SCT: Set Constrained Temporal Transformer for Set Supervised Action Segmentation

Mohsen Fayyaz and Juergen Gall
University of Bonn
Bonn, Germany
{fayyaz, gall}@iai.uni-bonn.de

Abstract

Temporal action segmentation is a topic of increasing interest, however, annotating each frame in a video is cumbersome and costly. Weakly supervised approaches therefore aim at learning temporal action segmentation from videos that are only weakly labeled. In this work, we assume that for each training video only the list of actions is given that occur in the video, but not when, how often, and in which order they occur. In order to address this task, we propose an approach that can be trained end-to-end on such data. The approach divides the video into smaller temporal regions and predicts for each region the action label and its length. In addition, the network estimates the action labels for each frame. By measuring how consistent the frame-wise predictions are with respect to the temporal regions and the annotated action labels, the network learns to divide a video into class-consistent regions. We evaluate our approach on three datasets where the approach achieves state-of-the-art results.

1. Introduction

For many applications large amount of video data needs to be analyzed. This includes temporal action segmentation, which requires to label each frame in a long video by an action class. In the last years, several strong models for temporal action segmentation have been proposed [17, 22, 10]. These models are, however, trained in a fully supervised setting, *i.e.*, each training video needs to be fully annotated by frame-wise labels. Since acquiring such annotations is very expensive, several works investigated methods to learn the models with less supervision. An example of weakly annotated training data are videos where only transcripts are provided [20, 12, 27, 29, 8, 4, 34, 24]. While transcripts of videos can be obtained from scripts or subtitles, they are still costly to obtain. In [28] it was therefore proposed to learn temporal action segmentation only from a set of action labels that are provided for a complete video of several minutes. In this case, it is only known which actions occur, but not when, in which order, or how often. This makes the task much more challenging compared to learning from transcripts or fully supervised learning.

In the work [28], the problem has been addressed by hypothesizing transcripts that contain each action label of a video at least once and then infer a frame-wise labeling of the video by aligning the hypothesized transcripts. While the approach showed that it is possible to learn from such weak annotation even for long videos, the approach does not solve the problem directly but converts it into a weakly supervised learning problem where multiple hypothesized transcripts per video are given. This is, however, ineffective since it is infeasible to align all transcripts that can be generated from a set of action labels and it uses the provided annotations not directly for learning.

In this work, we propose a method that uses the action labels that are given for each training video directly for the loss function. In this way, we can train the model in an end-to-end fashion. The main idea is to divide a video into smaller temporal regions as illustrated in Figure 1. For each region, we estimate its length and the corresponding action label. Since for each training video the set of actions is known, we can directly apply a set loss to the predicted action labels of the temporal regions, which penalizes the network if it predicts actions that are not present in the video or if it misses an action. The problem, however, is that we cannot directly apply a loss to the prediction of the region lengths. While a regularizer for the predicted length that penalizes if the lengths of the regions get too large improves the results, it is insufficient as we show in our experiments. We therefore introduce a second branch to make frame-wise predictions and measure how consistent the frame-wise predictions are with respect to the temporal regions and the annotated action labels. Using our differentiable Set Constrained Temporal Transformation (SCT), this loss affects the lengths of the regions, which substantially improves the accuracy of the model.

In our experimental evaluation on three datasets, we show that the proposed approach achieves state-of-the-art results. We furthermore thoroughly evaluate the impact of each component.

2. Related Work

Researchers in the field of action recognition have made significant advances in recent years. Methods for action recognition on trimmed video clips have acquired prominent achievements in recent years [3, 6, 5, 7, 38, 11, 35]. Although current methods achieve high accuracies on large datasets such as Kinetics [15], HMDB-51 [19], and UCF-101 [33], in realistic problems videos are not temporally trimmed.

Using the publicly available untrimmed video action segmentation datasets such as Breakfast [16] or ActivityNet [2], several works address action segmentation in videos [17, 22, 39, 10]. Early action segmentation methods utilized Markov models on top of temporal models [23, 17] or sliding window processing [30, 14]. [26] models context and length information. They show that length and context information significantly improve action segmentation. There are also other fully supervised methods that use grammars [25, 37, 18]. Recent methods try to capture the long range temporal dependencies using temporal convolutions with large receptive fields [22, 10].

The existing methods in weakly supervised action segmentation use ordered action sequences as annotation. The early works tried to get ordered sequences of actions from movie scripts [21, 9]. Bojanowski et al. [1] introduced the Hollywood extended dataset. They also proposed a method for action alignment based on discriminative clustering. Huang et al. [12] proposed to use an extended version of the CTC loss. Kuehne et al. [20] proposed a HMM-GMM based system that iteratively generates pseudo ground truth for videos during training. Richard et al. [27] use an RNN for short range temporal modeling. Most of these methods rely on iterative pseudo ground-truth generation approaches which does not allow for end-to-end training. Richard et al. [29] introduced the Neural Network Viterbi (NNV) method. They use a global length model for actions, which is updated during training. Souri et al. [34] introduce an end-to-end method which does not use any decoding during training. They use a combination of a sequence-to-sequence model on top of a temporal convolutional network to learn the given transcript of actions while learning to temporally segment the video. Li et al. [24] build upon NNV which achieves state-of-the-art results in weakly supervised action segmentation with ordering constraints.

When working with weak supervision without ordering constraints, only the set of actions is given during training. Richard et al. [28] address the problem by hypothesizing transcripts that contain each action label of a video at least once and then infer a frame-wise labeling of the video by aligning the hypothesized transcripts. They showed that it is possible to learn from such weak annotation even for long videos, but they do not solve the problem directly. They convert the problem into a weakly supervised learning problem where multiple hypothesized transcripts per video are given. This is, however, ineffective since it is infeasible to align all transcripts that can be generated from a set of action labels and it uses the provided annotations not directly for learning.

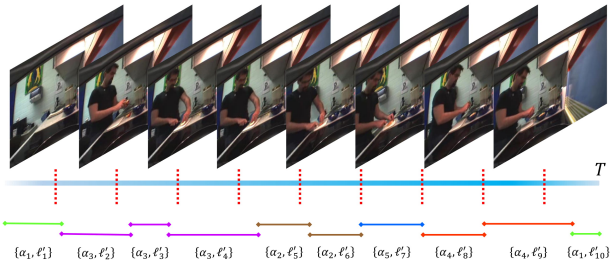

Figure 1. Our model estimates for K temporal regions the actions probabilities $A_{1:K} = (a_1, \ldots, a_K), a_k \in \mathbb{R}^C$, and the temporal lengths of the regions $L_{1:K} = (\ell_1, \ldots, \ell_K), \ell_k \in \mathbb{R}$. In this example, $K = 10$. Since temporal regions are not aligned with the action segments, the model estimates the temporal lengths to refine the corresponding temporal region of the predicted action.

3. Weakly Supervised Action Segmentation

Action segmentation requires to temporally segment all frames of a given video, *i.e.*, predicting the action in each frame of a video. The task can be formulated as follows. Given an input sequence of D-dimensional features $X_{1:T} = (x_1, \ldots, x_T)$, $x_t \in \mathbb{R}^D$, the task is to infer the sequence of framewise action labels $\hat{Y}_{1:T} = (\hat{y}_1, \ldots, \hat{y}_T)$ where there are C classes $\mathcal{C} = \{1, \ldots, C\}$ and $\hat{y}_t \in \mathcal{C}$.

In the case of fully supervised learning, the labels $\hat{Y}_{1:T}$ are provided for each training sequence. In this work, we investigate a weakly supervised setting as in [28]. In this setting, only the actions $\hat{A} = \{\hat{a}_1, \ldots, \hat{a}_M\}$ that occur in a long video are given where $\hat{a}_m \in \mathcal{C}$ and $M \leq C$. In contrast to other weakly supervised settings where transcripts are given, this is a much more difficult task since not only the lengths of the actions are unknown for the training sequences, but also the order of the actions and the number of the occurrences of each action.

4. Proposed Method

In order to address weakly supervised action segmentation, we propose a network that divides a temporal sequence into temporal regions and that estimates for each region the action and the length as illustrated in Figure 1. This representation is between a frame-wise representation where the length of each region is just one frame and an action

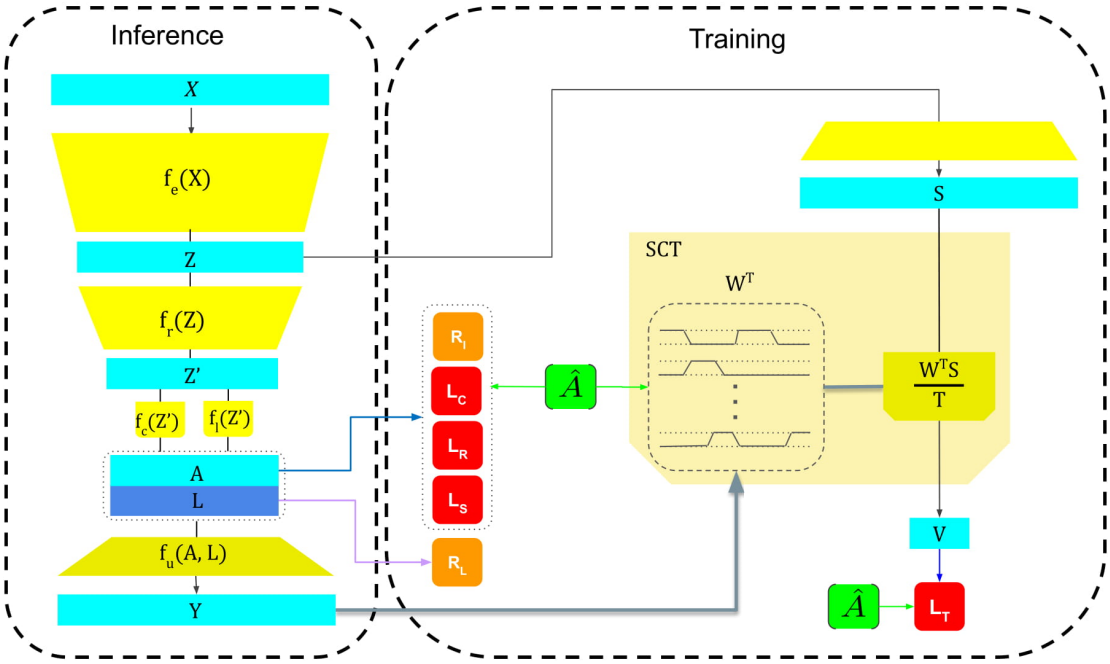

Figure 2. Overview of the proposed network with loss functions. The network gets a sequence of features $X_{1:T}$ as input. A temporal model $f_e(X)$ maps these features to a latent space Z with lower temporal resolution. The lower branch $f_r(Z)$ divides the temporal sequence into temporal regions $Z'_{1:K}$ and estimates for each region the action probabilities a_k and the length l_k. Since the temporal resolution has been decreased, the upsampling module $f_u(A, L)$ uses the lengths $L_{1:K}$ and the action probabilities $A_{1:K}$ of all regions to obtain estimates of the framewise probabilities $Y_{1:T}$. While $L_{1:K}$ is regularized by the length regularizer $\mathcal{R}_\mathcal{L}$, $A_{1:K}$ is trained to minimize $\mathcal{L}_\mathcal{S}$, $\mathcal{L}_\mathcal{R}$, $\mathcal{L}_\mathcal{C}$, and $\mathcal{R}_\mathcal{I}$. Since besides of the regularizer $\mathcal{R}_\mathcal{L}$, there is no loss term that provides supervision for L, we use a second branch $f_s(Z)$ to provide an additional supervisory signal. Using SCT, we transform the temporal representations $Y_{1:T}$ and $S_{1:T}$ to a set representation $V_{1:M}$ for the self supervision loss $\mathcal{L}_\mathcal{T}$.

segment representation where a region contains all neighboring frames that have the same action label.

Figure 2 illustrates our proposed network, which consists of three components. The first component $f_e(X)$, which is described in Section 4.1, maps the input video features $X \in \mathbb{R}^{T\times D}$ to a temporal embedding $Z \in \mathbb{R}^{T'\times D'}$ where $T' < T$ and $D' < D$. The second component $f_r(Z)$, which is described in Section 4.2, takes Z as input and estimates for K temporal regions the actions probabilities $A_{1:K} = (a_1, \ldots, a_K), a_k \in \mathbb{R}^C$, and the temporal lengths of the regions $L_{1:K} = (\ell_1, \ldots, \ell_K), \ell_k \in \mathbb{R}$. In order to obtain the frame-wise class probabilities $Y \in \mathbb{R}^{T\times C}$ from L and A, the third component $f_u(A, L)$, which is discussed in Section 4.3, upsamples the estimated regions such that there are T regions of length 1.

4.1. Temporal Embedding

Given the input video features $X \in \mathbb{R}^{T\times D}$ the temporal embedding component $f_e(X)$ outputs the hidden video representation $Z \in \mathbb{R}^{T'\times D'}$. Our temporal embedding is a fully convolutional network. In this network we first apply a 1-d convolution with kernel size 1 to reduce the input feature dimension from D to D'. On top of this layer we have used B temporal convolution blocks (TCB) with $B = 6$. Each TCB contains a dilated 1-d convolution layer with kernel size 3 for conducting the temporal structure. We increase the dilation rates as $\{2^b | b \in \mathbb{Z}^+, 0 \leq b \leq B\}$ where b is the index of the TCBs. Then a ReLU activation function is applied on top of the convolutional layer. On top of this combination, a 1-d convolution layer with kernel size 1 and a residual connection is used. Finally, a dropout with a probability of 0.25 is applied on top. The TCB is modeled after the WaveNet architecture [36]. To reduce the temporal dimension of the representation, we perform temporal max poolings with a kernel size of 2 on top of the TCBs with $b = \{1, 2, 4\}$. Using the TCBs and max poolings, we get large receptive fields on the input data X. Having such large receptive fields provides the capability of modeling long and short range temporal relations between the input frames.

4.2. Temporal Regions

On top of the temporal embedding, we use the temporal region estimator network $f_r(Z)$ to estimate the action

probabilities and the temporal lengths for K temporal regions. $f_r(Z)$ outputs the hidden representation $Z'_{1:K} = (z'_1, \dots, z'_K), z'_k \in \mathbb{R}^{D'}$, for the temporal regions. To have a better representation for estimating the actions probabilities A and region lengths L, we increase the receptive field size and decrease the temporal dimension. This network mostly follows the same architecture design as $f_e(X)$. It has B' TCBs with $B' = 4$. The dilation rates of the TCBs are set as $\{2^{b'} | b' \in \mathbb{Z}^+, B < b' \leq B + B'\}$. To reduce the temporal dimension of the representation, we perform temporal max poolings with kernel size 2 on top of the TCBs with indices 2 and 4. On top of the final TCB, we have two different heads f_c and f_l. $f_c(Z')$ predicts the class probabilities A. It consists of a 1-d convolution layer with a kernel size of 1 and an output channel size of C. A softmax function is applied on top of the convolution layer to get the action probabilities A. $f_l(Z')$ predicts the lengths L for the corresponding temporal regions. It consists of two 1-d convolution layers with kernel sizes 1 and output channels $D'/2$ and 1, respectively.

4.3. Region Upsampling

$f_c(Z')$ estimates the action probabilities $A_{1:K}$ for temporal regions. To get probabilities for temporal action segmentation, we need to upsample $A_{1:K}$ to $Y_{1:T}$. Since $f_l(Z')$ predicts the corresponding lengths $L_{1:K}$, we can use theses lengths to upsample the probabilities A. To do so, we first project the predicted lengths $L_{1:K}$ to absolute lengths $L'_{1:K} = (\ell'_1, ..., \ell'_K), \ell'_k \in \mathbb{Z}^+$, by:

$$\ell'_k = T \frac{e^{\ell_k}}{\sum_{i=1}^{K} e^{\ell_i}}. \tag{1}$$

In other words, we apply the softmax function on L to get the relative lengths, which sum up to 1 and then multiply them by T to get the absolute lengths. Therefore, the absolute lengths sum up to T, which is our desired final temporal size for Y. Given the absolute lengths L', we upsample A in a differentiable way such that $a_k \in \mathbb{R}^C$ becomes $a'_k \in \mathbb{R}^{\ell'_k \times C}$.

4.3.1 Temporal Sampling

Although it is possible to obtain a'_k by just copying ℓ'_k times the probabilities a_k, this operation is not differentiable with respect to ℓ'_k. However, we need a differentiable operation in order to update the parameters of f_l, which predicts L, during training.

We first generate our target matrix $a'_k \in \mathbb{R}^{H \times C}$ where $H = \max_k \ell'_k$, *i.e.*, the matrix is set such that the size is constant for all k. For a better temporal sampling, we also expand the source by copying j times a_k, where J is a canonical value equal to 100. Although a_k has been expanded to $\mathbb{R}^{J \times C}$, we still keep the notation a_k.

The idea is to fill the matrix a'_k by backward warping and a bilinear kernel. Similar to [13], we use normalized element indices, such that $-1 \leq i_a[j] \leq 1$ when $j \in [1 \dots J]$ and $-1 \leq i_{a'}[h] \leq 1$ when $h \in [1 \dots H]$. This means if we just interpolate the values for each column c, the operation is defined by

$$a'_k[h, c] = \sum_{j=1}^{J} a_k[j, c] \max(0, 1 - |i_{a'}[h] - i_a[j]|) \tag{2}$$

for $h \in [1 \dots H]$.

However, we do not want to fill the entire row but only until ℓ'_k. We therefore apply a 1D affine transformation to the index function

$$T_{\ell'_k}(i_{a'}[h]) = \frac{H}{\ell'_k} i_{a'}[h] + \frac{H}{\ell'_k} - 1. \tag{3}$$

This means that $T_{\ell'_k}(i_{a'}[1]) = -1$ and $T_{\ell'_k}(i_{a'}[\ell'_k]) = 1$. By integrating (3) into (2), we obtain the upsample operation

$$a'_k[h, c] = \sum_{j=1}^{J} a_k[j, c] \max\left(0, 1 - \left|T_{\ell'_k}(i_{a'}[h]) - i_a[j]\right|\right) \tag{4}$$

that is differentiable with respect to ℓ'_k.

Finally, the matrix is cropped to $a'_k \in \mathbb{R}^{\ell'_k \times C}$ and we obtain $Y \in \mathbb{R}^{T \times C}$ by concatenating the a'_ks for $k = 1, \dots, K$.

5. Training

In Section 4 we proposed a model that is capable of dividing a temporal sequence into temporal regions and predicting corresponding action probabilities A and lengths L. We now discuss the loss functions and regularizers for training the model.

5.1. Set Loss

In a set supervised problem we already have the set supervision. So we use a simple set prediction loss $\mathcal{L}_\mathcal{S}$ to use the given set of actions $\hat{A}$. We apply a global max pooling over the temporal dimension of A to output $a^{mc} \in \mathbb{R}^C$. Then we use the binary cross entropy loss for multiclass classification as

$$\mathcal{L}_\mathcal{S} = -\frac{1}{C} \left(\sum_{m \in \hat{A}} \log(a^{mc}[m]) + \sum_{m \notin \hat{A}} \log(1 - a^{mc}[m]) \right). \tag{5}$$

This loss encourages the model to assign at least one region to one of the classes in $\hat{A}$ and none to the other classes.

5.2. Region Loss

The set loss only enforces that there is one region with a high probability for each class. It can, however, happen that

the other regions have a uniform distribution for the classes in $\hat{A}$. Since it is unlikely that all actions occur at the same time, we introduce the region loss, which encourages the model to predict only one action from $\hat{A}$ per region. Since we know that only actions from $\hat{A}$ can occur, we first discard the unrelated actions from $A \in \mathbb{R}^{K \times C}$ and denote it by $A^{\mathcal{S}} \in \mathbb{R}^{K \times M}$, where each column belongs to one of the given action set members $\hat{a}_m$. We now prefer a prediction where for each k the probability is close to 1 for one action $\hat{a}_m$. Due to the softmax, this means that the probability is close to zero for the other actions.

This is achieved by applying a global max pooling over the m dimension of $A^{\mathcal{S}} \in \mathbb{R}^{K \times M}$ to obtain $a^{mk} \in \mathbb{R}^K$ and using the cross entropy loss:

$$\mathcal{L}_{\mathcal{R}} = -\frac{1}{K} \sum_{k=1}^{K} \log \left(a^{mk}[k] \right). \tag{6}$$

5.3. Inverse Sparsity Regularization

The set loss and the region loss ensure that (i) all actions that are not in the set $\hat{A}$ have a low probability, (ii) for each temporal region there is exactly one action $\hat{a}_m \in \hat{A}$ with high probability, and (iii) for each action $\hat{a}_m$ there is at least one region k where $a[k, m]$ is high. This, however, can result in unlikely solutions where for $M - 1$ actions there is only one region with high probability whereas the other regions are assigned to a single action class. To prevent such a sparse distribution of regions for some action classes, we introduce an inverse sparsity regularization term $\mathcal{R}_{\mathcal{I}}$, which prefers a more balanced class distribution averaged over all regions:

$$\mathcal{R}_{\mathcal{I}} = \frac{1}{M} \sum_{m \in \hat{A}} \left(1 - \frac{1}{K} \sum_{k=1}^{K} a[k, m] \right). \tag{7}$$

This regularizer encourages that the action classes compete for maximizing the number of temporal regions they are being predicted for.

5.4. Temporal Consistency Loss

As illustrated in Figure 1, the temporal regions are usually smaller than the action segments in the video and a single action often spans several regions. The likelihood of observing the same action in the neighboring temporal regions is therefore usually higher than observing a different action. We therefore introduce the temporal consistency loss $\mathcal{L}_{\mathcal{C}}$, that encourages the model to predict similar action labels for neighboring temporal regions:

$$\mathcal{L}_{\mathcal{C}} = \frac{1}{M} \sum_{m \in \hat{A}} \frac{1}{K} \sum_{k=2}^{K} |a[k, m] - a[k-1, m]|. \tag{8}$$

More precisely, $\mathcal{L}_{\mathcal{C}}$ encourages the model to have less prediction changes over the temporal dimension of A^S.

5.5. Self Supervision Loss

The aforementioned losses and regularizers only affect the class probabilities A of the temporal regions, but do not backpropagate gradients through the subnetwork f_l. This means that the network does not learn the corresponding lengths of the regions during training. In order to provide an auxiliary supervision signal to train L, we employ a self supervision technique which relies on using two different representations. The first representation Y is obtained by estimating the actions probabilities A and lengths L for K temporal regions as described in Section 4.2. Due to the temporal sampling, the representation Y is differentiable with respect to L.

To have another representation, we use a second branch $f_s(Z)$. This branch consists of a subnetwork that has a single 1-d convolution with kernel size 1 and output channel size C. It predicts frame-wise class probabilities for the temporal size T'. We linearly interpolate it to $S \in \mathbb{R}^{T \times C}$ along the temporal dimension, which corresponds to a setting where $K = T'$ and $\ell_k = \frac{T}{T'}$, *i.e.*, all regions have a constant length.

Since we do not know the ground-truth lengths L but only the set of present actions $\hat{A}$, we combine Y and S to compute class probabilities $V_{1:M} = (v_1, ...v_M), v_m \in \mathbb{R}^C$, for each element in the set $\hat{A}$. This is done by the Set Constrained Temporal Transformer module (SCT).

5.6. Set Constrained Temporal Transformer

As it is illustrated in Figure 2, we produce for each action class $\hat{a}_m \in \hat{A}$ masks w_m from Y. The masks indicate the temporal locations where the action $\hat{a}_m$ occurs in the video. We use these masks to sample from S:

$$v_m = \frac{1}{T} \sum_{t=1}^{T} w_m[t] S[t]. \tag{9}$$

If S and Y are consistent, $v_m[\hat{a}_m]$ should be high and $v_m[\hat{a}_n]$ should be close to zero for $n \neq m$.

To exploit this, we apply a softmax on v_m to get the predicted probabilities for the given action and use the cross entropy loss:

$$\mathcal{L}_{\mathcal{T}_m}(v_m, \hat{a}_m) = -\log \left(\frac{e^{v_m[\hat{a}_m]}}{\sum_{c=1}^{C} e^{v_m[c]}} \right). \tag{10}$$

Since w_m is differentiable with respect to a_m and l_m, the loss affects both.

As a more efficient way, we can apply all of the masks W on S using:

$$V = \frac{W^T S}{T} \tag{11}$$

where $V \in \mathbb{R}^{M \times C}$ and $W^T \in \mathbb{R}^{M \times T}$ denotes the transposed version of W. Therefore, we can define the loss for

V and the given actions set $\hat{A}$ as

$$\mathcal{L}_{\mathcal{T}}(V, \hat{A}) = -\frac{1}{M}\sum_{m=1}^{M} \log\left(\frac{e^{V[m,\hat{a}_m]}}{\sum_{c=1}^{C} e^{V[m,c]}}\right). \quad (12)$$

5.6.1 Backpropagation

Using the $\mathcal{L}_{\mathcal{T}}$ loss, the gradient can backpropagate through both S and Y. Y is the output of $f_u(A, L)$ which is a differential function over L and A. Therefore, we can update the f_l weights using the backpropagated gradients. To be able to backpropagate through the a'_ks we define the gradients with respect to the sampling indices $T_{\ell'_k}(i_{a'}[h])$ as

$$\frac{\partial a'_k[h,c]}{\partial T_{\ell'_k}(i_{a'}[h])} = \sum_{j=1}^{J} a_k[j,c] \begin{cases} 0 & |i_a[j] - T_{\ell'_k}(i_{a'}[h])| \geq 1 \\ 1 & i_a[j] - 1 < T_{\ell'_k}(i_{a'}[h]) \leq i_a[j] \\ -1 & i_a[j] < T_{\ell'_k}(i_{a'}[h]) < i_a[j] + 1 \end{cases}. \quad (13)$$

Since the sampling indices $T_{\ell'_k}(i_{a'}[h])$ are a function of the predicted lengths $L_{1:M}$, the loss gradients are backpropagated to the predicted lengths

5.6.2 Region Length Regularization

Learning the lengths based on $\mathcal{L}_{\mathcal{T}}$ may result in degenerated lengths which are close to zero. Therefore, we use a length regularizer $\mathcal{R}_{\mathcal{L}}$ to prevent such circumstances. We define $\mathcal{R}_{\mathcal{L}}$ as

$$\mathcal{R}_{\mathcal{L}} = \frac{1}{K}\sum_{t=1}^{K}(ReLU(\ell_t - \delta) + ReLU(-\ell_t - \delta)) \quad (14)$$

where δ is a canonical value equal to 1. This regularization term penalizes the lengths which are bigger or smaller than the length width of δ.

5.7. Overall Loss

All of the loss functions and regularizers that we mentioned in this section encourage the model to exploit the given weak supervision and also characteristics of actions to train the model for action segmentation. Therefore, the final loss function for the model is the weighted sum of the above mentioned losses and regularizers. In Section 6 we study the impact of all loss functions and regularizers.

6. Experiments

In this section, we analyze the components of our approach. We first analyze the model design. Then we evaluate the effect of using different loss functions and regularizers. Finally, we compare our method with the state-of-the-art.

6.1. Setup

Datasets. We evaluate our method on three popular datasets, namely the Breakfast dataset [16], Hollywood Extended [1], and MPII Cooking 2 [32].

The **Breakfast** dataset contains $1,712$ videos of different cooking activities, corresponding to about 67 hours of videos and 3.6 million frames. The videos belong to 10 different types of breakfast activities like *fried egg* or *coffee* which consist of 48 different fine-grained actions. The actions are densely annotated and only 7% of the frames are background. We report the average frame accuracy (MoF) metric over the predefined train/test splits following [28].

Hollywood Extended contains 937 video sequences with roughly $800,000$ frames. About 61% of the frames are background, which is comparably large compared to other datasets. The videos contain 16 different action classes. We report the Jaccard index (intersection over union) metric over the predefined train/test splits following [28].

MPII 2 Cooking consists of 273 videos with about 2.8 million frames. We use the 67 action classes without object annotations. About 29% of the frames of this dataset are background. The dataset provides a fixed split into a train and test set, separating 220 videos for training. For evaluation, we use the midpoint hit criterion following [31].

Feature Extraction. We use RGB+flow I3D [3] features extracted from the I3D network pretrained on the Kinetics400 dataset [15]. The I3D features are extracted for each frame. Moreover, for a fair comparison we also use the same IDT features as [28] and evaluate the effect of using different features.

Implementation Details. We train all modules of our network together. The hidden size of the temporal embedding module is 128. We use SGD optimizer with weight decay 0.005. The initial learning rate is set to 0.01. Additional details and code are available online.[1]

6.2. Ablation Experiments

In this section we first analyze the model design. Then we analyze the effect of our loss functions and regularizers on training the model.

6.2.1 Effect of different downsampling levels

As mentioned in Section 4.1, we downsample the input by applying temporal max pooling with kernel size 2. We evaluate the effect of downsampling by changing the numbers of temporal max pooling operations in the temporal embedding module. It should be mentioned that we apply each max pooling on top of each temporal convolution block (TCB). As can be seen in Table 1, a small number of max

[1] http://mohsenfayyaz89.github.io

Figure 3. Comparing the segmentation quality of our method to the ActionSet method. Our method has predicted the right order of actions occurring in this video. Our approach also estimates the actions lengths better.

#max poolings	0	1	2	3	4	5	6
MoF	12.3	15.4	20.8	28.1	27.2	21.3	18.2

Table 1. Evaluating the effect of changing the numbers of max pooling operations in the temporal embedding module. Experiments are run on Breakfast split 1.

pooling operations results in a relatively low frame-wise accuracy. This is due to high number of temporal regions, which may result in an over-segmentation problem. Furthermore, a drop in performance can be observed when the number of max pooling operations is too large. In this case, there are not enough temporal regions and a temporal region covers multiple actions. For the rest of the experiments, we use 3 max pooling operations in our temporal modeling module. It should be noted that we use 3 max pooling operations after the TCBs with indices $\{1, 2, 4\}$, while in this experiment 3 max pooling operations are applied after the TCBs with indices $\{1, 2, 3\}$.

6.2.2 Effect of using different loss functions and regularizers

As mentioned in Section 5, we use different loss functions and regularizers to train our model. To quantify the effect of using these loss functions and regularizers, we train our model with different settings in which we can evaluate the effect of them for training. We train our model on split 1 of the Breakfast dataset and report the results in Table 2.

As mentioned in Section 5.1, the **Set Loss** $\mathcal{L}_{\mathcal{S}}$ encourages the model to have at least one temporal region with a high class probability for each action in the target action set. Therefore, this loss does not affect the frame-wise prediction performance of the model. As it can be seen in Table 2, using only $\mathcal{L}_{\mathcal{S}}$ the model achieves MoF of 8.1%.

By adding the **Region Loss** $\mathcal{L}_{\mathcal{R}}$, we encourage the model to only predict one action per temporal region. Using this auxiliary supervision, the MoF slightly improves to 9.9% which is still relatively low.

As mentioned in Section 5.3, adding the **Inverse Sparsity Regularizer** $\mathcal{R}_{\mathcal{I}}$ helps the method to prevent a sparse distribution of regions for some action classes. Therefore, by adding $\mathcal{R}_{\mathcal{I}}$ to the overall loss, the MoF improves to 19.2%, which is significantly better than predicting every frame as background which covers about 7% of the frames.

We further add the **Temporal Consistency Loss** $\mathcal{L}_{\mathcal{C}}$ to encourage the model to predict similar actions for neighboring temporal regions. $\mathcal{L}_{\mathcal{C}}$ improves the result to 21.9%.

As mentioned in Section 5.5, all of the aforementioned losses and regularizers only affect the class probabilities A of the temporal regions and do not backpropagate gradients through the length estimator head f_l. We therefore add the **Self Supervision Loss** $\mathcal{L}_{\mathcal{T}}$ to evaluate the effect of learning lengths during training. Adding $\mathcal{L}_{\mathcal{T}}$ significantly improves the accuracy to 29.9%. This improvement shows the effect of refining the temporal regions using the predicted lengths.

We also evaluate the effect of using the **Region Length Regularization** $\mathcal{R}_{\mathcal{L}}$. As mentioned in Section 5.6.2, learning the lengths only based on $\mathcal{L}_{\mathcal{T}}$ may result in too diverse estimated lengths for temporal regions. Therefore, we evaluate the effect of $\mathcal{R}_{\mathcal{L}}$ by adding it to the overall loss. By adding this regularizer the accuracy improves to 30.8%. Since $\mathcal{L}_{\mathcal{T}}$ and $\mathcal{R}_{\mathcal{L}}$ are the only loss function and regularizer which affect the lengths L, we also evaluate the effect of only using $\mathcal{R}_{\mathcal{L}}$ as an effective regularizer on the lengths without $\mathcal{L}_{\mathcal{T}}$. This setting results in an MoF of 22.2%. The reason for such a significant drop in performance is that $\mathcal{R}_{\mathcal{L}}$ only encourages the model to not estimate too diverse lengths. This shows that the proposed self supervision loss based on the set constrained temporal transformer is important to learn proper lengths for the temporal regions.

To have a better understanding of the **Self Supervision**

$\mathcal{L}_S$	$\mathcal{L}_R$	$\mathcal{R}_I$	$\mathcal{L}_C$	$\mathcal{L}_T$	$\mathcal{R}_L$	$\mathcal{L}_J$	MoF
✓	-	-	-	-	-	-	8.1
✓	✓	-	-	-	-	-	9.9
✓	✓	✓	-	-	-	-	19.2
✓	✓	✓	✓	-	-	-	21.9
✓	✓	✓	✓	✓	-	-	29.9
✓	✓	✓	✓	✓	✓	-	**30.8**
✓	✓	✓	✓	-	✓	-	22.2
✓	✓	✓	✓	-	✓	✓	25.3

Table 2. Evaluating the effect of using different losses and regularizers. Experiments are run on Breakfast split 1.

	I3D	IDT
ActionSet [28]	20.1*	23.3
Ours	**30.4**	**26.6**

Table 3. Comparison of our method to [28] for different features. Experiments are run on the Breakfast dataset and MoF is reported. Our method achieves state-of-the-art results using both types of features. *The source code of the paper has been used for this experiment.

Loss, we also try to train the temporal regions' length estimator head f_l in a different way. Instead of using $\mathcal{L}_\mathcal{T}$, we use the Jensen Shannon Divergence loss which is a symmetric and smoothed version of the Kullback–Leibler divergence to match both representations Y and S as follows:

$$\mathcal{L}_\mathcal{J} = \frac{1}{2} D(Y \parallel M) + D(S \parallel M), \tag{15}$$

$$\mathcal{D}(\mathcal{P} \parallel \mathcal{Q}) = \sum_{x \in X} P(x) \log(\frac{P(x)}{Q(x)}) \tag{16}$$

where $M = \frac{1}{2}(Y + S)$. Using $\mathcal{L}_\mathcal{J}$ instead of $\mathcal{L}_\mathcal{T}$ results in an MoF of 25.3%, which shows the superiority of our self supervision loss.

6.2.3 Effect of using different features

As mentioned before, we use I3D features [3] as input to our model. To evaluate the effect of the input video features, we train our model using IDT features as well and report the results in Table 3. To have a better comparison with the previous state-of-the-art method [28], we also train this method with I3D features using the publicly available code. We observe that our method achieves state-of-the-art results using both types of features. The ActionSet [28] method does not perform well on I3D features which may be due to the limitations in its temporal architecture design that is not capable of learning a proper temporal embedding over the I3D features.

6.3. Comparison to State-of-the-Art

The task of learning temporal action segmentation using action sets as weak supervision has been so far addressed only by [28]. We compare our approach to this method on three datasets. As it can be seen in Table 4, our method achieves state-of-the-art results on all three datasets. While the methods with only set supervision work well on Breakfast and Hollywood Extended, the performance on the Cooking 2 dataset is lower. The Cooking 2 dataset has a high number of classes (67) while having a low number of training samples (220). The other problem is that this dataset contains very long videos which on average contain 50 different actions. Therefore, learning a temporal action segmentation model on this dataset using only weak set supervision is very difficult. In order to get a better understanding of the effect of such characteristics of this dataset, we have evaluated the effect of cutting this dataset into different parts followed by [28]. As it can be seen in Table 5, having more videos for training and fewer actions per video on average improves the results. Figure 3 shows a qualitative result of our method for a video from the Breakfast dataset.

Dataset	Break Fast	Holl. Ext.	Cooking 2
	MoF	*jacc. idx*	*midpoint hit*
ActionSet-monte-carlo [28]	23.3	9.3	9.8
ActionSet-text-based [28]	23.2	9.2	10.6
Ours	**30.4**	**17.7**	**14.3**

Table 4. Comparison of our method to [28] for weakly supervised temporal segmentation.

cuts per video	4	2	-
avg. actions per video	12.5	25	50
ActionSet [28]	17.4	12.1	9.8
Ours	**19.8**	**16.1**	**14.3**

Table 5. Different levels of video trimming for Cooking 2. More videos and fewer actions per video result in better performance.

7. Conclusion

In this work we presented a network for temporal action segmentation. The network is trained on long videos which are only annotated by the set of present actions. The network is trained by dividing the videos into temporal regions that contain only one action class and are consistent with the set of annotated actions. We thoroughly evaluated the approach on three datasets. For all three datasets, the proposed network outperforms previous work.

Acknowledgment The work has been funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) GA 1927/4-1 (FOR 2535 Anticipating Human Behavior) and the ERC Starting Grant ARCA (677650).

References

[1] Piotr Bojanowski, Rémi Lajugie, Francis Bach, Ivan Laptev, Jean Ponce, Cordelia Schmid, and Josef Sivic. Weakly supervised action labeling in videos under ordering constraints. In *ECCV*, 2014.

[2] Fabian Caba Heilbron, Victor Escorcia, Bernard Ghanem, and Juan Carlos Niebles. Activitynet: A large-scale video benchmark for human activity understanding. In *CVPR*, 2015.

[3] Joao Carreira and Andrew Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *CVPR*, 2017.

[4] Chien-Yi Chang, De-An Huang, Yanan Sui, Li Fei-Fei, and Juan Carlos Niebles. D^3tw: Discriminative differentiable dynamic time warping for weakly supervised action alignment and segmentation. *arXiv*, 2019.

[5] Ali Diba, Mohsen Fayyaz, Vivek Sharma, A. Hossein Karami, M. Mahdi Arzani, Rahman Yousefzadeh, and Luc Van Gool. Temporal 3d convnets using temporal transition layer. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, 2018.

[6] Ali Diba, Mohsen Fayyaz, Vivek Sharma, M. Mahdi Arzani, Rahman Yousefzadeh, Juergen Gall, and Luc Van Gool. Spatio-temporal channel correlation networks for action classification. In *ECCV*, 2018.

[7] Ali Diba, Mohsen Fayyaz, Vivek Sharma, Paluri. Manohar, Juergen Gall, Rainer Stiefelhagen, and Luc Van Gool. Large scale holistic video understanding. *arXiv*, 2019.

[8] Li Ding and Chenliang Xu. Weakly-supervised action segmentation with iterative soft boundary assignment. In *CVPR*, 2018.

[9] O. Duchenne, I. Laptev, J. Sivic, F. Bach, and J. Ponce. Automatic annotation of human actions in video. In *2009 IEEE 12th International Conference on Computer Vision*, 2009.

[10] Yazan Abu Farha and Juergen Gall. Ms-tcn: Multi-stage temporal convolutional network for action segmentation. In *CVPR*, 2019.

[11] Christoph Feichtenhofer, Haoqi Fan, Jitendra Malik, and Kaiming He. Slowfast networks for video recognition. In *ICCV*, 2019.

[12] De-An Huang, Li Fei-Fei, and Juan Carlos Niebles. Connectionist temporal modeling for weakly supervised action labeling. In *ECCV*, 2016.

[13] Max Jaderberg, Karen Simonyan, Andrew Zisserman, et al. Spatial transformer networks. In *NIPS*, 2015.

[14] Svebor Karaman, Lorenzo Seidenari, and Alberto Del Bimbo. Fast saliency based pooling of fisher encoded dense trajectories. In *ECCV THUMOS Workshop*, 2014.

[15] Will Kay, Joao Carreira, Karen Simonyan, Brian Zhang, Chloe Hillier, Sudheendra Vijayanarasimhan, Fabio Viola, Tim Green, Trevor Back, Paul Natsev, Mustafa Suleyman, and Andrew Zisserman. The kinetics human action video dataset. *arXiv:1705.06950*, 2017.

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

[17] Hilde Kuehne, Juergen Gall, and Thomas Serre. An end-to-end generative framework for video segmentation and recognition. In *WACV*, 2016.

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

[19] Hildegard Kuehne, Hueihan Jhuang, Estíbaliz Garrote, Tomaso Poggio, and Thomas Serre. Hmdb: a large video database for human motion recognition. In *ICCV*, 2011.

[20] Hilde Kuehne, Alexander Richard, and Juergen Gall. Weakly supervised learning of actions from transcripts. *CVIU*, 2017.

[21] I. Laptev, M. Marszalek, C. Schmid, and B. Rozenfeld. Learning realistic human actions from movies. In *2008 IEEE Conference on Computer Vision and Pattern Recognition*, 2008.

[22] Colin Lea, Michael D Flynn, Rene Vidal, Austin Reiter, and Gregory D Hager. Temporal convolutional networks for action segmentation and detection. In *CVPR*, 2017.

[23] Colin Lea, Austin Reiter, René Vidal, and Gregory D Hager. Segmental spatiotemporal cnns for fine-grained action segmentation. In *ECCV*, 2016.

[24] Jun Li, Peng Lei, and Sinisa Todorovic. Weakly supervised energy-based learning for action segmentation. In *ICCV*, 2019.

[25] H. Pirsiavash and D. Ramanan. Parsing videos of actions with segmental grammars. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, 2014.

[26] Alexander Richard and Juergen Gall. Temporal action detection using a statistical language model. In *CVPR*, 2016.

[27] Alexander Richard, Hilde Kuehne, and Juergen Gall. Weakly supervised action learning with rnn based fine-to-coarse modeling. In *CVPR*, 2017.

[28] Alexander Richard, Hilde Kuehne, and Juergen Gall. Action sets: Weakly supervised action segmentation without ordering constraints. In *CVPR*, 2018.

[29] Alexander Richard, Hilde Kuehne, Ahsan Iqbal, and Juergen Gall. Neuralnetwork-viterbi: A framework for weakly supervised video learning. In *CVPR*, 2018.

[30] Marcus Rohrbach, Sikandar Amin, Mykhaylo Andriluka, and Bernt Schiele. A database for fine grained activity detection of cooking activities. In *CVPR*, 2012.

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

[32] Marcus Rohrbach, Anna Rohrbach, Michaela Regneri, Sikandar Amin, Mykhaylo Andriluka, Manfred Pinkal, and Bernt Schiele. Recognizing fine-grained and composite activities using hand-centric features and script data. *Int. J. Comput. Vision*, 2016.

[33] Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah. Ucf101: A dataset of 101 human actions classes from videos in the wild. *arXiv:1212.0402*, 2012.

[34] Yasser Souri, Mohsen Fayyaz, and Juergen Gall. Weakly Supervised Action Segmentation Using Mutual Consistency. *arXiv*, 2019.

[35] Du Tran, Heng Wang, Lorenzo Torresani, and Matt Feiszli. Video classification with channel-separated convolutional networks. In *ICCV*, 2019.

[36] 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. In *SSW*, 2016.

[37] N. N. Vo and A. F. Bobick. From stochastic grammar to bayes network: Probabilistic parsing of complex activity. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, 2014.

[38] Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. In *CVPR*, 2018.

[39] Yue Zhao, Yuanjun Xiong, Limin Wang, Zhirong Wu, Xiaoou Tang, and Dahua Lin. Temporal action detection with structured segment networks. In *ICCV*, 2017.