long video into non-overlapping action segments.



(a) Frame-Based Method (b) Two-Stage Method (c) Our FACT Method

Figure 1. Different Action Segmentation Models.

Prior Works. Existing action segmentation methods can be divided into two main categories. The first group of methods uses a frame-based approach by learning temporal relations among frames and refining the feature representations and predictions using a multi-stage mechanism [16, 24, 37, 55, 61], see Figure 1 (a). In particular, MSTCN [16], MSTCN++ [37] and [24, 55] employ dilated temporal convolutions. However, they have difficulty in capturing long-range temporal dependencies and suffer from oversegmentation. To mitigate these issues, ASFormer [61], DiffAct [40] and [3, 5, 59] use transformers. However, given the high complexity of attention in transformers, they resort to windowed temporal attention, which still limits their ability for long-range temporal modeling.

To address the drawbacks of frame-based approaches, the second group of methods, such as MuCon [56], HARF [1], UVAST [7] and RTK [25] adopt a two-stage framework to learn action features and relations, which captures long-range temporal information. They compute frame features in the first stage, followed by constructing action features and predictions in the second stage, see Figure 1(b). However, they only fuse the outputs of two stages by merging their predictions. Given that action features capture high-level action dependencies and frame features capture low-level details, their complimentary information should be exploited

[1]Code available at github.com/ZijiaLewisLu/CVPR2024-FACT.

to refine both features and achieve better temporal modeling, which has been ignored in those works. Additionally, they often suffer from high computation during inference, as UVAST [7], RTK [25] require transformer for frame-level modeling, while MuCon [56] and UVAST [7] require a costly post-processing (Viterbi decoding) at the inference time.

Paper Contributions. We propose a new framework for action segmentation, referred to as **F**rame-**A**ction **C**ross-attention **T**emporal modeling (**FACT**), where our key idea is to learn a temporal model with both frame and action features, and conduct bidirectional information transfer between these features to refine them. Specifically, the FACT architecture consists of a frame branch (blue) for frame-level modeling with frames as input, an action branch (green) for action-level modeling with learned action tokens as input and cross-attentions (yellow) for cross-branch communication and feature refinement, see Figure 1(c). Our framework has the following contributions:

— FACT performs frame- and action-level temporal modeling *in parallel* (unlike two-stage methods), hence conveniently allows information transfer between the two levels. To model action relations, we learn a group of action tokens as input to the action branch with a new matching loss that guides FACT to assign the tokens to encode action segments.

— FACT achieves better temporal modeling than existing methods thanks to the cross-branch communication that exploits the complimentary information in action and frame branches and refines their features. Consequently, we only need to apply *convolution in the frame branch*, while majority of recent works need transformers, which incurs large complexity [5, 7, 25, 40]. On the other hand, we apply *transformer in the action branch* without a large computation overhead, as the number of action tokens is much smaller than the number of frames. FACT achieves high performance with a low inference cost, see Figure 2.

— FACT can also *leverage textual knowledge*, such as video transcripts [9, 12, 35, 41, 48, 56], during training and inference by using them to initialize action tokens, which improves accuracy with even less training data.

Finally, by extensive experiments on benchmark datasets [17, 26] and also two new challenging datasets [6, 11] that feature long complex videos, we show that FACT improves over state of the art while maintaining low inference time.

2. Related Works

Action Segmentation. Action segmentation has been studied in unsupervised [2, 14, 15, 20, 29, 52, 64], weakly-supervised [9, 12, 18, 33–35, 38, 41, 42, 46–48, 50, 56] and full-supervised [1, 7, 16, 23, 27, 30, 31, 36, 37, 40, 45, 49, 51, 53–56, 60–62] settings. FACT focuses on the fully-supervised setting. For frame-based methods, MSTCN [16] and subsequent works [37, 55] build multiple blocks

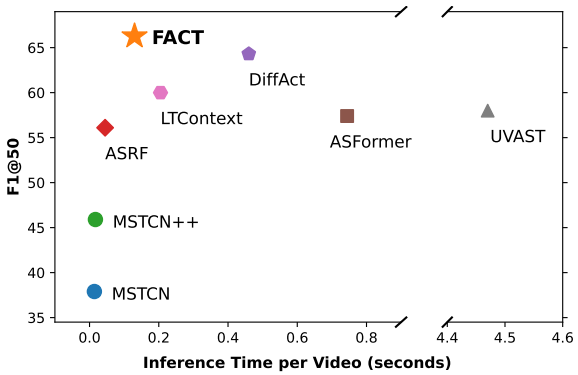


Figure 2. Performance vs Inference Time on Breakfast dataset: FACT outperforms existing methods while being *3.5 times faster* than the best previous method (DiffAct).

of temporal convolution, which enable feature refinement and computational efficiency, yet cannot capture temporal relations longer than the receptive fields. ASFormer [61] and [3, 5] replaces the convolution with transformer for better temporal modeling, but leads to higher computational cost, despite using windowed self attention. Recently, DiffAct [40] extends the multi-block refinement mechanism to a diffusion process, which further increases training and inference complexity.

On the other hand, Two-Stage methods [1, 7, 22, 25] recognize the importance of modeling action relations, thus first learn initial frame features and predictions, then estimate action features based on them and refine the predictions. Specifically, UVAST [7] refines predictions by aligning frame labels to its predicted action sequences and RTK [25] uses graph convolutions to update the labels of detected key-points. However, these methods lack *bidirectional* knowledge transfer between action and frame features, meaning the learned action relations cannot help improving frame features. In fact, action and frame features contain complimentary information, thus iteratively refining both features with each other leads to substantial improvement in temporal modeling. Therefore, we propose FACT to *simultaneously* perform temporal modeling on frame-level and action-level, hence allow bidirectional information transfer between the two levels and iterative feature refinement. Lastly, MuCon [56] explores a two-branch network in weakly-supervised setting to achieve cross-supervision between branches, yet it lacks communications between branches thus has the same limitations as the two-stage methods. [4, 59] propose model-agnostic techniques to improve action segmentation while FACT is a new model design for better temporal modeling and can be combined with them. [36, 62] employ a pre-trained vision-language model [58] to enable prompt learning thus use more training data than our and other works.

Matching Loss. We leverage action tokens to not only learn action relations but also the action labels and locations of action segments. While we allow FACT to dynamically

assign one action token to encode each action segment, we introduce a matching loss (see Section 3.2 for details) to ensure each token uniquely represents one or multiple action segments without overlap, i.e., *two tokens cannot represent the same segment*. While recent object detection methods, e.g., DETR[8] and [10, 43, 44, 63], apply a similar matching loss to encode objects in an image by object tokens, we are the first to adopt and extend the matching loss for action segmentation. The loss in DETR assigns one object token per object (one-to-one), as each object is unique, and obtains the optimal matching by Hungarian algorithm. However, videos often contain repeated actions that have similar semantics and can be encoded by a shared token, e.g., the longest video in EPIC-Kitchen dataset [11] contains 1,140 action segments with 78% of them being repeated. In such a case, it is inefficient to learn one token per action segment. Hence, we extend the loss to also allow one token encoding multiple segments (one-to-many), while ensuring that the segments has the same action class, which greatly reduces the number of required action tokens and computation complexity. We propose a new algorithm to estimate the optimal matching, which is either one-to-one or one-to-many based on the statistic of a dataset. Lastly, [7, 56] generate action features as a sequence, where their ordering indicates the matching to the action segments. Thus, such methods only allow one-to-one matching as the ordering becomes ambiguous if one feature can correspond to several segments at different locations. The imposed sequential dependency between features also leads to lower performance, as we show in ablation study and Figure 6.

3. Proposed Framework

To capture temporal relationships in long complex videos, we propose FACT, which performs temporal modeling simultaneously at both frame-level and action-level with iterative bidirectional information transfer between the two levels.

Given a video with T frames and pre-extracted[2] frame-wise features, our goal is to predict the action labels of all frames $\boldsymbol{y} = [y_1, \ldots, y_T] \in [1, \ldots, A]^T$, where A is the number of action classes. As shown in Figure 3, we depart from the previous two-stage frameworks [1, 7, 25] that build action features only after obtaining reliable frame features. More specifically, we use a frame branch (blue) that receives frame features to learn frame-level details via convolutions and a *parallel* action branch (green) that receives action tokens to learn action-level relationships via transformers. We partition these two branches into an input block for initial feature learning and multiple update blocks with bidirectional cross-attentions (yellow) in each block for cross-branch communication and iterative refinement of action token and frame features.

[2]While similar to prior works we use pre-extracted features to save computation, FACT can be made end-to-end.

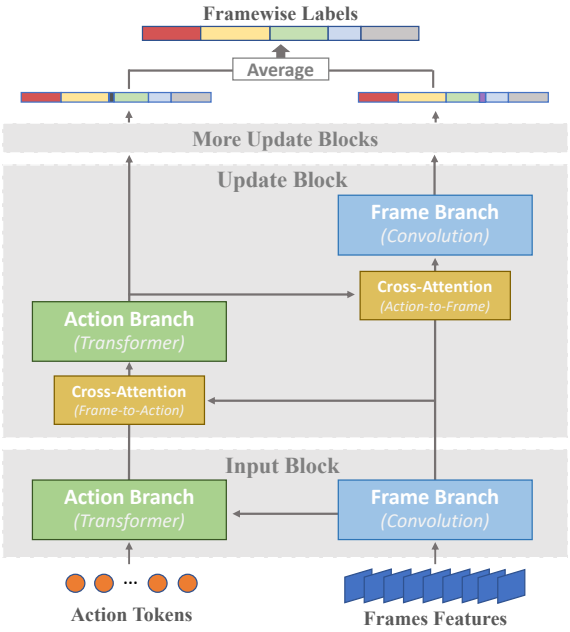


Figure 3. FACT performs temporal modeling simultaneously at frame- and action-level with bidirectional information transfer between two levels.

Ideally, we expect each action token to uniquely represent an action segment, thus encoding its action label and temporal location. However, it leaves the number of required action tokens for a new video unknown. To address this challenge, first, we use a *fixed-number of tokens*, M, as input, where M must be larger than the typical number of action segments in a video. Second, we design a new matching loss that guides FACT to dynamically assign a subset of tokens to encode action segments and the excessive ones to a special null class. Next, we discuss the details of our network in Section 3.1 and our loss functions in Section 3.2.

3.1. FACT's Network Architecture

FACT has an *input block* for initialization and multiple *update blocks* for refining frame features and action tokens.

Input Block. We use this block to perform the initial feature learning for action tokens and frames. Let $\mathbf{A}_0 \in \mathbb{R}^{M \times D}$ and $\mathbf{F}_0 \in \mathbb{R}^{T \times D}$ denote, respectively, the initial action tokens and pre-extracted frame features, where M is the number of action tokens, T is the number of frames and D is the feature dimension. We initialize the action tokens as $\mathbf{A}_0 = \mathbf{0}$. For (initial and refined) token features, we always use a learned positional encoding $\rho^a \in \mathbb{R}^{M \times D}$ to denote the identifies of the tokens. For (initial and refined) frame features, we use a fixed absolute sinusoidal positional encoding $\rho^f \in \mathbb{R}^{T \times D}$ to denote frame positions, which is needed for cross-attention. For simplicity of notation, we drop them in equations below.

In the frame branch, to compute the updated frame features, $\mathbf{F}_1$, we use dilated convolution layers [37] to capture

temporal information among frames,

$$\mathbf{F}_1 = \text{convolution}(\mathbf{F}_0). \quad (1)$$

In the action branch, to compute updated features of action tokens, $\mathbf{A}_1$, and learn their dependencies, we use transformer with multi-head cross-attention and self-attention [57],

$$\mathbf{A}_1 = \text{transformer}(\mathbf{A}_0, \mathbf{F}_1), \quad (2)$$

by using updated frame features $\mathbf{F}_1$ and initial token features $\mathbf{A}_0$. The outputs of the input block are the updated features of frames, $\mathbf{F}_1$, and action tokens, $\mathbf{A}_1$.

Update Block. The purpose of an update block is to use cross-attention between branches so that frame features can leverage high-level action dependencies captured by action tokens, while action tokens can access low-level information of the frame features. The inputs to the first update block are $(\mathbf{A}_1, \mathbf{F}_1)$ and its outputs are the refined features $(\mathbf{A}_2, \mathbf{F}_2)$.

First, we use a *frame-to-action cross-attention* layer with *one attention head* to update action tokens based on input frame features and input token features, i.e., we use $\mathbf{A}_1$ as queries and $\mathbf{F}_1$ as keys and values. We then refine action tokens with several self-attention layers,

$$(\mathbf{A}'_2, \mathbf{\Lambda}^a_2) = \text{cross-attention}_{\text{f}\to\text{a}}(\mathbf{A}_1, \mathbf{F}_1), \quad (3)$$

$$\mathbf{A}_2 = \text{transformer}(\mathbf{A}'_2), \quad (4)$$

where $\mathbf{A}'_2$ denotes the output of the frame-to-action cross-attention and $\mathbf{A}_2$ is the refined action tokens. Here, $\mathbf{\Lambda}^a_2 \in \mathbb{R}^{T\times M}$ denotes the obtained one attention map where each column is the attention from a token to all the frames and will sum to one. We use one attention map to capture an alignment between action tokens and frames by obtaining the temporal locations of segments associated with each token.

Next, we use a single-head *action-to-frame cross-attention* to update frame features using updated action tokens $\mathbf{A}_2$ and input frame features $\mathbf{F}_1$, followed by refining the frame feature with convolutions,

$$(\mathbf{F}'_2, \mathbf{\Lambda}^f_2) = \text{cross-attention}_{\text{a}\to\text{f}}(\mathbf{F}_1, \mathbf{A}_2), \quad (5)$$

$$\mathbf{F}_2 = \text{convolution}(\mathbf{F}'_2). \quad (6)$$

Here, $\mathbf{\Lambda}^f_2 \in \mathbb{R}^{T\times M}$ is the attention from frames to action tokens (each row sums to one). Lastly, we pass the updated features $(\mathbf{A}_2, \mathbf{F}_2)$ to the next update block and repeat the process util we obtain $\mathbf{A}_B$ and $\mathbf{F}_B$ from the last update block, where B is the total number of blocks.

Remark 1 *When the number of video frames is very large, computing cross-attention is costly. We can optionally apply temporal downsampling and upsampling before and after cross-attention. Our experiments show that it does not affect the accuracy much (see the supplementary materials).*

Generating Predictions. We use the output $(\mathbf{A}_b, \mathbf{F}_b)$ of each block b to compute action predictions for the tokens and frames,

$$\mathbf{P}^a_b = \text{fully-connected}(\mathbf{A}_b) \in \mathbb{R}^{M\times(A+1)}, \quad (7)$$

$$\mathbf{P}^f_b = \text{fully-connected}(\mathbf{F}_b) \in \mathbb{R}^{T\times A}, \quad (8)$$

where $\mathbf{P}^a_b, \mathbf{P}^f_b$ are the predicted action labels of action tokens and frame features from the b-th block. As we discuss in the next subsection, we supervised the predictions of all blocks using the ground-truth for enhanced feature learning. Notice a token belongs to either the A real action classes, indicating the label of the segment it encodes, or the null class, indicating that the token does not encode any segment.

To obtain the final framewise labels, we fuse the predictions from the two branches in the last block, B. Notice that for the frame branch, the predicted frame labels are already given by $\mathbf{P}^f_B \in \mathbb{R}^{T\times A}$. For the action branch, however, we have predicted token labels $\mathbf{P}^a_B \in \mathbb{R}^{M\times(A+1)}$, which we need to turn into predicted frame labels. We do so by using the last attention map from cross-attention, $\mathbf{\Lambda}^f_B$, which indicates the alignment between action tokens and frames. More specifically, we first exclude the tokens of the null class from $\mathbf{P}^a_B, \mathbf{\Lambda}^f_B$ by removing the corresponding rows/columns. Hence, we obtain $\bar{\mathbf{P}}^a_B \in \mathbb{R}^{\bar{M}\times A}, \bar{\mathbf{\Lambda}}^f_B \in \mathbb{R}^{T\times\bar{M}}$, where $\bar{M}$ is the number of tokens that belong to any of the A real classes. We then compute the framewise predictions from the action branch as $\bar{\mathbf{\Lambda}}^f_B \cdot \bar{\mathbf{P}}^a_B \in \mathbb{R}^{T\times A}$. This means that we set the action label of a frame to the action label of the token representing the frame.

Finally, we fuse the framewise action predictions from the frame and action branches to form the final predictions,

$$\mathbf{P} = w \times (\bar{\mathbf{\Lambda}}^f_B \cdot \bar{\mathbf{P}}^a_B) + (1-w) \times \mathbf{P}^f_B \in \mathbb{R}^{T\times A}, \quad (9)$$

where w is a weighting hyperparameter, which we select by cross-validation on the training set.

Remark 2 *Our action branch operates on a fixed-number of action tokens, thus can emit predictions for all tokens simultaneously, while MuCon [56] and UVAST [7] need to generate them auto-regressively, which cannot leverage parallel computation. While UVAST attempted to align its predicted framewise labels and action sequences using cross-attention, it suffered from low accuracy, therefore, resorted to the costly Viterbi decoding for better alignment. On the other hand, our cross-attention learns accurate alignment between frame features and action tokens thanks to the iterative information transfer between branches.*

3.2. FACT's Loss Functions

To learn the parameters of our model, we apply supervision to the outputs of two branches at every block. We supervise

the frame branch using ground-truth framewise labels $\boldsymbol{y}$. For action branch, action tokens should encode the ground-truth action segments and learn their action labels. Therefore, we estimate the optimal matching between tokens and segments and use it to supervise the action branch (see the end of this subsection for details of the matching). Assume a training video has N ground-truth action segments, where segment n has action label a_n and temporal interval $\mathcal{T}_n$. Let $\boldsymbol{\pi}^* = [\pi_1^*, \pi_2^*, \dots, \pi_N^*]$ be the optimal segment-token matching, where $\pi_n^* \in \{1, \dots, M\}$ is the token index to which segment n is assigned. We define the following losses.

Frame Loss. We enforce that the framewise predictions of the frame branch in each block must conform with the ground-truth labels $\boldsymbol{y} = [y_1, \dots, y_T]$,

$$\mathcal{L}_{\text{frame}} = \sum_b \frac{1}{T} \sum_t -\log \mathbf{P}_b^f(t, y_t), \tag{10}$$

where $\mathbf{P}_b^f(t, y_t)$ is the prediction probability in the block b of frame t belonging to the class y_t.

Action Token Loss. We supervise action predictions for tokens of the action branch in all blocks using the segment-token matching $\boldsymbol{\pi}^*$. Let $\mathbf{P}_b^a(i, j)$ be the (i, j)-th entry of $\mathbf{P}_b^a$, denoting the probability that token i belongs to action class j. We define

$$\mathcal{L}_{\text{action}} = \sum_b \frac{1}{M} \Big[-\sum_n \log \mathbf{P}_b^a(\pi_n^*, a_n) - \sum_{m \in \mathcal{N}} \log \mathbf{P}_b^a(m, A+1) \Big], \tag{11}$$

where the first term requires that for each ground-truth segment n, its assigned token, π_n^*, should learn its action label, a_n. In the second term, $\mathcal{N} = \{m | m \notin \boldsymbol{\pi}^*\}$ denotes the set of indices of the tokens not matched to any segment, i.e., belonging to the null class. Thus, it enforces that such null tokens best represent the null class.

Cross-Attention Loss. Using cross-attention outputs, we enforce that tokens attend to the frames of their matched segments via $\boldsymbol{\Lambda}_b^a$, while frames in each segment attend to their matched token via $\boldsymbol{\Lambda}_b^f$,

$$\mathcal{L}_{\text{cross-att}} = \sum_{b>1} \frac{1}{T} \sum_n \sum_{t \in \mathcal{T}_n} -\Big(\log \boldsymbol{\Lambda}_b^a(t, \pi_n^*) + \log \boldsymbol{\Lambda}_b^f(t, \pi_n^*) \Big), \tag{12}$$

where the two terms are the cross-attention weights between a frame t in segment n and the matched token of segment n, i.e., π_n^*. We apply no constraint on the attentions of the tokens of null class. Notice $\mathcal{L}_{\text{cross-att}}$ is not applied to the input block, since there is no cross-attention module in it.

Temporal Smoothing Loss. It addresses the over-segmentation issue where frawewise action predictions fluctuate around action boundaries. We apply a smoothing loss on the action probabilities of frames and alignment (attention map) between frames and action tokens,

$$\mathcal{L}_{\text{smooth}} = \omega \sum_b \big(h(\mathbf{P}_b^f) + h(\boldsymbol{\Lambda}_b^a) + h(\boldsymbol{\Lambda}_b^f) \big), \tag{13}$$

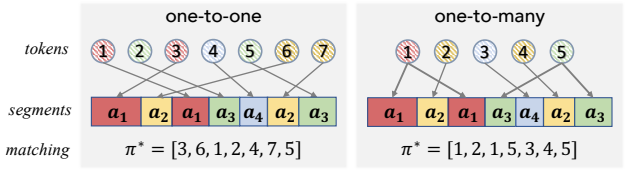


Figure 4. Visualization of one-to-one and one-to-many matching between action tokens and action segments.

where $h(\cdot)$ is a smoothing loss [16] (see the supplementary material for details) and ω is the weight, putting a trade-off with other losses. Finally, we minimize the overall loss function $\mathcal{L} = \mathcal{L}_{\text{frame}} + \mathcal{L}_{\text{action}} + \mathcal{L}_{\text{cross-att}} + \mathcal{L}_{\text{smooth}}$.

Optimal Segment-Token Matching. To obtain the labels for action tokens and cross-attentions, we compute the optimal matching between action tokens and ground-truth segments in a training video. By default, we match one token for each segment (*one-to-one*) to capture the fine-grained semantic of each segment. However, this can increase the computational cost on datasets with many segments in each video, e.g., EPIC-Kitchen. Thus, we also allow one token to match with multiple segments while ensuring the segments have the same action class (*one-to-many*), which reduces the number of required tokens and balances performance and efficiency. Figure 4 visualizes the two matching approaches. We use one-to-one matching on most datasets with one-to-many matching on EPIC-Kitchen, due to its much larger number of action segments per video than other datasets.

To compute the matching, we define a matching similarity $s(m, n)$ between a token m and a ground-truth segment n that belongs to action a_n and has temporal interval $\mathcal{T}_n$,

$$s(m, n) = \mathbf{P}_B^a(m, a_n) + \beta \frac{\sum_{t \in \mathcal{T}_n} \boldsymbol{\Lambda}_B^f(t, m)}{|\mathcal{T}_n| + \sum_{t \notin \mathcal{T}_n} \boldsymbol{\Lambda}_B^f(t, m)}. \tag{14}$$

Here, the first term, $\mathbf{P}_B^a(m, a_n)$ is the probability that token m belongs to the ground-truth action class a_n of segment n. The second term is a soft IoU score between the temporal intervals of segment n and token m, where $\boldsymbol{\Lambda}_B^f(t, m) \in [0, 1]$ is the probability that frame t belongs to token m. Lastly, β controls the balance between these two terms.

The optimal matching $\boldsymbol{\pi}^*$ should maximize the total matching similarity between tokens and ground-truth segments, $\boldsymbol{\pi}^* = \text{argmax}_{\boldsymbol{\pi}} \sum_n s(\boldsymbol{\pi}_n, n)$, subject to either one-to-one matching constraint, which we solve using Hungarian algorithm [28], or one-to-many matching constraint, which we propose a three-step algorithm to solve it. We discuss this algorithm in supplementary materials due to lack of space.

3.3. Leveraging Video Transcripts

When video transcript (the sequence of actions in a video) is available during testing, e.g., from meta data or when a user executes a procedural task according to instructions or manuals, FACT can efficiently leverage it to construct

	Note	Breakfast					GTEA				
		F1@{10, 25, 50}			Edit	Acc	F1@{10, 25, 50}			Edit	Acc
ED-TCN [30]	F_1	-	-	-	-	-	72.2	69.3	56.0	64.0	-
TDRN [32]	F_1	-	-	-	-	-	79.2	74.4	62.7	74.1	70.1
SSA-GAN [21]	F_2	-	-	-	-	43.3	80.6	79.1	74.2	76.0	74.4
Bridge-Prompt [36]	F_3	-	-	-	-	-	94.1	92.0	83.0	91.6	81.2
MSTCN [16]		52.6	48.1	37.9	61.7	63.3	87.5	85.4	74.6	81.4	79.2
MSTCN++ [37]		64.1	58.6	45.9	64.9	67.6	88.8	85.7	76.0	83.5	80.1
MuCon [56]	P	73.2	66.1	48.4	76.3	62.8	-	-	-	-	-
C2F-TCN [55]	A	72.2	68.7	57.6	69.6	76.0	84.3	81.8	72.6	76.4	84.9
ASRF[24]	C_1	74.3	68.9	56.1	72.4	67.6	89.4	87.8	79.8	83.7	77.3
HASR [1]		74.7	69.5	57.0	71.9	69.4	90.9	88.6	76.4	87.5	77.4
ASFormer[61]		76.0	70.6	57.4	75.0	73.5	90.1	88.8	79.2	84.6	79.7
DTL[59]	C_2	78.8	74.5	62.9	77.7	75.8	-	-	-	-	-
MVGA[4]	A	75.6	72.1	59.7	76.8	74.2	91.3	90.0	79.3	86.4	80.3
TCTr [3]		76.6	71.1	58.5	76.1	77.5	91.3	90.1	80.0	87.9	81.1
UVAST[7]	P	76.9	71.5	58.0	77.1	69.7	92.7	91.3	81.0	92.1	80.2
RTK [25]		76.9	72.4	60.5	76.1	73.3	91.2	90.6	83.4	87.9	80.3
LTContext [5]		77.6	72.6	60.1	77.0	74.2	-	-	-	-	-
DiffAct[39]	C_1	80.3	75.9	64.6	78.4	75.1	92.5	91.5	**84.7**	89.6	82.2
FACT		**81.4**	**76.5**	**66.2**	**79.7**	**76.2**	**93.5**	**92.1**	84.1	**91.4**	**86.1**

Table 1. **Action Segmentation Results on Breakfast, GTEA**. P: additional post-processing; C: additional constraints (C_1: boundary detection; C_2: action ordering); A: data augmentation; F: different features (F_1: Spatio-temporal CNN, F_2: Generative Adversarial Network, F_3: ActionCLIP).

action tokens for better training and testing. Notice temporal boundaries of actions are still unknown. Specifically, 1) we construct the action tokens as the embeddings of actions in the transcript, where the action embeddings are learned during training. The transcripts of training videos can be parsed from their ground-truth framewise labels $\boldsymbol{y}$. 2) creating the tokens based on the transcript means we know the ground-truth matchings between the tokens and segments, thus we can use it in our losses to replace the optimal matching $\boldsymbol{\pi}^*$. With these two simple changes, FACT incorporates textual transcripts while maintaining the same inference speed. This is in contrast to prior works [1, 5, 16, 37, 39, 55, 61] that do not directly use transcripts and can only be extended via computationally expensive and time-consuming post-processing.

4. Experiments

We evaluate our method for action segmentation on four challenging datasets and compare with prior methods [7, 16, 24, 37, 61], specifically with ASFormer [61], UVAST [7], LTContext [5] and DiffAct [39], which are the best frame-based and two-stage methods.

4.1. Experimental Setup

Datasets. We evaluate on four datasets, representing different test scenarios: **Breakfast** [26] is a *popular benchmark dataset* and contains 1716 videos from 10 cooking recipes and 48 actions with an average of 6.9 action segments per video. **GTEA** [17] is a small-scale dataset with 28 videos to test *learning with limited data*. It has 11 actions and on average 33 segments per video. **EgoProceL** [6] is an egocentric dataset featuring *diverse tasks*, such as repairing cars, assembling toys and cooking. It has 1055 videos, 130 actions and on average 21 segments per video. **EPIC-Kitchen** [11] is the most challenging dataset featuring *long complex videos*. It has 633 videos, 3796 actions and on average 195 segments per video, while the longest video contains 1436 segments.

	EgoProceL						EPIC-Kitchen					
	F1@{10, 25, 50}			Edit	Acc	AccB	F1@{10, 25, 50}			Edit	Acc	AccB
MSTCN++[37]	60.3	57.0	46.5	62.4	69.3	82.5	15.2	13.6	9.5	11.6	18.2	27.0
ASFormer[61]	63.3	60.9	51.0	64.9	71.1	84.9	16.4	14.8	10.5	12.6	19.1	26.9
UVAST[7]	60.5	58.3	46.6	67.7	67.8	83.2	10.8	8.0	4.3	4.8	25.2	15.8
LTContext[5]	64.2	61.3	51.2	61.3	70.3	84.7	19.5	17.1	11.5	16.7	20.2	31.9
DiffAct[40]	67.5	65.4	54.6	68.4	77.0	86.6	12.5	10.9	7.2	9.6	17.1	30.9
FACT	**73.0**	**69.8**	**60.8**	**75.7**	**77.6**	**88.0**	**27.6**	**25.3**	**18.5**	**22.3**	**20.8**	**34.7**

Table 2. **Action Segmentation Results on Long Complex Videos (EgoProceL and EPIC-Kitchen)**.

Metrics. Following prior works [7, 16, 37, 55, 61], we compute segmental Edit distance score (*Edit*), segmental F1 score (*F1*) at three overlapping thresholds 10%, 25%, 50%, denoted by F1@{10, 25, 50} and framewise accuracy (*Acc*). While EgoProceL and EPIC-Kitchen contain many background frames, we follow the conventional approach to exclude background frames in evaluation. To have comprehensive results, we also compute framewise accuracy with background frames on the two datasets, denoted by *AccB*.

Implementation. Our FACT model has 1 input block and 3 update blocks. The convolution in each block includes 9 layers of dilated convolution from [37]. The transformer contains 6 self- and cross-attention layers in input block and 1 self-attention layer in update block. We set $\beta = 0.2$ in Eq(14) and learn one-to-one matching between action tokens and segments on Breakfast, GTEA and EgoProceL with 60, 60 and 300 tokens, respectively, and one-to-many matching on EPIC-Kitchen with 300 tokens, since learning one-to-one matching requires at least 1500 tokens, which would be less efficient. On EgoProceL and EPIC-Kitchen, we reproduce the best prior works, DiffAct [39], UVAST [7] and other methods [5, 37, 61] using their released codes. We cannot replicate [3, 25] as their codes are not released. We include more implementation details in the supplementary materials.

4.2. Comparison with the State-of-the-Art

Action Segmentation. We report the action segmentation performance in Table 1 and 2. Notice that *FACT achieves new state-of-the-art results on all datasets*, despite many prior works leveraging e.g., post-processing, data augmentation and extra training constraints, which also increase the complexity of their methods. The top section of Table 1 uses different frame features thus is not comparable to us. We exceed the best prior works on F1@50 by *1.9%*, *6.2%* and *7.0%* on Breakfast, EgoProceL and EPIC-Kitchen, respectively. On GTEA, we obtain the best overall performance, especially with *3.0%* higher Edit score than DiffAct. UVAST and DiffAct cannot converge well on EPIC-Kitchen, due to

	Breakfast					GTEA					EgoProceL						Epic-Kitchen					
	F1@{10,25,50}			Edit	Acc	F1@{10,25,50}			Edit	Acc	F1@{10,25,50}			Edit	Acc	AccB	F1@{10,25,50}			Edit	Acc	AccB
MSTCN++[37]	84.0	75.7	59.4	-	72.2	94.3	90.5	78.9	-	78.3	71.4	66.9	53.2	-	73.9	84.8	55.0	42.5	22.9	-	28.1	44.6
ASFormer[61]	85.4	78.5	63.9	-	74.6	95.8	94.1	82.9	-	82.0	71.3	66.8	53.2	-	75.7	85.8	58.2	45.5	24.6	-	32.1	46.9
UVAST[7]	87.6	81.9	69.2	-	77.0	**96.9**	95.3	86.0	-	82.7	68.5	63.6	50.6	-	75.8	85.9	32.2	25.3	12.9	-	37.7	27.7
LTContext[5]	87.8	81.1	66.2	-	77.2	-	-	-	-	-	73.6	68.7	53.0	-	76.5	85.8	56.8	44.9	24.7	-	32.2	47.2
DiffAct[40]	87.8	81.8	67.0	-	76.2	94.9	92.6	83.7	-	84.7	74.0	70.6	55.8	-	79.8	86.7	42.5	33.0	17.4	-	31.0	42.5
FACT	**89.9**	**85.6**	**73.7**	**93.5**	**84.5**	96.1	**95.6**	**87.5**	**96.3**	**84.0**	**74.3**	**71.2**	**61.9**	**87.2**	**80.0**	**87.4**	**66.9**	**60.8**	**44.8**	**75.2**	**58.2**	**58.3**

Table 3. **Action Segmentation Performance with Transcripts Knowledge.**

	MSTCN++	ASFormer	UVAST	LTContext	DiffAct	FACT
Parameters (10^6)	1.0	1.1	1.1	0.7	1.7	1.9
Inference GFLOP	4.5	7.6	2.9 †	7.8	63.0	5.5

Table 4. **Comparison of Model Parameters and Inference GFLOP**. Models are evaluated with the same hidden dimension on a 5-min video. †: Complexity of UVAST is not fully reflected in GFLOPS as its requires costly post-processing that takes long computation time.

the longer video length and more action segments per video. However, our method captures long temporal relations with action tokens and learns one-to-many token-segment matching to largely reduce the number of tokens required. Thus it lowers the computation cost and learning difficulty, as validated by the improved F1.

Moreover, we show in Table 4 that FACT has *a lower inference complexity than pure-transformer methods* [5, 7, 39, 61]. While FACT has slightly more parameters due to having two branches, each branch is computation-efficient. We have *30%* and *91%* less GFLOP than the best competitors, LTContext and DiffAct, respectively. Notice UVAST requires costly postprocessing (Viterbi Decoding) that takes much longer computation time, which is not captured by its inference GFLOP. We show this in Figure 2, where FACT is *30* and *3.5* times faster than UVAST and DiffAct, respectively. MSTCN++ is a pure-convolution method, thus has smaller GFLOP but also lower performance.

Action Segmentation with Transcripts. When video transcripts are available during training and testing, action segmentation can be considered as learning alignments between frames and the action segments given by the transcripts. Thus, we extend prior works with Viterbi decoding [19] to obtain the optimal alignments between their framewise predictions and the transcripts, which has shown reliable performance in many prior works [7, 56]. Since it ensures their predictions obey the transcripts, the Edit score will always be 100% and cannot reflect the model performance. Notice our setting is different from weakly-supervised action segmentation [34, 41, 48], which learns from video transcripts.

As Table 3 shows, despite not using Viterbi decoding, FACT still achieves the best performance, improving F1@50 by *4.5%, 1.5%, 6.1%, 20.1%* on Breakfast, GTEA, EgoProceL and EPIC-Kitchen, respectively. It is because *FACT can use transcripts to initialize action tokens and improve feature learning* while prior works can only apply Viterbi decoding as a post-processing step. The high Edit scores also indicate the predictions of FACT closely follow the transcripts, thanks to the cross-attentions for aligning action tokens and frame features. We observe Viterbi decoding fails to correct action localization errors in a video if the predictions contain many false positive. Thus, it shows less improvement on some methods, e.g., DiffAct, while UVAST benefits more from it as its overall prediction confidence is lower.

4.3. Ablation Studies

We exam the key design of FACT, including the effect of action tokens, loss functions and more. We provide more ablation studies in the supplementary materials.

Performance of Long Temporal Modeling. We first validate FACT's ability for better long temporal modeling. In Figure 5, we split Breakfast videos into four groups based on their lengths and evaluate ASFormer, UVAST, DiffAct and FACT on each group. While model performance is close on short videos (0-1k frames), *the performance of FACT is more robust on longer videos.* As video length increases, FACT outperforms DiffAct by bigger margins while the results of UVAST degrades quickly, which explains its lower performance on EgoProceL and EPIC-Kitchen.

Effect of Action Tokens. In Figure 6, we test the effect of the number of action tokens under different segment-token matching approaches. We report on EgoProceL, since it has 130 action classes, allowing learning both *one-to-many* (OTM) or *one-to-one* (OTO) matching. First, *no-token* (red) is a baseline model with only frame branch, thus it cannot well capture the long temporal dependency, leading to a very low F1. On the other hand, using action tokens with OTM (green) or OTO (blue) matching improves F1 by 4-6%, showing action-level temporal modeling is key to good action segmentation. The accuracies of both OTO and OTM are not sensitive to the number of tokens.

Effect of Segment-Token Matching. In Figure 6, we also compare the effect of different matchings between action tokens and segments. First, OTO matching shows better F1 than OTM as it separately encodes each segment thus better learns their action classes and locations, yet the performance gap between the two is not large.

We further compare OTM with a *one-per-class* (OPC) matching, where we assign a fixed token to encode the segments of a specific action class, and compare OTO with *seq-*

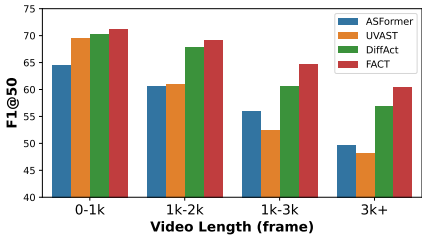


Figure 5. Performance by Video Length.

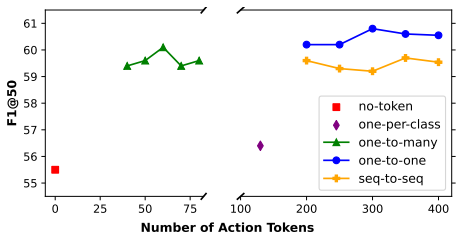


Figure 6. Effect of Action Tokens.

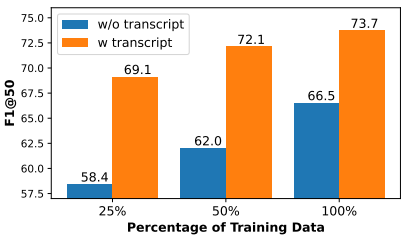


Figure 7. Effect of Leveraging Transcripts.

to-seq (STS), where we consider the tokens as a sequence, matching the first N tokens to the N segments (recall that N is the number of ground-truth segments) and the rest to the null class. For OPC (purple), notice the number of its required tokens equal to the number of *all* action classes and is more than that of OTM, which is linear to the number of action classes *within one video*. OPC also has a lower F1 as it always fuses the information of segments of the same action, while OTM allows the model to decide when to encode segments with one token. For STS (yellow), it obtains lower F1 than OTO because it imposes sequential dependency between tokens, thus mistakes in one token (e.g., token 5 should predict the 5-th segment but incorrectly predicts the 6-th segment) affect predictions of all subsequent tokens. In OTO, such error in a token will not affect others.

Benefit of Leveraging Transcripts. One important motivation for incorporating transcript is to leverage the information from textual modality to more effectively learn a model with less data and annotations. To valid this, we test learning with 25%, 50% and 100% training data on Breakfast. As Figure 7 shows, with just 25% training data, our model with transcripts outperforms the one not using transcripts but with 100% training data. *This shows our model can be applied to scenarios with a small number of training videos or where annotation is sparse*. For example, when videos are collected with textual transcripts while framewise labels being unknown [9, 12, 35, 41, 48, 56], it is possible to annotate the labels of a small portion of videos to learn our model and apply it to obtain pseudo-labels for other videos then iteratively refine the model.

A	C	F	S	F1@{10,25,50}			Edit	Acc
		✓	✓	54.0	49.7	40.0	60.6	66.0
✓		✓	✓	56.3	52.7	43.2	61.4	70.9
✓	✓		✓	73.4	68.2	57.3	73.3	70.9
✓	✓	✓		73.2	69.3	60.7	72.7	73.7
✓	✓	✓	✓	**79.1**	**75.5**	**65.5**	**78.3**	**74.1**

Table 5. **Effect of Proposed Loss Functions**. *A, C, F, S* stand for $\mathcal{L}_{\text{action}}$, $\mathcal{L}_{\text{cross-att}}$, $\mathcal{L}_{\text{frame}}$, $\mathcal{L}_{\text{smooth}}$ respectively.

Effect of Losses. In Table 5, we show the effect of our proposed losses on Split 1 of Breakfast. When we remove supervision for action branch (first and second rows), it harms the model's ability to capture long temporal relations, leading to drop in all metrics. In comparison, removing supervision for frame branch (third row) causes smaller performance drop, showing the importance of long-range temporal modeling. Removing $\mathcal{L}_{\text{smooth}}$ (forth row) leads to over-segmentation, where the model predicts many short incorrect segments, thus decreases F1 and Edit.

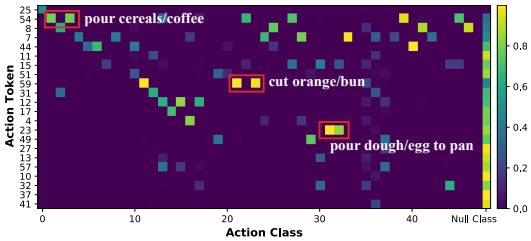


Figure 8. Matching between Action Token and Action Class.

Qualitative Results. We study how our model assigns action tokens to encode segments of different action classes. In Figure 8, for a subset of tokens on Breakfast, we show the frequency that the segments of one class is associated to a certain token (tokens are reordered for better visualization). Notice that, although we learned 60 tokens, more than the number of action classes, FACT does not simply assign one token per class but shares the token for similar classes. For example, token 54 is often related to *pouring* actions; token 59 to *cut* and token 23 to *pouring things to pan*, showing **tokens have learned the semantic of the actions**.

5. Conclusions

We proposed FACT that performs temporal modeling on frame and action levels in parallel and performs iterative bidirectional information transfer between them for frame and action feature refinement. FACT has a frame branch to learn frame features with convolution, an action branch to learn action tokens with transformer, and cross-attentions for cross-branch communication. By extensive experiments, we showed FACT exceeds all prior methods on four datasets with a low inference complexity, and is able to also incorporate textual transcripts when they are available.

Acknowledgements

This work is sponsored by DARPA PTG (HR00112220001), NSF (IIS-2115110), ARO (W911NF2110276). Content does not necessarily reflect the position/policy of the Government.

References

[1] Hyemin Ahn and Dongheui Lee. Refining action segmentation with hierarchical video representations. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 16302–16310, 2021. 1, 2, 3, 6

[2] J. B. Alayrac, P. Bojanowski, N. Agrawal, J. Sivic, I. Laptev, and S. Lacoste-Julien. Unsupervised learning from narrated instruction videos. *IEEE Conference on Computer Vision and Pattern Recognition*, 2016. 2

[3] Nicolas Aziere and Sinisa Todorovic. Multistage temporal convolution transformer for action segmentation. *Image and Vision Computing*, 128:104567, 2022. 1, 2, 6

[4] Nicolas Aziere and Sinisa Todorovic. Markov game video augmentation for action segmentation. In *International Conference on Computer Vision (ICCV)*, 2023. 2, 6

[5] Emad Bahrami, Gianpiero Francesca, and Juergen Gall. How much temporal long-term context is needed for action segmentation? In *International Conference on Computer Vision (ICCV)*, 2023. 1, 2, 6, 7

[6] Siddhant Bansal, Chetan Arora, and C.V. Jawahar. My view is the best view: Procedure learning from egocentric videos. In *European Conference on Computer Vision (ECCV)*, 2022. 2, 6

[7] Nadine Behrmann, S. Alireza Golestaneh, Zico Kolter, Juergen Gall, and Mehdi Noroozi. Unified fully and timestamp supervised temporal action segmentation via sequence to sequence translation. In *ECCV*, 2022. 1, 2, 3, 4, 6, 7

[8] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. *CoRR*, abs/2005.12872, 2020. 3

[9] Chien-Yi Chang, De-An Huang, Yanan Sui, Li Fei-Fei, and Juan Carlos Niebles. D3tw: Discriminative differentiable dynamic time warping for weakly supervised action alignment and segmentation. *IEEE Conference on Computer Vision and Pattern Recognition*, 2019. 2, 8

[10] Xiyang Dai, Yinpeng Chen, Jianwei Yang, Pengchuan Zhang, Lu Yuan, and Lei Zhang. Dynamic detr: End-to-end object detection with dynamic attention. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 2988–2997, 2021. 3

[11] Dima Damen, Hazel Doughty, Giovanni Maria Farinella, Antonino Furnari, Jian Ma, Evangelos Kazakos, Davide Moltisanti, Jonathan Munro, Toby Perrett, Will Price, and Michael Wray. Rescaling egocentric vision: Collection, pipeline and challenges for epic-kitchens-100. *International Journal of Computer Vision (IJCV)*, 130:33–55, 2022. 2, 3, 6

[12] Li Ding and Chenliang Xu. Weakly-supervised action segmentation with iterative soft boundary assignment. *IEEE Conference on Computer Vision and Pattern Recognition*, 2018. 2, 8

[13] G. Donahue and E. Elhamifar. Learning to predict task progress by self-supervised video alignment. *IEEE Conference on Computer Vision and Pattern Recognition*, 2024. 1

[14] E. Elhamifar and D. Huynh. Self-supervised multi-task procedure learning from instructional videos. *European Conference on Computer Vision*, 2020. 2

[15] E. Elhamifar and Z. Naing. Unsupervised procedure learning via joint dynamic summarization. *International Conference on Computer Vision*, 2019. 2

[16] 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, 5, 6

[17] Alireza Fathi, Xiaofeng Ren, and James M Rehg. Learning to recognize objects in egocentric activities. *In IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021. 2, 6

[18] Mohsen Fayyaz and Jurgen Gall. Sct: Set constrained temporal transformer for set supervised action segmentation. *IEEE Conference on Computer Vision and Pattern Recognition*, 2020. 2

[19] G.D. Forney. The viterbi algorithm. *Proceedings of the IEEE*, 1973. 7

[20] Daniel Fried, Jean-Baptiste Alayrac, Phil Blunsom, Chris Dyer, Stephen Clark, and Aida Nematzadeh. Learning to segment actions from observation and narration. *Annual Meeting of the Association for Computational Linguistics*, 2020. 2

[21] Harshala Gammulle, Simon Denman, Sridha Sridharan, and Clinton Fookes. Fine-grained action segmentation using the semi-supervised action gan. *Pattern Recognition*, 2020. 6

[22] 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)*, 2020. 1, 2

[23] Noureldien Hussein, Efstratios Gavves, and Arnold WM Smeulders. Videograph: Recognizing minutes-long human activities in videos. In *ICCV Workshop on Scene Graph Representation and Learning*, 2019. 2

[24] Yuchi Ishikawa, Seito Kasai, Yoshimitsu Aoki, and Hirokatsu Kataoka. Alleviating over-segmentation errors by detecting action boundaries. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 2322–2331, 2021. 1, 6

[25] 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 (ICCV)*, 2023. 1, 2, 3, 6

[26] H. Kuehne, A. Arslan, and T. Serre. The language of actions: Recovering the syntax and semantics of goal-directed human. *IEEE Conference on Computer Vision and Pattern Recognition*, 2014. 2, 6

[27] H. Kuehne, J. Gall, and T. Serre. An end-to-end generative framework for video segmentation and recognition. *IEEE Winter Conference on Applications of Computer Vision*, 2016. 2

[28] Harold W. Kuhn. *The Hungarian Method for the Assignment Problem*. 1955. 5

[29] Anna Kukleva, Hilde Kuehne, Fadime Sener, and Jurgen Gall. Unsupervised learning of action classes with continuous temporal embedding. *IEEE Conference on Computer Vision and Pattern Recognition*, 2019. 2

[30] C. Lea, M. D. Flynn, R. Vidal, A. Reiter, and G. D. Hager. Temporal convolutional networks for action segmentation and detection. *IEEE Conference on Computer Vision and Pattern Recognition*, 2017. 2, 6

[31] S. Lee, Z. Lu, Z. Zhang, M. Hoai, and E. Elhamifar. Error detection in egocentric procedural task videos. *IEEE Conference on Computer Vision and Pattern Recognition*, 2024. 1, 2

[32] Peng Lei and Sinisa Todorovic. Temporal deformable residual networks for action segmentation in videos. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018. 6

[33] Jun Li and Sinisa Todorovic. Set-constrained viterbi for set-supervised action segmentation. *IEEE Conference on Computer Vision and Pattern Recognition*, 2020. 2

[34] J. Li and S. Todorovic. Anchor-constrained viterbi for set-supervised action segmentation. *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021. 7

[35] J. Li, P. Lei, and S. Todorovic. Weakly supervised energy-based learning for action segmentation. *International Conference on Computer Vision*, 2019. 2, 8

[36] M. Li, L. Chen, Y. Duarr, Z. Hu, J. Feng, J. Zhou, and J. Lu. Bridge-prompt: Towards ordinal action understanding in instructional videos. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. 2, 6

[37] Shi-Jie Li, Yazan AbuFarha, Yun Liu, Ming-Ming Cheng, and Juergen Gall. Ms-tcn++: Multi-stage temporal convolutional network for action segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1–1, 2020. 1, 2, 3, 6, 7

[38] Zhe Li, Yazan Abu Farha, and Jurgen Gall. Temporal action segmentation from timestamp supervision. *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021. 2

[39] Daochang Liu, Qiyue Li, Anh-Dung Dinh, Tingting Jiang, Mubarak Shah, and Chang Xu. Diffusion action segmentation. In *International Conference on Computer Vision (ICCV)*, 2023. 6, 7

[40] Yang Liu, Jiayu Huo, Jingjing Peng, Rachel Sparks, Prokar Dasgupta, Alejandro Granados, and Sebastien Ourselin. Skit: a fast key information video transformer for online surgical phase recognition. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 21074–21084, 2023. 1, 2, 6, 7

[41] Z. Lu and E. Elhamifar. Weakly-supervised action segmentation and alignment via transcript-aware union-of-subspaces learning. *International Conference on Computer Vision*, 2021. 2, 7, 8

[42] Z. Lu and E. Elhamifar. Set-supervised action learning in procedural task videos via pairwise order consistency. *IEEE Conference on Computer Vision and Pattern Recognition*, 2022. 2

[43] Tim Meinhardt, Alexander Kirillov, Laura Leal-Taixe, and Christoph Feichtenhofer. Trackformer: Multi-object tracking with transformers. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. 3

[44] Depu Meng, Xiaokang Chen, Zejia Fan, Gang Zeng, Houqiang Li, Yuhui Yuan, Lei Sun, and Jingdong Wang. Conditional detr for fast training convergence. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 3651–3660, 2021. 3

[45] Megha Nawhal and Greg Mori. Activity graph transformer for temporal action localization, 2021. 2

[46] Rahul Rahaman, Dipika Singhania, Alexandre Thiery, and Angela Yao. A generalized and robust framework for timestamp supervision in temporal action segmentation. In *Computer Vision–ECCV 2022: 17th European Conference*, 2022. 2

[47] A. Richard, H. Kuehne, and J. Gall. Action sets: Weakly supervised action segmentation without ordering constraints. *IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

[48] A. Richard, H. Kuehne, A. Iqbal, and J. Gall. Neuralnetwork-viterbi: A framework for weakly supervised video learning. *IEEE Conference on Computer Vision and Pattern Recognition*, 2018. 2, 7, 8

[49] M. Rohrbach, S. Amin, M. Andriluka, and B. Schiele. A database for fine grained activity detection of cooking activities. *IEEE Conference on Computer Vision and Pattern Recognition*, 2012. 2

[50] Y. Shen and E. Elhamifar. Semi-weakly-supervised learning of complex actions from instructional task videos. *IEEE Conference on Computer Vision and Pattern Recognition*, 2022. 2

[51] Y. Shen and E. Elhamifar. Progress-aware online action segmentation for egocentric procedural task videos. *IEEE Conference on Computer Vision and Pattern Recognition*, 2024. 1, 2

[52] Y. Shen, L. Wang, and E. Elhamifar. Learning to segment actions from visual and language instructions via differentiable weak sequence alignment. *IEEE Conference on Computer Vision and Pattern Recognition*, 2021. 2

[53] G. A. Sigurdsson, S. Divvala, A. Farhadi, and A. Gupta. Asynchronous temporal fields for action recognition. *IEEE Conference on Computer Vision and Pattern Recognition*, 2017. 2

[54] B. Singh, T. K. Marks, M. Jones, O. Tuzel, and M. Shao. A multi-stream bi-directional recurrent neural network for finegrained action detection. *IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[55] Dipika Singhania, Rahul Rahaman, and Angela Yao. Coarse to fine multi-resolution temporal convolutional network. *CoRR*, abs/2105.10859, 2021. 1, 2, 6

[56] Yaser Souri, Mohsen Fayyaz, Luca Minciullo, Gianpiero Francesca, and Juergen Gall. Fast Weakly Supervised Action Segmentation Using Mutual Consistency. *PAMI*, 2021. 1, 2, 3, 4, 6, 7, 8

[57] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. *Neural Information Processing Systems*, 2017. 4

[58] Mengmeng Wang, Jiazheng Xing, and Yong Liu. Actionclip: A new paradigm for video action recognition. *CoRR*, 2021. 2

[59] 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 *Advances in Neural Information Processing Systems*, 2022. 1, 2, 6

[60] S. Yeung, O. Russakovsky, N. Jin, M. Andriluka, G. Mori, and L. Fei-Fei. Every moment counts: Dense detailed labeling of actions in complex videos. *International Journal of Computer Vision*, 2018. 2

[61] Fangqiu Yi, Hongyu Wen, and Tingting Jiang. Asformer: Transformer for action segmentation. In *The British Machine Vision Conference (BMVC)*, 2021. 1, 2, 6, 7

[62] Junbin Zhang, Pei-Hsuan Tsai, and Meng-Hsun Tsai. Semantic2graph: Graph-based multi-modal feature fusion for action segmentation in videos, 2022. 1, 2

[63] Xizhou Zhu, Weijie Su, Lewei Lu, Bin Li, Xiaogang Wang, and Jifeng Dai. Deformable detr: Deformable transformers for end-to-end object detection. *arXiv preprint arXiv:2010.04159*, 2020. 3

[64] D. Zhukov, J. B. Alayrac, R. G. Cinbis, D. Fouhey, I. Laptev, and J. Sivic. Cross-task weakly supervised learning from instructional videos. *IEEE Conference on Computer Vision and Pattern Recognition*, 2019. 2