Alleviating Over-segmentation Errors by Detecting Action Boundaries

Yuchi Ishikawa[1,2], Seito Kasai[1,2], Yoshimitsu Aoki[2], Hirokatsu Kataoka[1]

[1]National Institute of Advanced Industrial Science and Technology (AIST)
[2]Keio University

{yuchi.ishikawa, seito-kasai, hirokatsu.kataoka}@aist.go.jp, aoki@elec.keio.ac.jp

Abstract

*We propose an effective framework for the temporal action segmentation task, namely an **Action Segment Refinement Framework (ASRF)**. Our model architecture consists of a long-term feature extractor and two branches: the Action Segmentation Branch (ASB) and the Boundary Regression Branch (BRB). The long-term feature extractor provides shared features for the two branches with a wide temporal receptive field. The ASB classifies video frames with action classes, while the BRB regresses the action boundary probabilities. The action boundaries predicted by the BRB refine the output from the ASB, which results in a significant performance improvement. Our contributions are three-fold: (i) We propose a framework for temporal action segmentation, the ASRF, which divides temporal action segmentation into frame-wise action classification and action boundary regression. Our framework refines frame-level hypotheses of action classes using predicted action boundaries. (ii) We propose a loss function for smoothing the transition of action probabilities, and analyze combinations of various loss functions for temporal action segmentation. (iii) Our framework outperforms state-of-the-art methods on three challenging datasets, offering an improvement of up to 13.7% in terms of segmental edit distance and up to 16.1% in terms of segmental F1 score. Our code is publicly available*[1].

1. Introduction

In recent years, with the exponential increase in the number of videos uploaded on the Internet, more attention is now being paid to video analysis. One of the most active topics in video analysis is video classification, the goal of which is to classify a trimmed video into a single action label [1]. The rise of sophisticated architectures (e.g. two-stream convnets [38, 13], 3D CNNs [43, 2, 16, 47, 12, 11], (2+1)D CNN [45]) and large-scale video datasets such as Kinetics [21, 3] and Sports-1M [20] has greatly improved the performance of video classification.

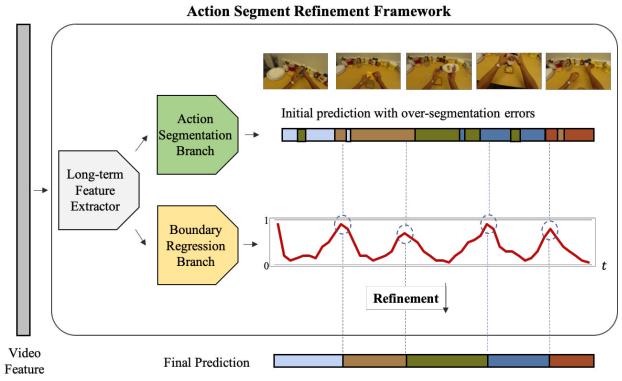

Figure 1. Overview of proposed framework. The Action Segment Refinement Framework (ASRF) consists of a long-term feature extractor(Section 3.1), an Action Segmentation Branch (ASB; Section 3.2) and a Boundary Regression Branch (BRB; Section 3.3). Video features are input for a long-term feature extractor, and it provides shared features for the ASB and the BRB. Then these two branches output frame-wise action class predictions and boundary probabilities, respectively. The ASRF reassigns frame-level action predictions from the ASB utilizing the action boundaries predicted by the BRB.

However, natural videos are untrimmed and may contain multiple action instances. Therefore, analysis of these videos demands the recognition of action sequences. Motivated by this, we address the task of temporal action segmentation, which aims to capture and classify each action segment of an untrimmed video into an action category. Action segmentation has various potential applications in robotics, surveillance and the analysis of human activities.

Extant research typically approaches this task through two phases: i) extracting spatial or spatiotemporal features with 2D CNNs, two-stream convnets [38, 13] or 3D CNNs [2] and ii) temporally classifying the extracted features using an RNN [17] or temporal convolutional net-

[1]https://github.com/yiskw713/asrf

works [25, 28, 9]. These models claim better results on datasets with a small number of action classes and video clips than earlier approaches [37, 19]. However, it is difficult for these methods to recognize action segments, especially on large datasets with diverse action classes, which results in continuously fluctuating action class predictions, or *over-segmentation errors*.

When considering temporal action segmentation, over-segmentation is a critical issue. The prediction in the upper part of Figure 1 shows an example prediction with over-segmentation errors. This error is crucial when analyzing untrimmed videos with a sequence of actions. For example, when analyzing the steps that are taken in cooking videos, over-segmentation will result in detecting extra steps. Assuming that actions in videos exist as chunks and do not rapidly change, reducing these errors is essential for improving action segmentation performance.

To address this problem, we propose a framework for action segmentation that leverages predicted action boundaries for action segmentation, namely Action Segment Refinement Framework (ASRF). The ASRF consists of a long-term feature extractor and two branches, an Action Segmentation Branch (ASB) and a Boundary Regression Branch (BRB) as in Figure 1. The long-term feature extractor expands the temporal receptive field and provides shared features for the following two branches. Then, the ASB broadly predicts frame-wise actions in a video, while the BRB detects action boundaries regardless of action class. Our framework refines the outputs from the ASB using action boundaries predicted by the BRB. The ASB and the BRB are complementary to each other, which enables the reduction of over-segmentation errors. In summary, the main contributions of this work are highlighted as follows:

1. We propose a simple but effective framework for action segmentation, an ASRF. Our framework decouples frame-wise action classification and action boundary regression, which enables the capturing of reliable segments and correct classification. Having a decoupled architecture enhances single model classification results by up to 10.6% in terms of segmental F1 score and 10.8% in terms of segmental edit distance.
2. We propose a loss function for smoothing the transition of action probabilities and investigate appropriate combinations of loss functions for our framework. Combining this loss function with a class-weighted classification loss function enables a 8.2% improvement of segmental F1 score and a 5.9% improvement of segmental edit distance.
3. Our approach outperforms state-of-the-art methods on three challenging datasets for action segmentation: 50 Salads [41], Georgia Tech Egocentric Activities (GTEA) [10] and the Breakfast dataset [22], by up to 16.1% improvement of segmental F1 score, 13.7% improvement of segmental edit distance and 2.6% improvement for frame-wise accuracy.

2. Related Work

Video Representation. Action segmentation aims to classify each frame in a video into action classes, assuming that each frame contains a single action. Extant methods are typically divided into two phases: first extracting frame-wise spatio-temporal features by 2DCNNs[30], two-stream convnets [38, 13] or 3DCNNs [2, 47, 12, 44, 11], and then conducting the frame-wise classification. Some works [40, 26, 39, 35] focus on extracting effective features for action segmentation. However, our work focuses on the method for classifying action and feature extraction is beyond the scope of this work. Following [9], we use I3D [2] features as input to our framework.

Action Segmentation. Earlier approaches detect action segments using a sliding window and filter out redundant hypotheses with non-maximum suppression [37, 19]. Other approaches model the temporal action sequence with a Markov model [23, 36] or an RNN [39] to classify frame-wise actions. Lea *et al.* [26] propose a spatiotemporal CNN for extracting features, which takes RGB and motion history images as input, and uses a semi-Markovian model for jointly segmenting and classifying actions.

Following the success in the speech synthesis domain, some studies have adopted a temporal convolutional network from the WaveNet model [46]. Lea *et al.* [25] propose two types of temporal convolutional networks (TCN) for action segmentation. Lei *et al.* [28] propose a network with temporal deformable convolutions and a residual stream. These approaches use temporal pooling, which help to capture long-range dependencies between actions. However, pooling operations may lose temporal information that is indispensable for action segmentation.

To address this problem, Farha *et al.* [9] propose a multi-stage architecture, namely MS-TCN. MS-TCN stacks several TCNs, which have dilated convolutions with residual connections. They also propose a loss function for penalizing over-segmentation errors. This architecture and the loss function enable the refinement of action segmentation results through each stage. Zhang *et al.* [48] propose a bilinear pooling module and combines it with MS-TCN. The above two methods are capable of capturing dependencies between actions and reducing over-segmentation errors. Yifei *et al.* [18] propose Graph-based Temporal Reasoning Module, which can be added to top of action segmenation models. Combining this module with MS-TCN improves the performance. However, there is still room for improvement, especially on large datasets with diverse action classes such as the Breakfast dataset [22]. Existing works perform with high frame-wise accuracy, but produce many false positive. which results in over-segmentation error de-

spite the multi-stage architecture and the smoothing loss. Our experiments are conducted with comparison to these state-of-the-art methods to demonstrate the efficacy of our proposal. Some works [4, 5] apply domain adaptation techniques to action segmentation, but we do not compare these works as their setting is different from our work.

Action Proposal Generation. Substantial research has been carried out in this domain, some of which is related to our approach. Existing methods for action proposal generation can broadly be divided into two types of approaches: anchor-based approaches [32, 14, 15] and anchor-free approaches [49, 33, 31]. Anchor-based approaches define multi-scale anchors with even intervals as proposals and generate confidence scores for each proposal. Anchor-free approaches first evaluate actionness or the likelihood of a frame being the start or end of an action, and then generate final predictions by leveraging these cues. Inspired by these anchor-free approaches, we additionally use an action boundary regression network for the action segmentation task. Our action boundary network regresses only boundary probabilities, regardless of it being a start or end of an action. Using predicted action boundaries, our framework refines frame-wise predictions to improve the performance of action segmentation.

3. Our Proposed Method

In this section, we introduce our approach for action segmentation, ASRF. Our framework decouples frame-wise action classification and action boundary regression. Our proposed framework consists of a long-term feature extractor and two branches, an Action Segmentation Branch (ASB) and a Boundary Regression Branch (BRB), as in Figure 1. A long-term feature extractor takes video features as input, expands the receptive field and captures long-term dependencies between action segments. Then, both branches take the features as inputs and frame-level action predictions and action boundary probabilities as outputs. Let $X = [x_1, ...x_T] \in \mathbb{R}^{T \times D}$ be the input to the ASRF, where T is the number of frames in a video and D is the dimension of the feature. Given X, our goal is to classify frame-level action classes $C = [c_1, ..., c_T]$. For each frame, we predict action boundaries $B = [b_1, ..., b_T]$, and use this to improve the classification.

3.1. Long-term Feature Extractor

Given X, the goal of a long-term feature extractor is to capture long-term dependencies between action segments and extract rich features $X' \in \mathbb{R}^{T \times D'}$, where D' is the dimension of the feature. As a long-term feature extractor, we use a temporal convolutional network (TCN) with dilated residual layers proposed in [9]. This architecture can convolve features with full temporal resolution and a large receptive field. This enables the network to capture long-term dependencies between action segments and extract shared features for the following branches. Our feature extractor consists of 10 dilated residual layers with 64 filters, each of which are followed by a dropout layer with a dropout rate of 0.5. The dilation rate is doubled at every residual convolution.

3.2. Action Segmentation Branch

Given X', the goal of the Action Segmentation Branch (ASB) is to predict frame-wise action classes C. To predict action classes, we simply use a 1D convolutional layer followed by a softmax layer. However this prediction contains some errors such as over-segmentation errors. So we add a multi-stage architecture proposed in [9] after the output layer. The first layer takes X' as input and outputs the initial predictions, and the subsequent stages refine the predictions made from previous stages. This architecture facilitates capturing temporal dependencies and recognizes action segments, preventing over-segmentation errors. Herein, each stage consists of a single temporal convolution with a kernel size of 1 and 64 filters, 10 dilated residual convolutions, and another temporal convolution for reducing the feature dimension to the number of action classes. The parameters of each dilated convolutional layer are the same as in the long-term feature extractor. We set the number of stages to 3 after the initial prediction layer as per [9].

3.3. Boundary Regression Branch

Although stacking several TCNs improves performance of action segmentation, the predictions still contain over-segmentation errors. To address this, we introduce a Boundary Regression Branch (BRB) in addition to the ASB. Given X', the BRB aims to regress the action boundary probabilities $P \in [0, 1]^T$ in a video, which are used later for refining the action segmentation results from the ASB (see Section 3.4). Action boundaries are defined as frames when an action starts and ends irrespective of action classes. Unlike the methods which use a Hidden Markov model to determine the most probable sequence of actions [23, 42], the BRB is class-agnostic, which eliminates the need for modeling the probabilities from each class to every other. Rather, the BRB only regresses the likelihood of general action boundaries. Therefore, the BRB requires far less data for training and can improve the robustness in comparison with the class-aware methods. We applies the the same structure as in the ASB to the BRB. The stacked structure also allows the refinement of action boundary predictions within the branch. In Section 4.4.1, we will explore the effect of the number of stages for action boundary regression.

3.4. Refining Action Segmentation Results

We describe how to refine the action segmentation results C from the ASB using action boundary probabilities

P_b from the BRB. First, we determine action boundaries $B \in \{0,1\}^T$ from P_b. We define B as the frame-level prediction where $P_{b,t}$ scores a local maximum and is over a certain threshold θ_p. B is our prediction for action boundaries, so we divide action segments based on these predictions. Assuming that each segment contains a single action, we assign action classes to segments based on the action segmentation predictions from the ASB. We make final predictions by majority voting on the action class within each segment. Note that this refinement process is done only during inference. In the experiments, the efficacy of this refinement strategy will be illuminated (see Section 4.2).

3.5. Loss Function

Our framework outputs both frame-wise action predictions and action boundaries. Hence, our loss function is defined as:

$$\mathcal{L} = \mathcal{L}_{asb} + \lambda \mathcal{L}_{brb} \tag{1}$$

where $\mathcal{L}_{asb}$ and $\mathcal{L}_{brb}$ are loss functions for the ASB and the BRB respectively and λ is the weight of the $\mathcal{L}_{brb}$. In our work, we set λ to 0.2 for GTEA and 0.1 for 50 Salads and the Breakfast dataset. In the following sections, we describe loss functions for each branch.

3.5.1 Loss Function for ASB.

Existing works usually adopt cross entropy as a classification loss.

$$\mathcal{L}_{ce} = \frac{1}{T} \sum_{t} -\log\left(y_{t,c}\right) \tag{2}$$

where $y_{t,c}$ is the action probability for class c at time t. However, this approach cannot penalize over-segmentation errors because there is no constraint for temporal transition of probabilities. To overcome this, the authors of [9] additionally use the Truncated Mean Squared Error (TMSE).

$$\mathcal{L}_{TMSE} = \frac{1}{TN} \sum_{t,c} \tilde{\Delta}_{t,c}^2 \tag{3}$$

$$\tilde{\Delta}_{t,c} = \begin{cases} \Delta_{t,c} & : \Delta_{t,c} \leq \tau \\ \tau & : \textit{otherwise} \end{cases} \tag{4}$$

$$\Delta_{t,c} = |\log y_{t,c} - \log y_{t-1,c}| \tag{5}$$

where T is the length of a video, N is the number of classes and τ is a threshold for the transition of probabilities.

Herein, in addition to these two loss functions, we validate two other loss functions. First, we simply impose a class weight for the cross entropy loss $L_{ce,cw}$). The frequency of different action segments differs for each action class, leading to an imbalance during training. For weighting, we use median frequency balancing [8], where the weight to each class is calculated by dividing the median of class frequencies by each class frequency. In our experiments, we also compare this weighting method with Focal Loss [34] (see Section 4.4.3).

Next, we introduce Gaussian Similarity-weighted TMSE (GS-TMSE) as a loss function, which improves upon TMSE. TMSE penalizes all frames in a video to smooth the transition of action probabilities between frames. However, this results in penalizing the frames where actions actually transition. To address this problem, we apply the Gaussian kernel to TMSE as follows:

$$\mathcal{L}_{GS-TMSE} = \frac{1}{TN} \sum_{t,c} \exp\left(-\frac{\left\|\mathbf{x}_t - \mathbf{x}_{t-1}\right\|^2}{2\sigma^2}\right) \tilde{\Delta}_{t,c}^2 \tag{6}$$

where $\mathbf{x_t}$ is an index of similarity for frame t and σ denotes variance. Because of the Gaussian kernel based on the similarity of frames, this function penalizes adjacent frames with large differences with a smaller weight. Here, we use the frame-level input feature for an index of similarity and set σ to 1.0. We also set τ in TMSE and GS-TMSE as 4, following [9].

The loss function for each prediction in the ASB is defined:

$$\mathcal{L}_{as} = \mathcal{L}_{ce} + \mathcal{L}_{GS-TMSE} \tag{7}$$

Then we average the losses for each prediciton in the ASB as follows:

$$\mathcal{L}_{asb} = \frac{1}{N_{as}} \sum_{i} \mathcal{L}_{as,i} \tag{8}$$

where N_{as} is the number of predictions in the ASB (N_{as} = 4 in our framework).

In the experiments, we compare various loss functions and their combinations for action segmentation as well as Equation 10. When combining them, we simply add each loss function except TMSE. We multiply L_{TMSE} by 0.15 and then add the other loss functions as per [9].

3.5.2 Loss Function for BRB.

We use a binary logistic regression loss function for the action boundary regression:

$$\mathcal{L}_{bl} = \frac{1}{T} \sum_{t=1}^{T} \left(w_p y_t \cdot \log p_t + (1 - y_t) \cdot \log\left(1 - p_t\right)\right) \tag{9}$$

where y_t and p_t are the ground truth and the action boundary probability for frame t, respectively. We weight positive samples by w_p since the number of frames that are action boundaries is much smaller than that of the others. We calculate the ratio of positive data points over the whole training data and use the reciprocal of this as the weight. As in the ASB, we average the losses for each boundary prediction in the BRB as follow:

$$\mathcal{L}_{brb} = \frac{1}{N_{br}} \sum_{i} \mathcal{L}_{bl,i} \tag{10}$$

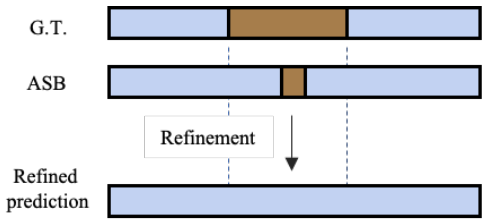

Figure 2. An example where the prediction refined the by ground truth boundaries is worse than that of the ASB. The color bands differentiate action classes, and the horizontal direction denotes time. Our framework decides action classes of each segment by majority voting. Therefore segments can be reassigned as the wrong class.

where N_{br} is the number of predictions in the BRB (N_{br} is also 4 in our framework).

4. Experiments

Datasets. For our evaluation, we use three challenging datasets: 50 Salads [41], Georgia Tech Egocentric Activities (GTEA) [10], and the Breakfast dataset [22]. The 50 Salads dataset contains 50 videos in which 25 people in total are preparing two kinds of mixed salads. The videos, each consisting of 9000 to 18000 RGB frames, depth maps and accelerometer data, are annotated with 17 action classes every frame. The GTEA dataset contains 28 videos of 7 types of daily activities, each performed by 4 different subjects. Each video is recorded egocentrically, from a camera mounted on a subject's head. The Breakfast dataset contains over 77 hours of videos, where 10 classes of actions are performed by 52 individuals in 18 different kitchens.

As per [9], for all datasets, we use spatiotemporal features extracted by I3D [2] as input to both ASB and BRB. Following [9], we downsample the videos in the 50 Salads dataset from 30 to 15 fps to ensure consistency between the datasets. For evaluation, we use 5-fold cross-validation on 50 Salads dataset and 4-fold cross-validation on the others as in [41, 22, 9].

Evaluation Metrics for Action Segmentation. The following three metrics are used for action segmentation as in [27, 26, 25]: frame-wise accuracy (Acc), segmental edit distance (Edit), and segmental F1 score with overlapping threshold $k\%$ (F1@k). Although frame-wise accuracy is commonly used as a metric for action segmentation, this measure is not sensitive to over-segmentation errors. Therefore, as well as frame-wise accuracy, we also use segmental edit distance [27, 26] and segmental F1 score [25] because both of the latter penalize over-segmentation errors.

The segmental edit distance, $S_{edit}(G, P)$ is a metric for measuring the difference between ground truth segments $G = \{G_1, ..., G_M\}$ and predicted segments $P = \{P_1, ..., P_N\}$. This metric is calculated using the Levenshtein Distance[29] between hypotheses and ground truths. For the sake of clarity, we report $(1 - S_{edit}(G, P)/max(M, N)) \times 100$ as segmental distance.

The segmental F1 score is averaged per class, in which a prediction is classified as correct if the temporal Intersection over Union (IoU) is larger than a certain threshold. This metric is invariant with respect to temporal shifts in predictions emanating from the ambiguity of the action boundary or human annotation noise.

Evaluation Metrics for Boundary Regression. The action boundary F1 score is used as an evaluation metric for the action boundary regression. We define the action boundary F1 score referencing the boundary F1 score used for semantic segmentation [6]. Let $B_{gt} \in \{0, 1\}^T$ denote if each frame is a boundary or not and $P_b \in [0, 1]^T$ denote the predicted boundary probability map. We define $B_{pred} \in \{0, 1\}^T$ as the frame-level prediction where $P_{b,t}$ is both over a threshold θ_p and is a local maximum. The precision and the recall for action boundaries are defined as:

$$Precision = \frac{1}{|B_{pred}|} \sum_{x \in B_{pred}} I\left[d\left(x, B_{gt}\right) < \theta_b\right], \quad (11)$$

$$Recall = \frac{1}{|B_{gt}|} \sum_{x \in B_{gt}} I\left[d\left(x, B_{pred}\right) < \theta_b\right], \quad (12)$$

where $I[\cdot]$ denotes the indicator function, $d(\cdot)$ is the L1 distance for temporal span and θ_b is a threshold over timestamp. We set θ_b as 5 and θ_p as 0.5 in all experiments. Then the boundary F1 metric is defined as $BF = \frac{2 \times Precision \times Recall}{Precision + Recall}$.

In addition, assuming that we have an oracle ASB which outputs the ground truth frame-wise labels, we use the BRB and refine the output of this oracle ASB. Using a single BRB, this is the upper bound of the achievable score. Therefore, when comparing multiple BRBs, this upper bound can be used to evaluate the efficacy. Note that we do not assume an oracle BRB for refining action segmentation results from the ASB, as upper bounds are not achievable even with an oracle BRB (See Figure 2).

Learning Scheme. We train the entire framework using the Adam optimizer with a learning rate of 0.0005 and batch size of 1 as per [9]. We also find the optimal number of epochs with nested cross-validation. During inference, action segmentation results from the ASB are refined using predicted action boundaries from the BRB.

4.1. Comparing ASRF with the state-of-the-art

We compare our proposed framework with existing methods on three challenging datasets: 50 Salads, Georgia Tech Egocentric Activities (GTEA) and the Breakfast dataset. Table 1 shows the results for the first two datasets.

Therein, our ASRF is superior to the state-of-the-art in terms of segmental edit distance and segmental F1 score

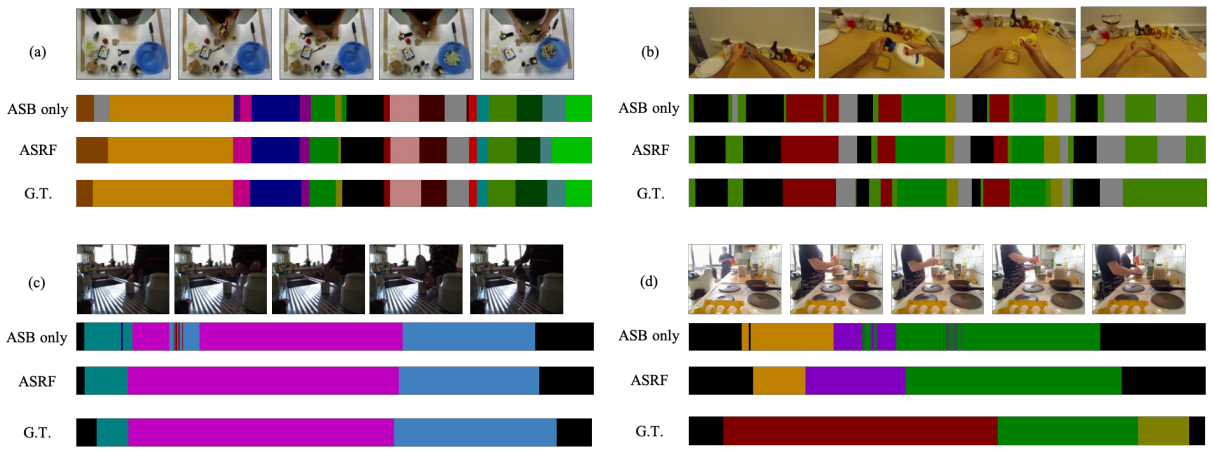

Figure 3. Qualitative results using (a) 50 Salads, (b) GTEA and (c) Breakfast dataset in comparison with predictions before refinement. (d) Failure example on Breakfast dataset

Dataset	50 Salads					GTEA				
Method	F1@{10, 25, 50}			Edit	Acc	F1@{10, 25, 50}			Edit	Acc
Bi-LSTM [39]	62.6	58.3	47.0	55.6	55.7	66.5	59.0	43.6	-	55.5
ED-TCN [25]	68.0	63.9	52.6	59.8	64.7	72.2	69.3	56.0	-	64.0
TDRN [28]	72.9	68.5	57.2	66.0	68.1	79.2	74.4	62.7	74.1	70.1
MS-TCN [25]	76.3	74.0	64.5	67.9	80.7	85.8	83.4	69.8	79.0	76.3
MS-TCN + BPGaussian [48]	78.4	75.8	66.7	71.0	80.6	86.7	84.3	72.7	77.2	**82.3**
MS-TCN + GTRM [18]	75.4	72.8	63.9	67.5	82.6	-	-	-	-	-
ASRF	**84.9**	**83.5**	**77.3**	**79.3**	**84.5**	**89.4**	**87.8**	**79.8**	**83.7**	77.3
Improvement	**+6.5**	**+5.1**	**+10.6**	**+8.3**	**+1.9**	**+2.7**	**+3.5**	**+7.1**	**+4.7**	-5.0

Table 1. Comparing our proposed method with existing methods on 50 Salads and GTEA.

with competitive frame-wise accuracy on each dataset, having up to 8.3% improvement for segmental edit distance, and up to 10.6% improvement for the segmental F1 score on 50 Salads and GTEA. As per Table 2, our framework also outperforms the existing methods by a large margin on the Breakfast dataset with respect to all evaluation metrics.

We find that our framework offers better results for stricter overlap thresholds when calculating the segmental F1 score. On GTEA, our framework and MS-TCN with bilinear pooling [48] perform similarly in terms of F1@10. However, our framework outperforms [48] on F1@50 by 7.1%. This shows that our framework is capable of recognizing action segments which overlap markedly with the ground truth segments.

In the case of GTEA, the frame-wise accuracy for our framework is inferior to that of [48]. We hypothesize that our framework predicted action boundaries off by some margin, therefore affecting the frame-wise accuracy. The rise in the segmental metrics support our hypothesis, as these errors are not accounted for here.

Qualitative results are presented in Figure 3. Predictions before refinement have some over-segmentation errors, but our framework can reduce these using action boundaries (Figure 3 (a), (b), (c)). Figure 3 (d) shows a failure case of our framework. However, the BRB has limits in the sense that it cannot reassign completely incorrect segments of inferred action classes by the ASB.

4.2. Effect of our refining paradigm

In this section, we show the effect of refinement on the action segmentation metrics using action boundaries predicted by the BRB. In addition to the refinement with the BRB, we evaluate three other postprocessing methods: *i)* **Relabeling.** relabeling actions of segments shorter than a certain temporal span θ_t with the action of the previous segment, *ii)* **Smoothing.** smoothing action probabilities using the 1D Gaussian filter with a kernel size K, and *iii)* **Similarity.** refinement with predicted action boundaries based on frame-level similarity. We measure the similarity of frames using frame-level features as in Section 3.5.1, and decide action boundary positions based on where the similarity is a local minimum.

Table 3 shows the action segmentation results of the ASB and those after refinement on the 50 Salads dataset. As observed, our refinement method with action boundaries improves action segmentation results by over 8.8% for every segmental metric with comparable frame-wise accuracy. Especially the stricter we set the IOU threshold for the segmental F1 score, the better the ASRF scores. This shows

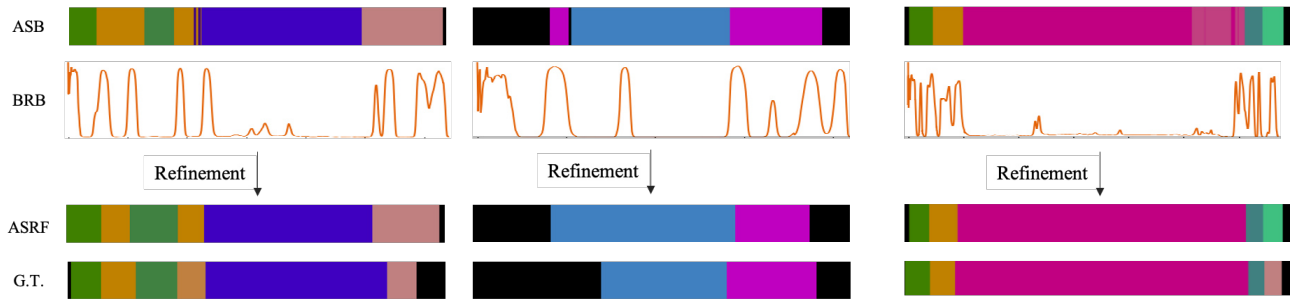

Figure 4. Examples of refinement process on the Breakfast dataset. The first row shows predictions from the ASB and the second row shows predicted boundary probabilities. Combining them, the ASRF outputs final predictions (the third row).

Breakfast	F1@{10, 25, 50}			Edit	Acc
ED-TCN [25] *	-	-	-	-	43.3
HTK [24]	-	-	-	-	50.7
TCFPN [7]	-	-	-	-	52.0
HTK(64) [23]	-	-	-	-	56.3
GRU [7] *	-	-	-	-	60.6
MS-TCN [9]	58.2	52.9	40.8	61.4	65.1
MS-TCN+GTRM [18]	57.5	54.0	43.3	58.7	65.0
ASRF	**74.3**	**68.9**	**56.1**	**72.4**	**67.6**
Improvement	**+16.1**	**+14.9**	**+12.8**	**+13.7**	**+2.5**

Table 2. Comparing our proposed method with existing methods on the Breakfast dataset. * reported results by the author of [7].

	F1@{10, 20, 50}			Edit	Acc
w/o post-processing	76.1	74.5	66.7	68.5	82.6
Relabeling ($\theta_t = 5$)	81.5	79.7	71.6	74.8	82.6
Relabeling ($\theta_t = 15$)	82.4	80.6	72.5	76.2	82.7
Smoothing ($K = 5$)	80.7	78.9	70.8	73.5	82.6
Smoothing ($K = 15$)	80.8	79.0	70.9	73.6	82.6
Similarity	39.8	30.9	18.1	32.8	40.0
ASRF	**84.9**	**83.5**	**77.3**	**79.3**	**84.5**

Table 3. Effect of refinement strategy using action boundaries from the BRB on the 50 Salads dataset.

	F1@{10, 20, 50}			Edit	Acc
Segment classifier	51.8	49.0	40.0	42.8	58.5
ASRF	**84.9**	**83.5**	**77.3**	**79.3**	**84.5**

Table 4. Comparing our proposed method with the segment-level classification method on the 50 Salads dataset.

that the ASRF not only prevents over-segmentation errors but also generates predictions that highly overlap with the ground truth. In addition, our method is superior to other ways of postprocessing in terms of every metric. Smoothing has less impact on the metrics than our ASRF, and Similarity has a negative effect. Although Relabeling is better than the other two methods, this method is highly dependent on the hyperparameter θ_t. With a large θ_t, the method can reduce over-segmentation errors, but drops small action segments. On the other hand, our framework can detect action boundaries and capture action segments irrespective of temporal length, which results in better performance.

Figure 4 shows our refinement process on the Breakfast dataset. The BRB outputs some false positives, but they are ignored when the ASB captures action segments without over-segmentation errors (the middle of Figure 4). This shows that our framework not only refines frame-level action predictions with boundaries, but selects reasonable action boundaries during refinement as well. Therefore, the ASB and the BRB are mutually supportive.

4.3. Comparison with segment-level classifier

Our framework classifies actions by the frame level before predicting action boundaries, then reassigning action classes for each predicted action segment. Another variant is to use a TCN to classify each predicted action segment. Therefore, we compare our framework with a combination of a boundary regression model and segment-level classifier. We use a single-stage TCN with 10 dilated convolutions and two convolutions as a segment-level classifier. Note that we add global average pooling before the last convolutional layer of the single-stage TCN to aggregate temporal features. Otherwise, for both the action boundary regression and segment-level classification, we use the same networks as our method. As seen in Table 4, our framework outperforms this variant by a large margin. This shows the importance of capturing long-range dependencies between action segments in the context of action segmentation.

4.4. Ablation Study

4.4.1 Effect of the number of stages

We use a multi-stage architecture for BRB as well as ASB, which must be validated. Therefore we train a variety of multi-stage networks and evaluate their performances. Each network stage has 10 dilated convolutions and two convolutions like our proposed BRB. As can be observed in Table 5, the three-stage architecture after the initial prediction layer outperforms others in terms of action segmentation. This shows that stacking TCNs helps to regress boundaries as well. However, using four stages on the 50 Salads dataset does not improve the predictions, which is thought to be the result of overfitting due to the size of the 50 Salads dataset.

We found that the precision of the BRB is low, but

	Boundary Regression			Action Segmentation					Action Segmentation (Oracle)				
	Precision	Recall	F1 Score	F1@{10, 20, 50}			Edit	Acc	F1@{10, 20, 50}			Edit	Acc
No stage	18.8	76.4	29.9	82.0	80.4	74.4	76.7	82.7	86.8	86.2	83.5	82.2	88.5
1 stage	34.8	69.9	46.4	82.5	81.1	73.1	76.8	80.9	86.0	85.4	81.7	81.4	87.2
2 stages	38.7	66.0	48.7	83.3	82.3	76.4	78.3	82.7	86.2	85.8	84.1	82.7	88.6
3 stages	37.3	63.2	46.8	**84.9**	**83.5**	**77.3**	**79.3**	**84.5**	86.9	86.8	84.5	83.2	89.3
4 stages	37.4	63.0	46.9	84.4	82.9	75.7	78.1	82.7	86.5	86.0	83.8	82.5	87.8

Table 5. Effect of the number of stages for the BRB after the initial prediction layer on the 50 Salads dataset.

	Boundary Regression			Action Segmentation				
θ_p	Precision	Recall	F1 Score	F1@{10, 20, 50}			Edit	Acc
0.1	36.5	63.7	46.3	84.6	83.3	76.7	79.0	84.4
0.3	37.0	63.4	46.7	84.7	83.4	76.8	78.9	84.4
0.5	37.3	63.2	46.8	**84.9**	**83.5**	**77.3**	**79.3**	**84.5**
0.7	37.6	63.1	47.1	84.5	83.2	77.3	78.8	83.9
0.9	37.6	63.1	47.1	84.5	83.2	77.3	78.8	83.9

Table 6. Comparing the effect of θ_p for action boundary decision

Loss Function	F1@{10, 20, 50}			Edit	Acc
$\mathcal{L}_{ce}$ + $\mathcal{L}_{TMSE}$	79.7	77.7	69.1	73.4	79.8
$\mathcal{L}_{ce}$ + $\mathcal{L}_{GS-TMSE}$	80.9	79.4	72.7	74.6	81.6
$\mathcal{L}_{focal}$ + $\mathcal{L}_{TMSE}$	77.3	75.8	67.1	71.3	78.0
$\mathcal{L}_{focal}$ + $\mathcal{L}_{GS-TMSE}$	78.3	76.0	66.3	71.0	78.0
$\mathcal{L}_{ce,cw}$ + $\mathcal{L}_{TMSE}$	83.3	81.8	75.1	77.0	81.8
$\mathcal{L}_{ce,cw}$ + $\mathcal{L}_{GS-TMSE}$	**84.9**	**83.5**	**77.3**	**79.3**	**84.5**

Table 7. Comparing combinations of loss functions on the 50 Salads dataset.

overdetected boundaries can still help the refinement of frame-wise action predictions. The reason of the low precision is that it is difficult to predict action boundaries precisely because of the ambiguity of human annotations and the action boundary itself. In addition, the θ_b for calculating precision is only 5 frames, which is a strict threshold. Also, we highlight that the low precision of the BRB does not have a negative impact. In our framework, the ASB and the BRB are complementary, so even if nonexistent action boundaries are predicted, some of them disappear when combining outputs from the ASB. In addition, segmental edit distance and segmental F1 score with IoU thresholds tolerate the shifts of action starting and ending points. Results show that there is little to no correlation between boundary scores and action segmentation scores. This is because segmental metrics are tolerant to shifts in action boundary predictions. Therefore, oracle experiments are selected for the evaluation of boundary regression in our framework.

4.4.2 Impact of θ_p for boundary regression

As described in Section 3.5.2, we set the threshold $\theta_p = 0.5$ for deciding action boundaries from outputs predicted by BRB. Table 6 shows the impact of θ_p for boundary regression. Note that we do not use an oracle ASB in this experiment, because an overdetecting BRB would obtain perfect results, therefore setting θ_p to a minimal value would be best performing. As in Table 6, our framework is not particularly sensitive to θ_p. For our framework, $\theta_p = 0.5$ showed the best balance.

4.4.3 Comparing loss functions for the ASB

Table 7 compares the results of each combination of loss functions. Our proposed smoothing loss $\mathcal{L}_{GS-TMSE}$ improves the segmental metrics by up to 3.6%. This shows that $\mathcal{L}_{GS-TMSE}$ can smooth the transition of action probabilities without penalizing frames where actions actually transition, which results in alleviating over-segmentation errors. In addition, imposing class weight to the cross also improves all the metrics while $\mathcal{L}_{focal}$ has a negative influence. Though tuning hyperparameters in $\mathcal{L}_{focal}$ may still help to improve performance, it is costly to do so for every dataset. Our combination of loss functions enable better performance in comparison with loss functions used in existing works.

5. Conclusions

We proposed an effective framework for the action segmentation task. In addition to an action segmentation network, we use an action boundary regression network for refining action segmentation results with predicted action boundaries. We also compared and evaluated various loss functions and their combinations for action segmentation. Through experiments, it was confirmed that our framework is capable of recognizing action segments and alleviating over-segmentation errors. Our framework outperforms state-of-the-art methods on the three challenging datasets, especially on the Breakfast dataset by a large margin, which contains a larger number of videos and action classes than the others.

Acknowledgement. Computational resource of AI Bridging Cloud Infrastructure (ABCI) provided by National Institute of Advanced Industrial Science and Technology (AIST) was used.

References

[1] J. K. Aggarwal and M. S. Ryoo. Human activity analysis: A review. *ACM Computing Surveys (CSUR)*, 43(3), 2011.

[2] A. Carreira and A. Zisserman. Quo Vadis, Action Recognition? A New Model and the Kinetics Dataset. In *IEEE/CVF International Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6299–6308, 2017.

[3] J. Carreira, E. Noland, C. Hillier, and A. Zisserman. A Short Note on the Kinetics-700 Human Action Dataset. In *CoRR:1907.06987*, 2019.

[4] Min-Hung Chen, Baopu Li, Yingze Bao, and Ghassan Al-Regib. Action segmentation with mixed temporal domain adaptation. *The IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 605–614, 2020.

[5] Min-Hung Chen, Baopu Li, Yingze Bao, Ghassan AlRegib, and Zsolt Kira1. Action segmentation with joint self-supervised temporal domain adaptation. *IEEE/CVF International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[6] Gabriela Csurka, Diane Larlus, and Florent Perronnin. What is a good evaluation measure for semantic segmentation? *British Machine Vision Conference (BMVC)*, 2013.

[7] Li Ding and Chenliang Xu. Weakly-supervised action segmentation with iterative soft boundary assignment. *IEEE/CVF International Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6508–6516, 2018.

[8] David Eigen and Rob Fergus. Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. *IEEE International Conference on Computer Vision (ICCV)*, pages 2650–2658, 2015.

[9] Yazan Abu Farha and Juergen Gall. Ms-tcn: Multi-stage temporal convolutional network for action segmentation. *IEEE/CVF International Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3575–3584, 2019.

[10] Alireza Fathi, Xiaofeng Ren, and James M Rehg. Learning to recognize objects in egocentric activities. *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3281–3288, 2011.

[11] Christoph Feichtenhofer. X3d: Expanding architectures for efficient video recognition. *IEEE/CVF International Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 203–213, 2020.

[12] Christoph Feichtenhofer, Haoqi Fan, Jitendra Malik, and Kaiming He. Slowfast networks for video recognition. *IEEE International Conference on Computer Vision (ICCV)*, pages 6202–6211, 2019.

[13] Christoph Feichtenhofer, Axel Pinz, and Andrew Zisserman. Convolutional Two-Stream Network Fusion for Video Action Recognition. In *IEEE/CVF International Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1933–1941, 2016.

[14] Jiyang Gao, Zhenheng Yang, and Ram Nevatia. Cascaded boundary regression for temporal action detection. *British Machine Vision Conference (BMVC)*, 2017.

[15] Jiyang Gao, Zhenheng Yang, Chen Sun, Kan Chen, and Ram Nevatia. Turn tap: Temporal unit regression network for temporal action proposals. *IEEE International Conference on Computer Vision (ICCV)*, pages 3648–3656, 2017.

[16] K. Hara, H. Kataoka, and Y. Satoh. Can Spatiotemporal 3D CNNs Retrace the History of 2D CNNs and ImageNet? In *IEEE/CVF International Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6546–6555, 2018.

[17] De-An Huang, Fei Fei Li, and Juan Carlos Niebles. Connectionist temporal modeling for weakly supervised action labeling. *European Conference on Computer Vision (ECCV)*, pages 137–153, 2016.

[18] Yifei Huang, Yusuke Sugano, and Yoichi Sato. Improving action segmentation via graph based temporal reasoning. *IEEE/CVF International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[19] Svebor Karaman, Lorenzo Seidenari, Alberto, and Del Bimbo. Fast saliency based pooling of fisher encoded dense trajectories. *European Conference on Computer Vision (ECCV), THUMOS Workshop*, 2014.

[20] Andrej Karpathy, George Toderici, Sanketh Shetty, Thomas Leung, Rahul Sukthankar, and Li Fei-Fei. Large-scale video classification with convolutional neural networks. In *IEEE/CVF International Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1725–1732, 2014.

[21] W. Kay, J. Carreira, K. Simonyan, B. Zhang, C. Hillier, S. Vijayanarasimhan, F. Viola, T. Green, T. Back, P. Natsev, Suleyman M., and A. Zisserman. The Kinetics Human Action Video Dataset. In *CoRR:1705.06950*, 2017.

[22] Hilde Kuehne, Ali Arslan, and Thomas Serre. The language of actions: Recovering the syntax and semantics of goaldirected human activities. *IEEE/CVF International Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 780–787, 2014.

[23] Hilde Kuehne, Juergen Gall, and Thomas Serre. An end-to-end generative framework for video segmentation and recognition. *IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1–8, 2016.

[24] Hilde Kuehne, Alexander Richard, and Juergen Gall. Weakly supervised learning of actions from transcripts. *Computer Vision and Image Understanding*, 163:78–89, 2017.

[25] Colin Lea, Michael D. Flynn, Rene Vidal, Austin Reiter, and Gregory D. Hager. Temporal convolutional networks for action segmentation and detection. *IEEE/CVF International Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1003–1012, 2017.

[26] Colin Lea, Austin Reiter, René Vidal, and Gregory D. Hager. Segmental spatiotemporal cnns for fine-grained action segmentation. *European Conference on Computer Vision (ECCV)*, pages 36–52, 2016.

[27] Colin Lea, René Vidal, and Gregory D. Hager. Learning convolutional action primitives for fine-grained action recognition. *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1642–1649, 2016.

[28] Peng Lei and Sinisa Todorovic. Temporal deformable residual networks for action segmentation in videos. *IEEE/CVF International Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6742—6751, 2018.

[29] Vladimir Iosifovich Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics-Doklady*, 10(8):707–710, 1966.

[30] Ji Lin, Chuang Gan, and Song Han. Tsm: Temporal shift module for efficient video understanding. *IEEE International Conference on Computer Vision (ICCV)*, pages 7083–7093, 2019.

[31] Tianwei Lin, Xiao Liu, Xin Li, Errui Ding, and Shilei Wen. Bmn: Boundary-matching network for temporal action proposal generation. *IEEE International Conference on Computer Vision (ICCV)*, pages 3889–3898, 2019.

[32] Tianwei Lin, Xu Zhao, and Zheng Shou. Single shot temporal action detection. *ACM Multimedia*, pages 988–996, 2017.

[33] Tianwei Lin, Xu Zhao, Haisheng Su, Chongjing Wang, and Ming Yang. Bsn: Boundary sensitive network for temporal action proposal generation. *European Conference on Computer Vision (ECCV)*, pages 3–19, 2018.

[34] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. *IEEE International Conference on Computer Vision (ICCV)*, 2017.

[35] Khoi-Nguyen C. Mac, Dhiraj Joshi, Raymond A. Yeh, Jinjun Xiong, Rogerio S. Feris, and Minh N. Do. Learning motion in feature space: Locally-consistent deformable convolution networks for fine-grained action detection. pages 6282–6291, October 2019.

[36] Alexander Richard, Hilde Kuehne, and Juergen Gall. Weakly supervised action learning with rnn based fine-to-coarse modeling. *IEEE/CVF International Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 754–763, 2017.

[37] Marcus Rohrbach, Sikandar Amin, Mykhaylo Andriluka, and Bernt Schiele. A database for fine grained activity detection of cooking activities. *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1194–1201, 2012.

[38] Karen Simonyan and Andrew Zisserman. Two-Stream Convolutional Networks for Action Recognition in Videos. In *Proceedings of the Advances in Neural Information Processing Systems (NIPS)*, pages 568–576, 2014.

[39] Bharat Singh, Tim K Marks, Michael Jones, Oncel Tuzel, and Ming Shao. A multi-stream bi-directional recurrent neural network for fine-grained action detection. *IEEE/CVF International Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1961–1970, 2016.

[40] Suriya Singh, Chetan Arora, and C. V. Jawahar. First person action recognition using deep learned descriptors. *IEEE/CVF International Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2620–2628, 2016.

[41] Sebastian Stein and Stephen J McKenna. Combining embedded accelerometers with computer vision for recognizing food preparation activities. *ACM International Joint Conference on Pervasive and Ubiquitous Computing*, pages 729–738, 2013.

[42] Kevin Tang, Li Fei-Fei, and Daphne Koller. Learning latent temporal structure for complex event detection. *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1250–1257, 2012.

[43] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri. Learning Spatiotemporal Features with 3D Convolutional Networks. In *IEEE International Conference on Comuter Vision (ICCV)*, pages 4489–4497, 2015.

[44] Du Tran, Heng Wang, Lorenzo Torresani, and Matt Feiszli. Video classification with channel-separated convolutional networks. *IEEE International Conference on Computer Vision (ICCV)*, pages 5552–5561, 2019.

[45] Du Tran, Heng Wang, Lorenzo Torresani, Jamie Ray, Yann LeCun, and Manohar Paluri. A Closer Look at Spatiotemporal Convolutions for Action Recognition. In *IEEE/CVF International Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6450–6459, 2018.

[46] Aäron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew W. Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio. *ISCA Speech Synthesis Workshop (SSW)*, 2016.

[47] Saining Xie, Chen Sun, Jonathan Huang, Zhuowen Tu, and Kevin Murphy. Rethinking spatiotemporal feature learning: Speed-accuracy trade-offs in video classification. *European Conference on Computer Vision (ECCV)*, 2018.

[48] Yan Zhang, Krikamol Muandet, Qianli Ma, Heiko Neumann, and Siyu Tang. Low-rank random tensor for bilinear pooling. *CoRR*, abs/1906.01004, 2019.

[49] Yue Zhao, Yuanjun Xiong, Limin Wang, Zhirong Wu, Xiaoou Tang, and Dahua Lin. Temporal action detection with structured segment networks. *IEEE International Conference on Computer Vision (ICCV)*, pages 2933–2942, 2017.