Refining Action Segmentation with Hierarchical Video Representations

Hyemin Ahn[1] and Dongheui Lee[1,2]
[1]German Aerospace Center (DLR) [2]Technical University of Munich
{hyemin.ahn, dongheui.lee}@dlr.de

Abstract

*In this paper, we propose **Hierarchical Action Segmentation Refiner (HASR)**, which can refine temporal action segmentation results from various models by understanding the overall context of a given video in a hierarchical way. When a backbone model for action segmentation estimates how the given video can be segmented, our model extracts segment-level representations based on frame-level features, and extracts a video-level representation based on the segment-level representations. Based on these hierarchical representations, our model can refer to the overall context of the entire video, and predict how the segment labels that are out of context should be corrected. Our HASR can be plugged into various action segmentation models (MS-TCN, SSTDA, ASRF), and improve the performance of state-of-the-art models based on three challenging datasets (GTEA, 50Salads, and Breakfast). For example, in 50Salads dataset, the segmental edit score improves from 67.9% to 77.4% (MS-TCN), from 75.8% to 77.3% (SSTDA), from 79.3% to 81.0% (ASRF). In addition, our model can refine the segmentation result from the unseen backbone model, which was not referred to when training HASR. This generalization performance would make HASR be an effective tool for boosting up the existing approaches for temporal action segmentation. Our code is available at https://github.com/cotton-ahn/HASR_iccv2021.*

1. Introduction

Enabling an intelligent agent to understand a human action from videos is crucial for various applications such as interactive robots, surveillance, and activity analysis. Regarding this, one main line of research would be a video action recognition [14, 1, 7], which predicts the action class label for properly trimmed videos. On the other hand, there exist researches for understanding untrimmed videos with fine-grained class labels [19, 27, 21], so that agents can localize or segment human actions from long-term videos. In this paper, we focus on a task of temporal action segmentation, which is to divide video frames into segments as well as to predict action class labels for the segments.

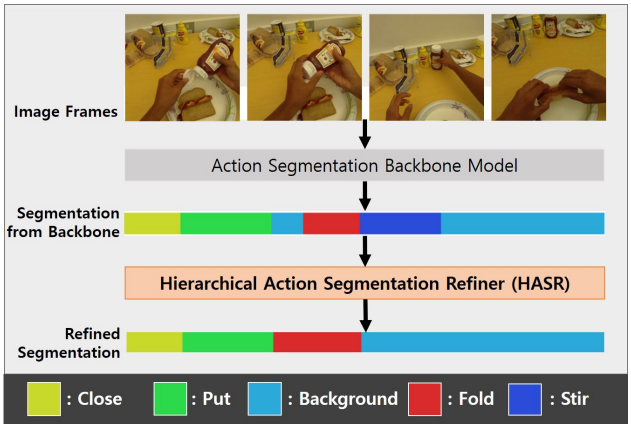

Figure 1. An illustration of how the proposed Hierarchical Action Segmentation Refiner (HASR) works. HASR refines the action segmentation result from the backbone model, after understanding the overall context of the entire video in a hierarchical way. This example is obtained when HASR is applied to improve the performance of MS-TCN [5], given a video of making a hot-dog as an input. It shows that the label of 'stir', which is not relevant to a hot-dog, changes to the 'background' (no action) label.

Researches in temporal action segmentation have been improved to successfully segment thousands of video frames recorded with 15 fps [5, 3, 10]. However, we find out that existing state-of-the-art models sometimes generate segmentation results including action labels that are out of overall context. For example, as shown in Figure 1, the label of 'stir' appears when the input video is about making a hot-dog. We claim that this phenomenon happens since existing approaches focus on frame-level feature information, but not on the overall context of the video.

In this paper, we propose a Hierarchical Action Segmentation Refiner (HASR), which can extract the hierarchical video representations to understand the overall context, and exploit them to refine the results from action segmentation backbone models. Here, the action segmentation backbone models refer to any existing approaches for a temporal action segmentation task, such as MS-TCN [5], SSTDA [3] and ASRF [10]. Figure 1 shows an illustration of how

HASR works. Based on the action segmentation results from a backbone model, HASR extracts segment-level representations based on given frame-level features, and extracts a video-level representation based on the segment-level representations. With these hierarchical video representations, our model predicts how the segment labels that are out of context should be corrected.

HASR is trained in a supervised way, by referring to the segmentation results from the pretrained backbone models as well as the ground truth segment information. The interesting point is, after HASR is trained to learn how to refine the segmentation results from backbone models A, B and C, it is also effective for refining the results from another unseen backbone model D. We validate this generalization performance from experiments, and show that our model can be extensively used to improve the segmentation results from unseen backbone models. Our contribution can be summarized as follows:

- We propose Hierarchical Action Segmentation Refiner (HASR), which can refine the action segmentation results from the backbone model by understanding the overall context of the entire video in a hierarchical way. HASR can be plugged into various backbone models, and it is also possible to use HASR to refine segmentation results from unseen backbone models.
- Our approach can boost up the performance of existing state-of-the-art action segmentation models. For example, based on the 50Salads dataset [22], our model improves the segmental edit score from 67.9% to 77.4% for MS-TCN [5], from 75.8% to 77.3% for SSTDA [3], from 79.3% to 81.0% for ASRF [10].

2. Related Works

In the study of temporal action segmentation, the main objective is to segment given video frames, and to label each segment with corresponding action classes. Due to its various applications, many researchers contributed to developing methodologies for action segmentation. For example, there are approaches that use a sliding window with non-maximum suppression to detect action segments [20, 11], approaches based on the hidden Markov model [13, 23], and approaches based on the temporal convolutional networks (TCN) [15, 16, 17, 5].

Among these works, we would like to highlight a multi-stage temporal convolutional network (MS-TCN) [5], which is a stack of several TCN. Here, TCN consists of several dilated 1D convolutions with residual connections, and these multiple dilated convolutions enable MS-TCN to make its modeling capacity and temporal receptive field larger. Based on this, MS-TCN could conduct action segmentation with higher frame-per-second compared to other works [15, 17], and achieved state-of-the-art results.

In addition, there exist several approaches to improve the performance of action segmentation models such as MS-TCN [3, 26, 9, 10]. Chen *et al.* [3] proposed to apply self-supervised domain adaptation techniques when training a model such as MS-TCN, and it exploits unlabeled videos to boost the performance of action segmentation. Wang *et al.* [26] suggested a framework named boundary-aware cascade network (BCN), and Yifei et al. [9] suggested a graph-based temporal reasoning module (GTRM). These [26, 9] can be easily attached to various action segmentation models to improve performance. Ishikawa *et al.* proposed an action segment refinement framework (ASRF) [10], which decouples the problems of frame-level action segmentation and action boundary regression. The framework consists of two branches, one for generating frame-level action segmentation information, another for estimating the time boundary information of action segments. It firstly estimates frame-level segment class labels, and refines the result based on the estimated time boundary of segments. Based on this, [10] has recently earned state-of-the-art performance, with three challenging datasets [6, 22, 12], which will be also used in our experiments.

However, as we mentioned earlier, we observed that existing state-of-the-art models such as [5, 3, 10] sometimes generate segment class labels that are out-of-context. For example, a 'stir' label appears in a hot-dog making video as shown in Figure 1, or a 'cut cheese' label appears in a salad-making video even if the human finished mixing up the cheese to the salad. Our goal is to refine these out-of-context segmentation results by understanding the overall context of a given video input in a hierarchical way.

3. Hierarchical Action Segmentation Refiner

3.1. Overall Structure

Overview Figure 2 shows the overview of our Hierarchical Action Segmentation Refiner (HASR). It aims to refine the frame-level action segmentation result from a backbone by incorporating hierarchical video representations. Based on the segmentation result from the backbone model and frame-level features, our model first extracts hierarchical video representations, which are segment-level and video-level representations. Segment-level representation encodes the frame-level features of each segment, and video-level representation encodes the obtained segment-level representations to represent the video consisting of multiple segments. Then, our model refines the segment-level labels based on the hierarchical video representations and the embedding vectors of segment labels. Refined segment-level labels are rolled out into frame-level labels, based on the time boundary information of refined segments.

Note that the time boundary information of the refined segments is maintained as same as the unrefined segments,

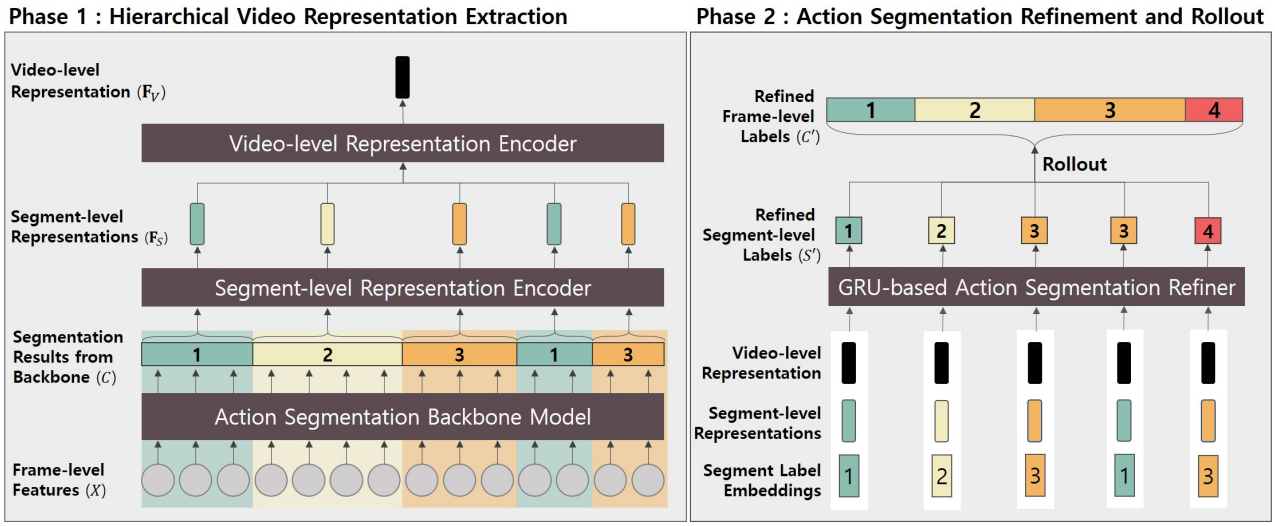

Figure 2. The overview of the proposed Hierarchical Action Segmentation Refiner (HASR). The objective of our model is to refine the segmentation result from the backbone model pretrained for action segmentation. First, our model extracts hierarchical video representations, which are segment-level and video-level representations. Second, it refines the backbone's segmentation result based on the hierarchical video representations. Note that our model is designed to be attached to various action segmentation backbone models such as [5, 3, 10].

which makes our model focus on correcting false segment labels from the backbone model. This design is based on our observation that many failure cases are due to over-segmentation or mis-segmentation. Therefore, we chose to propose a refiner model which trusts the time boundary information of segments that are estimated from backbone models, and mainly focuses on fixing false segment labels.

Formulations The objective of a temporal action segmentation task is to estimate class labels of video frames. Let $I = \{i(t)\}_{t=1...T}$ denote T image frames consisting given video input. Frame-level features $X = \{x(t)\}_{t=1...T}$ can be extracted from I, based on the existing approaches for feature extraction from human action videos [2, 25]. Let $\mathcal{F}_B$ denote a backbone model pretrained for temporal action segmentation. The output of $\mathcal{F}_B$ is a set of class labels from each frame, such that $C = \mathcal{F}_B(X)$, where $C = \{c(t)\}_{t=1...T}$. If input frames can be divided into N segments based on C, the class label and time information of N segments can be represented as $S = \{(c_n, t_n, l_n)\}_{n=1...N}$. Here, c_n denotes the class label of the n-th segment, t_n denotes the start time index of the n-th segment, and l_n denotes the number of frames consisting the n-th segment.

After the backbone model $\mathcal{F}_B$ estimates class labels C, our HASR refines C into C' based on two processing phases. In the first phase, our model tries to understand the input video based on the segmentation results from the backbone model. First, our model extracts a set of segment-level representations $\mathbf{F}_S = \{f_S(n)\}_{n=1...N}$. Here, $f_S(n)$ denotes the representation of the n-th segment, which is extracted by a segment-level representation encoder $\mathcal{F}_{SE}$. After extracting $\mathbf{F}_S$ from N segments, S and $\mathbf{F}_S$ are given to video-level representation encoder $\mathcal{F}_{VE}$. Then, a video-level representation $\mathbf{F}_V$ is extracted.

In the second phase, our model refines the segmentation result from the backbone by predicting which segment label needs to be corrected when considering the hierarchical video representations. Let $\mathcal{F}_R$ denotes our action segmentation refiner, which refines segment-level labels S into S'. Then, segments can be refined as $S' = \mathcal{F}_R(S, \mathbf{F}_S, \mathbf{F}_V) = \{(c'_n, t_n, l_n)\}_{n=1...N}$. Note that the time boundary information of S' is maintained as the same as S. Based on S', the frame-level action segmentation labels $C' = \{c'(t)\}_{t=1...T}$ can be simply calculated by rolling out the information inside S'.

3.2. Segment-level Representation Encoder

As mentioned in Section 3.1, the segment-level representation of the n-th segment $f_S(n)$ is resulted from the relevant encoder $\mathcal{F}_{SE}$, such that:

$$f_S(n) = \mathcal{F}_{SE}\left(c_n, X_{t_n:t_n+l_n}\right) \tag{1}$$

Here, the class label information of the n-th segment c_n as well as the frame-level features consisting the n-th segment $X_{t_n:t_n+l_n}$ are used as inputs to $\mathcal{F}_{SE}$.

Figure 3 shows how the proposed segment-level representation encoder works. First, the class label information c_n is converted into the embedding vector $L_{c_n} \in \mathbb{R}^{d_L}$, based on the embedding weight matrix $\mathbf{L} \in \mathbb{R}^{n_c \times d_L}$, where n_c denotes the number of action classes and d_L is a dimension of the segment label embedding vector. Naturally, the

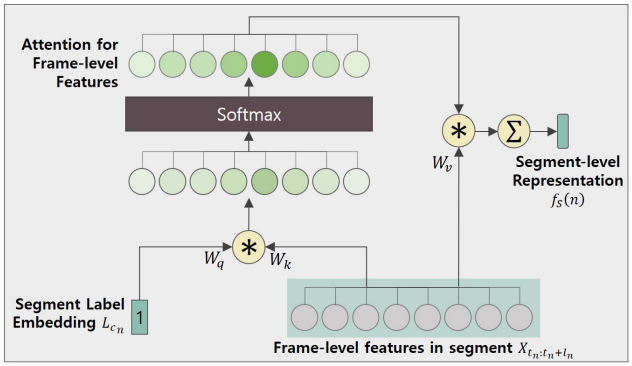

Figure 3. The visualization of the segment-level representation encoder. The segment label embedding vector is used as a query to generate attention weights for the frame-level features. Based on the attention weights, the frame-level features in the segment is weighted summed to generate a segment-level representation.

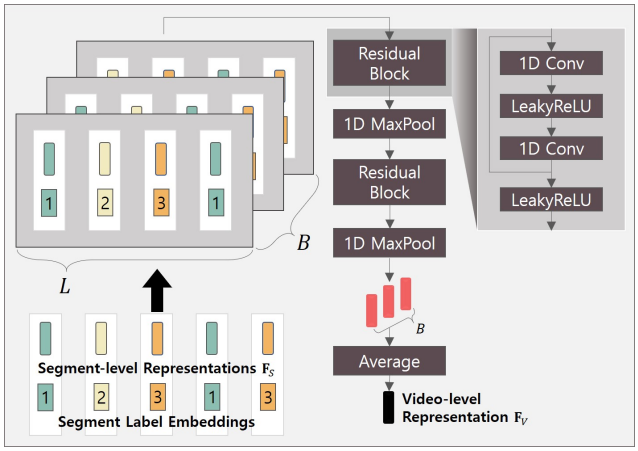

Figure 4. The structure of the video-level representation encoder. Rather than observing the whole segment information once, it observes the sampled subsequence for multiple times.

c_n-th row of $\mathbf{L}$ would be L_{c_n}, and note that $\mathbf{L}$ is also a learnable parameter.

Based on the segment label embedding vector L_{c_n}, $\mathcal{F}_{SE}$ applies a key-query-value based attention mechanism [24] to extract $f_S(n)$ from the frame-level features $X_{t_n:t_n+l_n}$. To generate a query vector q, we multiply a weight matrix W_q with L_{c_n}, such that $q = W_q L_{c_n}$. Key and value vectors are obtained by multiplying $X_{t_n:t_n+l_n}$ with W_k and W_v. Let $k(t) = W_k x(t)$ and $v(t) = W_v x(t)$ denote key and value vectors obtained from $x(t) \in X_{t_n:t_n+l_n}$. Then, an attention weight $w(t)$ can be obtained by passing the inner-product result between q and $k(t)$ to a temperature softmax layer, such that $w(t) = \frac{\exp(q^T k(t)/\tau)}{\sum_{i=t_n}^{t_n+l_n} \exp(q^T k(i)/\tau)}$, where τ denotes a temperature parameter. Based on this, the segment-level representation can be obtained by weighted summation between the attention weights and values, such that $f_S(n) = \sum_{t=t_n}^{t_n+l_n} w(t)v(t)$.

3.3. Video-level Representation Encoder

Intuitively, the process of extracting $f_S(n)$ in Section 3.2 is to interpret the visual information of the n-th segment based on the class label c_n that the backbone model estimated. However, if the backbone model fails to estimate c_n correctly, $f_S(n)$ might represent the n-th segment in a wrong way. This might affect our video-level representation encoder $\mathcal{F}_{VE}$, which gets $\mathbf{F}_S = \{f_S(n)\}_{n=1...N}$ as an input to encode a representation of the whole video.

To address this issue, we propose $\mathcal{F}_{VE}$ which understands the video-level information in a sample-based way. Figure 4 shows how the proposed video-level representation encoder works. The input I_{VE} is a sequence of segment information, which is made by concatenating segment-level representations and segment-label embedding vectors, such that $I_{VE} = \{[f_S(n); L_{c_n}]\}_{n=1...N}$. Rather than observing the whole sequence I_{VE} once, our strategy is to observe a set of noisy sequences which are sampled from I_{VE}. This would enable our $\mathcal{F}_{VE}$ to catch the overall context of the entire video even if an incorrectly interpreted segment-level representation $f_S(n)$ is included in the input I_{VE}.

After sampling B sequences which length is L and the dimension of each element is D, a set of sampled sequences from I_{VE} can be reshaped into a matrix size of $B \times L \times D$. To extract a meaningful video-level representation from this, $\mathcal{F}_{VE}$ passes this matrix into several residual blocks and max-pooling layers as shown in Figure 4. The number of residual blocks and max-pooling layers is $\lfloor \log_2 L \rfloor$, so that the matrix can shrink into the matrix which size of $B \times D$. Here, the residual block is a modified version of the ones from [8]. After that, $\mathcal{F}_{VE}$ averages the resulted in $B \times D$ matrix into a vector of dimension D, which would become our video-level representation $\mathbf{F}_V$.

3.4. Action Segmentation Refiner

The input for action segmentation refiner $\mathcal{F}_R$ is constructed as $\{[L_{c_n}; f_S(n); \mathbf{F}_V]\}_{n=1...N}$. Based on this, $\mathcal{F}_R$ can refer to (1) a segment label information from the backbone model, (2) a segment-level representation which summarizes the frame-level features consisting of each segment, (3) a video-level representation which encodes overall contextual information of the given video. Our action segmentation refiner $\mathcal{F}_R$ is recurrent neural networks (RNNs) with gated recurrent unit (GRU) cells [4]. Based on given inputs, it classifies segment-level label information $S' = \{c'_n, t_n, l_n\}_{n=1...N}$. Here, note that the time boundary information of S' is maintained same as unrefined segments S. Based on S', the frame-level action segmentation class labels $C' = \{c'(t)\}_{t=1...T}$ can be calculated by rolling out the information of S'.

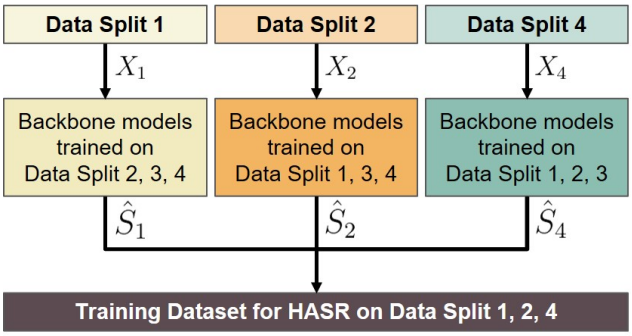

Figure 5. An example of data collection process for training our model, when the video dataset can be split into four.

3.5. Training Procedure

To train our model in a supervised way, we need a dataset consists of $(X, \hat{S}, \bar{S})$, where $\hat{S} = \{(\hat{c}_n, t_n, l_n)\}_{n=1...N}$ denotes a predicted segmentation information from the backbone model, and $\bar{S} = \{(\bar{c}_n, t_n, l_n)\}_{n=1...N}$ denotes a ground truth segmentation information that our model aims for. $\bar{S}$ can be obtained from $\hat{S}$ and $\bar{C}$, where $\bar{C} = \{\bar{c}(1) \dots \bar{c}(T)\}$ denotes a ground truth class label information in a frame-level. To annotate $\bar{c}_n$, we sample out $\mathbf{c} = \{\bar{c}(t_n), \dots \bar{c}(t_n + l_n)\}$ from $\bar{C}$, and choose $\bar{c}_n$ as the one most present in $\mathbf{c}$.

To increase the performance of our model, we observed it is crucial to collect a training dataset from various examples of $\hat{S}$. Figure 5 shows an illustration of how the training dataset for our model can be collected when a whole data can be divided into 4 splits. This figure shows the process of collecting $\hat{S}$ when training our model based on data splits 1, 2, and 4. From each split, $\hat{S}$ is collected from pretrained backbone models that supposed to be tested with this split. For example, when collecting $\hat{S}$ from the data split 1, a set of backbone models trained based on data splits 2, 3, and 4 are used to generate $\hat{S}$ from given X. In the experiment, backbone models that were saved through 10-50 training epochs are used for this data collection.

In addition, our model can be trained based on the dataset collected from various types of backbone models. For example, after being trained based on the dataset collected from backbone models such as SSTDA [3] and ASRF [10], our model can be applied to refine the segmentation result from another backbone model such as MS-TCN [5]. Relevant experiment results will be shown in Section 4.3.2.

To train our model, we use a cross-entropy loss function between ground truth segment-level labels and refined segment-level labels. Note that our loss function is not calculated from rolled-out frame-level labels. Based on the loss function, we use Adam optimizer with a learning rate of 0.0001, and with a weight decay rate of 0.0001.

4. Experiments

4.1. Datasets and Evaluation Metrics

Dataset In our experiments, three challenging datasets are used: Georgia Tech Egocentric Activities (GTEA) [6], 50Salads, [22] and Breakfast dataset [12]. The GTEA dataset consists of 28 egocentric videos of 7 different activities from 4 human subjects, such that the dataset can be divided into four splits. The 50Salads dataset consists of 50 videos of preparing salads from 25 human subjects, such that the dataset can be divided into five splits. The dataset also contains depth and accelerometer data, but here we only use the RGB frames dataset. The Breakfast dataset consists of 1712 videos of 18 different activities in kitchens from 52 human subjects. The dataset can be divided into four splits, and it is the largest one among the three datasets. For consistency, all videos from these datasets are set to 15 fps. For input to our model, we use I3D [2] features which are extracted from all video frames and provided by [5].

Evaluation Metrics When evaluating action segmentation results, we use rolled-out frame-level segment labels from our HASR. For evaluation, a frame-level accuracy (*Acc*), a segmental edit distance (*Edit*), and segmental F1 scores with different overlapping threshold $k\%$ (*F1@k*) ($k = \{10, 25, 50\}$) are used. *Acc* is the most common metric, but note that it cannot reflect the over-segmentation issues. Therefore, the segmental edit distance based on Levenshtein distance [18], whose higher value means that predicted segments need to change less to become like ground truth segments, as well as the segmental F1 scores to measure the prediction quality are additionally used.

Backbone Models Our HASR can be plugged into various existing models for action segmentation, and it can also refine the segmentation result from an unseen backbone model. In order to validate these points, we chose 3 different state-of-the-art backbone models for our experiments, which are MS-TCN [5], SSTDA [3], ASRF [10]. In addition, we also add a single-layer GRU-based model for action segmentation, which receives frame-level features and results in frame-level action class labels.

4.2. Qualitative Results

Figure 6 shows several example refinement results from HASR. Figure 6(a) shows how HASR refines the segmentation results from ASRF [10]. The given video is from the Breakfast dataset, and it is about a human making orange juice. The result shows that the backbone model misunderstands 'orange' as 'bun', the action of 'squeezing' as 'cutting', which could be due to the low brightness of the input video. Fortunately, the backbone model could correctly estimate 'take glass' and 'pour juice' in the end, but

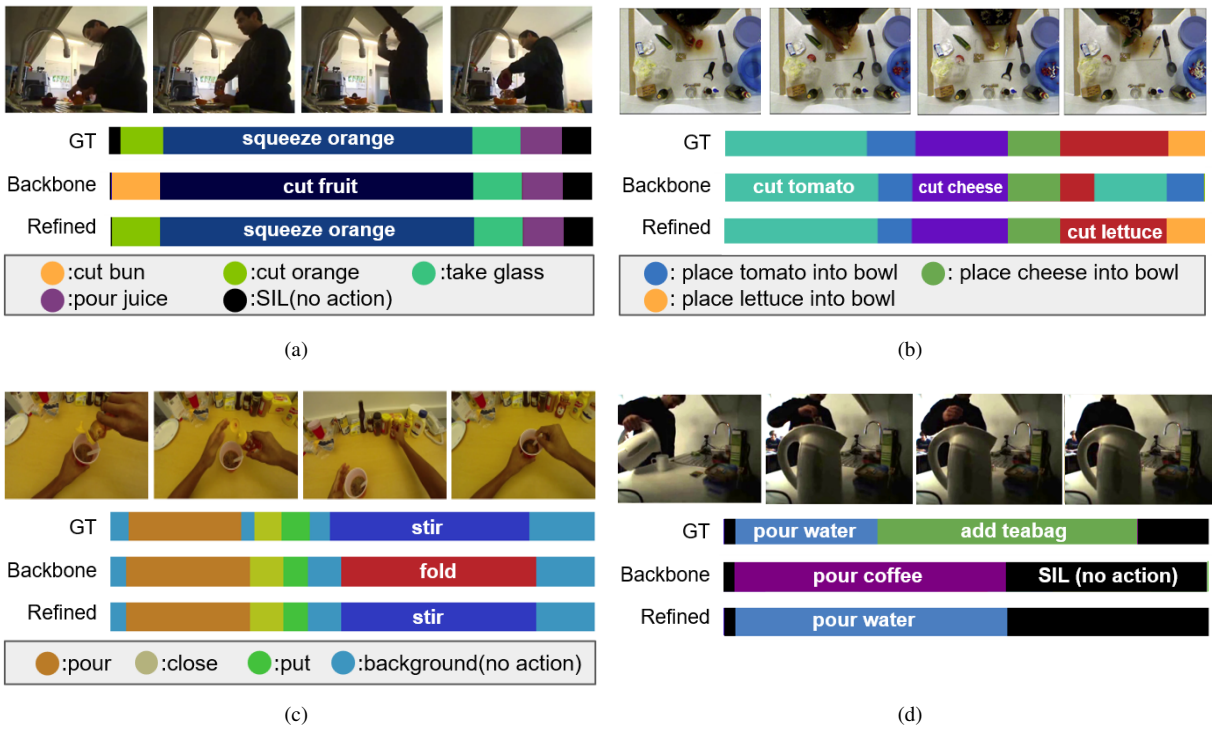

Figure 6. Qualitative results from HASR with various backbone models and datasets. Best view in color. (a) Refinement from ASRF with Breakfast dataset. (b) Refinement from SSTDA from 50Salads dataset. (c) Refinement from MS-TCN from GTEA dataset, when manually changed false segment label 'fold' exists in the backbone prediction. (d) Failure case from MS-TCN with Breakfast dataset.

the overall segmentation results are inconsistent. The refinement result shows that our model successfully corrects the segment action labels which do not match the overall context. It predicts that it is appropriate to cut and squeeze the orange first, given the video which ends with pouring the juice.

Figure 6(b) shows another result when HASR refines the result from the SSTDA [3]. The given video is from the 50Salads dataset, and it shows an egocentric video from a human when making a salad. In this video, the human cuts tomato/cheese/lettuce and moves them to a bowl. However, the backbone model estimates that the human cuts lettuce for a while, but suddenly changes to cut tomato, and move the tomato to a bowl, without moving lettuce to the bowl. The refinement result shows that our model successfully corrects these false segmentation results, which make no sense when considering the process of making a salad.

In addition, we manually change the segmentation result from the backbone in an incorrect way, and check how the proposed HASR corrects it. Figure 6(c) shows when the result from MS-TCN [5] were adjusted in a wrong way and given to the HASR. Here, the input video is about making a coffee with honey, from the GTEA dataset. After the backbone model segments the given video, we manually changed the label of 'stir' to 'fold', which is not likely to be observed in the video of making a coffee with honey. The refinement result shows that our model successfully corrects that the label should be 'stir' instead of 'fold', based on the understood context of the input video.

However, the performance of HASR is influenced by how accurately the backbone model predicts the time boundary of segments. Figure 6(d) shows a failure case, when the estimated time boundary from the backbone differs too much from the ground truth. If the backbone model segments the video with too incorrect time boundary information, it will affect HASR when extracting the video-level representation. Therefore, our future work is to fix this phenomenon with an additional module that can also correct the segment time boundary information which is estimated from the backbone model.

4.3. Quantitative Results

4.3.1 Refinement from State-of-the-Art Models

To validate that our HASR can be used to boost up the performance of existing models for action segmentation, we plugged in our HASR to various backbones such as MS-TCN [5], SSTDA [3], ASRF [10], and GRU-based model. For a single backbone model, a training dataset for HASR is collected as mentioned in Section 3.5, and it is trained to learn how to refine the segmentation result from the pre-

GTEA					
Method	F1@{10, 25, 50}			Edit	Acc
GRU	84.1	80.2	67.3	79.1	76.5
GRU + HASR	88.7	85.6	71.8	85.2	76.2
Gain	4.6	5.4	4.5	6.1	-0.3
MS-TCN [5]	85.8	83.4	69.8	79.0	76.3
MS-TCN (our impl.)	88.6	86.4	72.5	83.9	78.3
MS-TCN + HASR	89.2	87.3	73.2	85.4	77.4
Gain	0.6	0.9	0.7	1.5	-0.9
SSTDA [3]	90.0	89.1	78.0	86.2	79.8
SSTDA (our impl.)	91.1	88.8	75.6	87.9	79.4
SSTDA + HASR	90.9	88.6	76.4	87.5	78.7
Gain	-0.2	-0.2	0.8	-0.4	-0.8
ASRF [10]	89.4	87.8	79.8	83.7	77.3
ASRF (our impl.)	87.9	86.1	75.2	81.9	77.1
ASRF + HASR	89.2	87.2	74.8	84.5	76.9
Gain	1.3	1.1	-0.4	2.6	-0.2

Table 1. Refinement results based on GTEA dataset.

50Salads					
Method	F1@{10, 25, 50}			Edit	Acc
GRU	62.4	60.0	52.2	55.6	80.5
GRU + HASR	78.1	76.0	67.7	72.2	80.9
Gain	15.7	16.0	15.5	16.5	0.4
MS-TCN [5]	76.3	74.0	64.5	67.9	80.7
MS-TCN (our impl.)	77.2	74.7	64.8	70.4	80.3
MS-TCN + HASR	83.4	81.8	71.9	77.4	81.7
Gain	6.2	7.1	7.1	7.0	1.4
SSTDA [3]	83.0	81.5	73.8	75.8	83.2
SSTDA (our impl.)	80.6	78.7	70.8	74.9	82.5
SSTDA + HASR	83.5	82.1	74.1	77.3	82.7
Gain	2.9	3.4	3.3	2.4	0.2
ASRF [10]	84.9	83.5	77.3	79.3	84.5
ASRF (our impl.)	85.1	83.3	77.7	79.9	83.7
ASRF + HASR	86.6	85.7	78.5	81.0	83.9
Gain	1.5	2.4	0.9	1.2	0.2

Table 2. Refinement results based on 50Salads dataset.

Breakfast					
Method	F1@{10, 25, 50}			Edit	Acc
GRU	24.1	21.4	16.2	32.3	64.9
GRU + HASR	60.2	54.6	42.4	61.2	64.9
Gain	36.1	33.2	26.2	28.9	0.0
MS-TCN	52.6	48.1	37.9	61.7	66.3
MS-TCN (our impl.)	63.5	58.3	45.9	66.2	67.7
MS-TCN + HASR	73.2	67.9	54.4	70.8	69.8
Gain	9.7	9.6	8.6	4.6	2.0
SSTDA	75.0	69.1	55.2	73.7	70.2
SSTDA (our impl.)	70.9	64.7	50.3	70.2	67.8
SSTDA + HASR	73.1	67.1	52.6	70.0	67.6
Gain	2.2	2.4	2.3	-0.1	-0.3
ASRF	74.3	68.9	56.1	72.4	67.6
ASRF (our impl.)	73.8	68.6	56.4	72.2	68.5
ASRF + HASR	74.7	69.5	57.0	71.9	69.4
Gain	0.9	1.0	0.7	-0.3	0.9

Table 3. Refinement results based on Breakfast dataset.

trained backbone model. However, since official repositories for the backbones do not distribute pretrained models, we trained the backbone models by ourselves based on the officially distributed codes.

Table 1, 2 and 3 show the refinement results when applying HASR to various backbones. In these tables, we show the best performance of backbone models when they are trained by ourselves based on official codes. But for reference, we also show the official performance records from their papers [5, 3, 10]. Among datasets, note that the complexity of GTEA consisting of 28 videos is the lowest, and the complexity of Breakfast consisting of 1712 videos is the highest.

Results show that the performance gain from our HASR is higher when the backbone model has a lower performance. For example, it is shown that the performance gain when applying HASR to the GRU-based model or MS-TCN tends to be higher than in other cases. On the other hand, the performance gain from HASR becomes lower when the backbone model already performs well enough. Especially, on the GTEA dataset, the effect of our HASR becomes reduced when it is applied to SSTDA, which already shows the high performance.

4.3.2 Refinement from Unseen Backbones

To show the generalization performance of our HASR, we trained HASR to learn how to refine the segmentation results from the backbone models A, B and C, and used the trained HASR to refine the segmentation results from the unseen backbone model D. For this experiment, we used the GRU-based model or MS-TCN as unseen backbone models for test, and used others for training our HASR (i.e., use segmentation results from MS-TCN, SSTDA, ASRF to train HASR when GRU-based model is the unseen model).

Table 4 shows the refinement results for unseen backbone models. It is shown that the performance gains are comparable to the ones from Table 1, 2 and 3. This result shows that the proposed HASR successfully learned a general methodology about how to correct the out-of-context segment labels when considering the overall context of the given video. Based on this, we would like to highlight that our HASR has a potential to be extensively used as an effective tool for boosting up the performance of any action segmentation models.

4.3.3 Effect of Hierarchical Representations

As mentioned in Section 3, our HASR refines the segmentation results from the backbone model based on the extracted hierarchical representations of the video, which are

GTEA					
Method	F1@{10, 25, 50}			Edit	Acc
GRU	84.1	80.2	67.3	79.1	76.5
GRU+HASR	**87.4**	**83.9**	**69.2**	**83.4**	**76.8**
MS-TCN (our impl.)	88.6	86.4	72.5	83.9	**78.3**
MS-TCN+HASR	**90.0**	**88.1**	**74.8**	**85.6**	77.5
50Salads					
Method	F1@{10, 25, 50}			Edit	Acc
GRU	62.4	60.0	52.2	55.6	80.5
GRU+HASR	**74.1**	**71.8**	**63.3**	**66.6**	**80.7**
MS-TCN (our impl.)	77.2	74.7	64.8	70.4	80.3
MS-TCN+HASR	**83.5**	**82.0**	**72.1**	**77.2**	**82.1**
Breakfast					
Method	F1@{10, 25, 50}			Edit	Acc
GRU	24.1	21.4	16.2	32.3	**64.9**
GRU+HASR	**49.1**	**44.3**	**34.2**	**53.0**	62.7
MS-TCN (our impl.)	63.5	58.3	45.9	66.2	67.7
MS-TCN+HASR	**73.2**	**68.1**	**54.0**	**71.0**	**69.0**

Table 4. Refinement results for **unseen backbones**, which are MS-TCN and GRU-based action segmentation model.

segmentation-level and video-level representations. To understand the effect of each representation, we conducted an ablation study that neglects segment-level or video-level representation from the inputs for action segmentation refiner $\mathcal{F}_R$. Table 5 shows the result of the ablation study based on the 50Salads dataset. Here, MS-TCN is used as a backbone model. It shows that the highest performance can be obtained when using both representations as we proposed in Section 3. Both representations contribute to enhance the performance of our HASR, and we find that the effect of the segment-level representation tends to be higher than the video-level representation.

4.3.4 Video-level Representation Encoder: Sample-based Residual Blocks vs. RNNs

In Section 3.3, we discussed that sample-based residual blocks are used for our video-level representation encoder $\mathcal{F}_{VE}$, to catch the overall context of the given video robustly even with the noisy input information. To check the effectiveness of our method, we compared our sample-based residual blocks with simple GRU-based RNNs. Note that RNNs can be also used for $\mathcal{F}_{VE}$ since the input of $\mathcal{F}_{VE}$ is a sequence of information of segments. Table 6 shows the comparison result, which shows that the proposed sample-based residual blocks perform better than RNNs. Here, the backbone model is MS-TCN. Even the performance gap between the two could be shown as not that significant, we would like to highlight that our proposed $\mathcal{F}_{VE}$ is better than using vanilla RNNs for video-level representation encoder.

50Salads					
	F1@{10, 25, 50}			Edit	Acc
w/o Segment-level Repesentation	79.7	77.5	66.0	72.3	78.7
w/o Video-level Representation	81.6	80.2	70.0	74.9	80.7
Proposed HASR	**83.4**	**81.8**	**71.9**	**77.4**	**81.7**

Table 5. Refinement results with 50Salads dataset when segment-level or video-level representations are neglected from HASR. Here, MS-TCN is used as a backbone model.

GTEA					
Method	F1@{10, 25, 50}			Edit	Acc
RNN-based $\mathcal{F}_{VE}$	89.0	86.9	73.0	84.9	77.0
Proposed $\mathcal{F}_{VE}$	**89.2**	**87.3**	**73.2**	**85.4**	**77.4**
50Salads					
Method	F1@{10, 25, 50}			Edit	Acc
RNN-based $\mathcal{F}_{VE}$	83.3	**81.8**	71.3	76.7	81.1
Proposed $\mathcal{F}_{VE}$	**83.4**	**81.8**	**71.9**	**77.4**	**81.7**
Breakfast					
Method	F1@{10, 25, 50}			Edit	Acc
RNN-based $\mathcal{F}_{VE}$	72.8	67.6	54.0	70.7	69.2
Proposed $\mathcal{F}_{VE}$	**73.2**	**67.9**	**54.4**	**70.8**	**69.8**

Table 6. Refinement results when using a proposed sample-based residual blocks or vanilla RNNs as a video-level representation encoder. Here, MS-TCN is used as a backbone model.

5. Conclusion

In this paper, we proposed Hierarchical Action Segmentation Refiner (HASR), which can be used to boost up the performance of existing models for temporal action segmentation. The overall framework consists of an action segmentation backbone model, and our proposed refiner HASR. After the action segmentation backbone model predicts how the given video frames can be segmented, HASR first extracts a segment-level representation for each segment, based on the frame-level features consisting of the segment. Then, HASR extracts a video-level representation based on the extracted segment-level representations. Based on these hierarchical video representations, HASR is able to refine the action segmentation result from the backbone by understanding the overall context of the given video. From qualitative experiments, we showed that segment labels that are out of context can be refined based on HASR. Quantitative results showed that HASR can improve the performance of existing state-of-the-art backbone models. In addition, the result showed that HASR can improve the performance of unseen backbone models, which implies that our HASR could be an extensive tool for improving the performance of various models for temporal action segmentation.

Acknowledgements. This work has been partially supported by the Helmholtz Association.

References

[1] Joao Carreira and Andrew Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *Proceedings of the conference on computer vision and pattern recognition*, pages 6299–6308, 2017. 1

[2] Joao Carreira and Andrew Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *Poceedings of the conference on computer vision and pattern recognition*, pages 6299–6308, 2017. 3, 5

[3] Min-Hung Chen, Baopu Li, Yingze Bao, Ghassan AlRegib, and Zsolt Kira. Action segmentation with joint self-supervised temporal domain adaptation. In *Proceedings of the conference on computer vision and pattern recognition*, pages 9454–9463, 2020. 1, 2, 3, 5, 6, 7

[4] Junyoung Chung, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. In *NIPS 2014 Workshop on Deep Learning, December 2014*, 2014. 4

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

[6] Alireza Fathi, Xiaofeng Ren, and James M Rehg. Learning to recognize objects in egocentric activities. In *Proceedings of the conference on computer vision and pattern recognition*, pages 3281–3288, 2011. 2, 5

[7] Kensho Hara, Hirokatsu Kataoka, and Yutaka Satoh. Can spatiotemporal 3d cnns retrace the history of 2d cnns and imagenet? In *Proceedings of the conference on computer vision and pattern recognition*, pages 6546–6555, 2018. 1

[8] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the conference on computer vision and pattern recognition*, pages 770–778, 2016. 4

[9] Yifei Huang, Yusuke Sugano, and Yoichi Sato. Improving action segmentation via graph-based temporal reasoning. In *Proceedings of the conference on computer vision and pattern recognition*, pages 14024–14034, 2020. 2

[10] Yuchi Ishikawa, Seito Kasai, Yoshimitsu Aoki, and Hirokatsu Kataoka. Alleviating over-segmentation errors by detecting action boundaries. In *Proceedings of the winter conference on applications of computer vision*, pages 2322–2331, 2021. 1, 2, 3, 5, 6, 7

[11] Svebor Karaman, Lorenzo Seidenari, and Alberto Del Bimbo. Fast saliency based pooling of fisher encoded dense trajectories. In *ECCV THUMOS Workshop*, volume 1, page 5, 2014. 2

[12] H. Kuehne, A. B. Arslan, and T. Serre. The language of actions: Recovering the syntax and semantics of goal-directed human activities. In *Proceedings of the conference on computer vision and pattern recognition*, 2014. 2, 5

[13] Hilde Kuehne, Juergen Gall, and Thomas Serre. An end-to-end generative framework for video segmentation and recognition. In *Proceedings of the winter conference on applications of computer vision*, pages 1–8. IEEE, 2016. 2

[14] H. Kuehne, H. Jhuang, E. Garrote, T. Poggio, and T. Serre. HMDB: a large video database for human motion recognition. In *Proceedings of the international conference on computer vision*. 1

[15] Colin Lea, Michael D Flynn, Rene Vidal, Austin Reiter, and Gregory D Hager. Temporal convolutional networks for action segmentation and detection. In *Proceedings of the conference on computer vision and pattern recognition*, pages 156–165, 2017. 2

[16] Colin Lea, Rene Vidal, Austin Reiter, and Gregory D Hager. Temporal convolutional networks: A unified approach to action segmentation. In *European Conference on Computer Vision*, pages 47–54. Springer, 2016. 2

[17] Peng Lei and Sinisa Todorovic. Temporal deformable residual networks for action segmentation in videos. In *Proceedings of the conference on computer vision and pattern recognition*, pages 6742–6751, 2018. 2

[18] Vladimir I Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet physics doklady*, volume 10, pages 707–710. Soviet Union, 1966. 5

[19] Marcus Rohrbach, Sikandar Amin, Mykhaylo Andriluka, and Bernt Schiele. A database for fine grained activity detection of cooking activities. In *Proceedings of conference on computer vision and pattern recognition*, pages 1194–1201. IEEE, 2012. 1

[20] Marcus Rohrbach, Sikandar Amin, Mykhaylo Andriluka, and Bernt Schiele. A database for fine grained activity detection of cooking activities. In *Proceedings of the conference on computer vision and pattern recognition*, pages 1194–1201. IEEE, 2012. 2

[21] Dian Shao, Yue Zhao, Bo Dai, and Dahua Lin. Finegym: A hierarchical video dataset for fine-grained action understanding. In *Proceedings of the conference on computer vision and pattern recognition*, pages 2616–2625, 2020. 1

[22] Sebastian Stein and Stephen J McKenna. Combining embedded accelerometers with computer vision for recognizing food preparation activities. In *Proceedings of the 2013 ACM international joint conference on Pervasive and ubiquitous computing*, pages 729–738, 2013. 2, 5

[23] Kevin Tang, Li Fei-Fei, and Daphne Koller. Learning latent temporal structure for complex event detection. In *Proceedings of the conference on computer vision and pattern recognition*, pages 1250–1257. IEEE, 2012. 2

[24] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30, 2017. 4

[25] Heng Wang and Cordelia Schmid. Action recognition with improved trajectories. In *Proceedings of the international conference on computer vision*, pages 3551–3558, 2013. 3

[26] Zhenzhi Wang, Ziteng Gao, Limin Wang, Zhifeng Li, and Gangshan Wu. Boundary-aware cascade networks for temporal action segmentation. In *European Conference on Computer Vision*, pages 34–51. Springer, 2020. 2

[27] Hang Zhao, Antonio Torralba, Lorenzo Torresani, and Zhicheng Yan. Hacs: Human action clips and segments dataset for recognition and temporal localization. In *Proceedings of the international conference on computer vision*, pages 8668–8678, 2019. 1