Fast Weakly Supervised Action Segmentation Using Mutual Consistency

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

Abstract

Action segmentation is the task of predicting the actions in each frame of a video. Because of the high cost of preparing training videos with full supervision for action segmentation, weakly supervised approaches which are able to learn only from transcripts are very appealing. In this paper, we propose a novel end-to-end approach for weakly supervised action segmentation based on a two branch network. The two branches of our network predict two redundant but different representations for action segmentation. During training we introduce a new mutual consistency loss (MuCon) that enforces that these two representations are consistent. Using MuCon and a transcript prediction loss, our network achieves performance competitive or better than state-of-the-art while being fully differentiable, 14 *times faster to train and* 20 *times faster at test time.*

1. Introduction

The production, availability, and consumption of video data is increasing everyday with an exponential rate. With such growth comes the need to analyze, monitor, annotate and learn from this huge amount of often publicly available data. The computer vision community has therefore shown great interest in video data and approaches for action recognition on trimmed video clips have shown great results in recent years [25, 3, 6, 30, 9]. Although these results on trimmed video clips are important, there is also a need for methods that temporally segment untrimmed videos that often contain multiple actions with different lengths [16, 8, 18, 32] .

Since annotating exact temporal boundaries of actions in long videos is very cumbersome and costly, methods that can utilize weaker types of supervision are of special interest. A popular type of weak supervision are transcripts [17, 11, 22, 23, 7, 4, 20], which provide for each video in the training set, an ordered list of actions. To learn a model from transcripts, previous approaches try to align the transcripts to the training videos, *i.e.*, they infer a frame-wise labeling of each training video based on the provided transcripts. For the transcript alignment, the Viterbi algorithm is commonly used. The Viterbi algorithm takes the frame-wise class probabilities of a video, which are estimated by the intermediate model, to estimate a path of frame labels that does not violate the action order of the given transcript. While [22, 7] perform the alignment after each epoch for all training videos, [23, 20] apply it at each iteration to a single video. Since the alignment using the Viterbi algorithm as in [22, 23, 20] is non-differentiable, a differentiable form of dynamic time warping has been proposed in [4].

These approaches, however, have two major limitations that we address by our method. The first issue is the restriction of the solution space to transcripts that have been observed in the training set, because these approaches are not trained to predict the transcript. In order to perform action segmentation on a test video, the methods perform alignment for all transcripts from the training set and choose the one that best aligns to the test video according to their model. This means that an action order that never occurred in the training data cannot be recovered. The second issue is the computation time for training and inference. During training, dynamic programming is performed several times for each training video which increases the training time of a network substantially.

In order to address these two limitations, we propose an approach for weakly supervised action segmentation that does not require costly dynamic programming and Viterbi decoding at training time. Furthermore because the network is trained to predict the transcript, at test time it is 20 times faster and not limited to the set of previously observed transcripts. The network consists of two branches. The first branch predicts the transcript for a video and we can directly apply a transcript prediction loss that enforces that the predicted transcript matches the ground-truth transcript. In addition, the branch also predicts the length of each action in the transcript, which provides directly an alignment of the predicted transcript to the video. Since we do not know

* Yaser Souri and Mohsen Fayyaz contributed equally to this work.

the ground-truth alignment of the transcripts in the training data, we have to introduce an additional supervisory signal. To this end, we add a second branch which predicts frame-wise class probabilities instead of a transcript and the lengths of the actions. We then exploit the fact that both branches predict redundant representations, which should yield the same action segmentation. This is achieved by adding a new differentiable Mutual Consistency (MuCon) loss that enforces that theses two representations are mutually consistent and match each other.

Our method achieves statistically the accuracy of current state-of-the-art methods on standard benchmarks like the Breakfast [15] and the Hollywood Extended [2] datasets, but it is 14 times faster to train and more importantly 20 times faster during testing. This means that our model provides by far the best trade-off between accuracy and speed.

To summarize our contributions are three folds. First, we introduce a novel, fast and end-to-end method for weakly supervised action segmentation and alignment. Second, we report results statistically similar to the state-of-the-art while being significantly faster to train and test. Third, we perform a thorough evaluation of the methods for weakly supervised action segmentation that have their code publicly available and provide new insights.

2. Related Work

Action segmentation in videos has already been addressed in many works [16, 18, 32, 8]. Previously, action segmentation approaches relied on multi-scale sliding window processing [24, 13] or Markov models on top of frame-classifiers [16, 19]. These approaches where typically slow at inference time. Newer approaches for fully supervised action segmentation try to capture the long range temporal dependencies using temporal convolutions [8, 18].

A variety of different approaches for weakly supervised action segmentation has already been proposed for video data. Bojanowski et al. [2] introduced the Hollywood extended dataset and proposed a method based on discriminative clustering for the task of action alignment. Huang et al. [11] have proposed to use an extended version of the CTC loss where they take into account frame similarity for frames of the same action. Inspired by methods that are used in speech processing, Kuehne et al. [17] proposed a HMM-GMM based system that iteratively generates pseudo ground truth for videos at the start of each epoch and refines the pseudo ground truth at the end of the epoch. Richard et al. [22] builds on this work by replacing the GMM with an RNN for short range temporal modelling. But their method is still an iterative method that involves per epoch pseudo ground truth generation. The approach has been further improved by Ding and Xu [7]. Almost all of these methods rely on iterative approaches with two-step optimization which does not allow for direct, end-to-end training. They also involve time consuming dynamic programming steps during training. Richard et al. [23] introduced the Neural Network Viterbi (NNV) method. This method solves some of the problems of previous works, but it is still expensive to train due to Viterbi decoding and cannot generate unseen transcripts during testing. They use a global length model for actions, which is updated heuristically during training. Recently, Li et al. [20] introduced an extension of NNV which performs more accurately on standard benchmarks. They introduce a new constrained discriminative loss that discriminates between the energy of valid and invalid segmentations of the training videos. This results in large improvement in performance compared to NNV. But CDFL still has the same drawbacks as NNV.

Our work is also related to methods that perform sequence to sequence learning. There has been a lot of work in sequence to sequence learning mostly in natural language processing[27, 1, 10, 29] but our problem is different in one major aspect. The size mismatch between the video features and transcripts are very large. For example in the Breakfast dataset [15], the average input video length is around 2100 frames and the longest video has around 9700 frames while the average transcript length is 6.8 and the longest is 25. Because of this issue, we have specially designed our network to be able to temporally model such long sequences.

After the introduction of WaveNet [28], Abu Farha and Gall [8] introduced the MS-TCN model and showed that a non-causal variant of WaveNet networks are fully capable of temporally modelling untrimmed and long video sequences if trained with full supervision. We use a variant of MS-TCN [8] as part of the backbone of our network to help with temporal modelling of the input sequence and we train our network using weak supervision.

Wei et al. [31] proposed a method for segment detection in videos. Although their method is similar in spirit to ours the models are different and they are only able to train their model using full supervision.

3. Weakly Supervised Action Segmentation

Action segmentation is the task of predicting the action in each frame of a video, *i.e.*, temporally segmenting the video. More formally, given an input sequence of D-dimensional frame features $X_{1:T} = [x_1, \dots, x_T]$, $x_t \in \mathbb{R}^D$, the goal is to predict the output sequence of framewise action labels $\hat{Y}_{1:T} = [\hat{y}_1, \dots, \hat{y}_T]$ where there are N classes $\mathcal{C} = \{1, \dots, N\}$ and $\hat{y}_t \in \mathcal{C}$. The framewise action labels $\hat{Y}_{1:T}$ can also be represented as an ordered sequence of M segments $\hat{S}_{1:M}$ where each segment $\hat{s}_m$ is defined as an action label $\hat{a}_m \in \mathcal{C}$ and its corresponding length $\hat{\ell}_m \in \mathbb{R}_+$.

In fully supervised action segmentation, the target labels for every given frame $\hat{y}_t$ are known. However, in weakly supervised action segmentation, we only have the ordered sequence of actions $\hat{A}_{1:M} = [\hat{a}_1, \dots, \hat{a}_m]$ also termed as

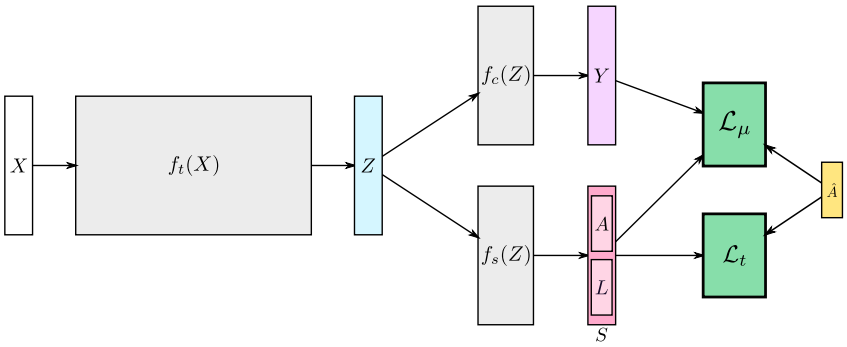

Figure 1. Our proposed network consists of three main modules. The temporal modelling module f_t converts the input features to the hidden representation Z which is used for two branches. While the frame classification module f_c predicts framewise class probabilities for action segmentation, the segment generation module f_s predicts the segment representation for action segmentation. We train our network using two losses. While the transcript prediction loss $\mathcal{L}_t$ enforces that the predicted transcript A matches the ground-truth transcript $\hat{A}$, our proposed mutual consistency (MuCon) loss $\mathcal{L}_\mu$ enforces that the two redundant representations Y and S are consistent.

transcript of a video, during training while the lengths of actions $\hat{L}_{1:M} = [\hat{\ell}_1, \ldots, \hat{\ell}_M]$ are unknown.

The two target representations $\hat{Y}_{1:T}$ and $\hat{S}_{1:M}$ for action segmentation are redundant and it is possible to generate one given the other. In our method we exploit this redundancy by predicting both representations and enforcing them to match each other as mutual supervisory signal.

4. Proposed Method

During training we are given the input video features $X_{1:T}$ and its corresponding target weak labels $\hat{A}_{1:M}$, which is the transcript of a training video. At test time two tasks are defined. For action alignment the ordered sequence of actions $\hat{A}_{1:M}$ is also given for a test video and the task is to predict only the temporal boundaries of the given actions, while for action segmentation both the ordered sequence of actions and their temporal boundaries need to be predicted.

Since we have only weak labels in form of the transcripts for training, we propose a) to use the weak labels directly for training and b) exploit the redundant representation discussed in Section 3. For the first part, transcript prediction part of our network has to predict the sequence of actions $\hat{A}_{1:M}$ that occur in a training video given the sequence of input features $X_{1:T}$. Because of the ability of our model to predict the transcript at test time we are able to perform decoding using the predicted transcript which is two orders of magnitude faster than methods that have to find the best matching transcript from the set of training transcripts. For the second part, our network predicts the two redundant representations for action segmentation and we enforce that the two predicted representations match each other. We term this approach mutual consistency learning.

Our proposed network for mutual consistency learning is illustrated in Figure 1 and it consists of three main modules. First, a temporal modelling module $f_t(X)$ outputs for the input video features $X \in \mathbb{R}^{T \times D}$ the hidden video representation $Z \in \mathbb{R}^{T' \times D'}$ with smaller temporal resolution due to temporal pooling and hidden dimension D'. We design the temporal modelling module which is the base of our network, using a non-causal variant of WaveNet [28] which is similar to MS-TCN [8]. Second, a frame classification module $f_c(Z)$ estimates the class probabilities of each frame $Y \in \mathbb{R}^{T \times N}$ given Z. Third, a segment generation module $f_s(Z)$ predicts the segment representation S and it consists of a bi-directional LSTM encoder and a LSTM decoder with attention [1]. Because of the design of the temporal modelling module, the receptive field of each element z_t is very large which is important for temporal modelling. This hidden video representation is shared between f_c and f_s which each predict one of the redundant representations for action segmentation Y and S, respectively. The training loss in our proposed method consists of two loss functions, namely the transcript prediction loss $\mathcal{L}_t$ and our proposed mutual consistency (MuCon) loss $\mathcal{L}_\mu$. We now describe each part of our proposed method in detail.

4.1. Temporal Modelling Module

This subnetwork consists of a set of 1-dimensional dilated convolutional layers with increasing dilation sizes modelled after the WaveNet architecture [28]. More specifically, we first apply a 1-d convolution with kernel size 1 to perform dimension reduction. Then a set of 11 WaveNet layers $(f_{w_i}; i = 0, \ldots, 10)$ with increasing dilations are applied. Each WaveNet layer consists of a dilated 1-d convolution with non-causal kernel of size of 3, ReLu nonlinearity followed by a 1-d convolution with kernel size 1, a residual connection and dropout with probability 0.25. The amount of dilation for each WaveNet layer f_{w_i} is 2^i. We perform temporal max pooling with kernel size of 2 after

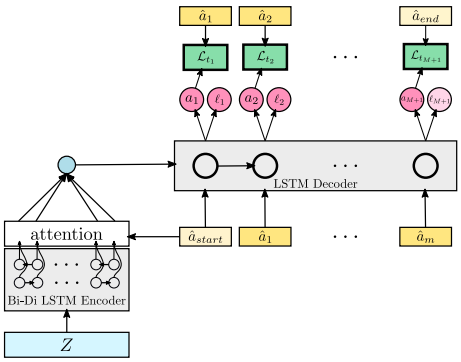

Figure 2. Overview of the segment generation module f_s and the transcript prediction loss $\mathcal{L}_t$. Given the hidden video representation Z, we use a bidirectional LSTM encoder to encode it. Our decoder is a LSTM recurrent neural network with attention. Using a special starting symbol $\hat{a}_{start}$, we start the decoding process and our decoder predicts the action and length of the first segment. While the transcript prediction loss $\mathcal{L}_t$ compares the predicted action class probabilities a_m with the ground-truth action label $\hat{a}_m$, the predicted lengths ℓ_m are evaluated by the mutual consistency loss, which is depicted in Figure 3. The last predicted segment indicates the end of the segment generation and should have a high probability for the special end symbol $\hat{a}_{end}$. We use teacher forcing during training which means that we do not sample from the predicted action labels a_m to feed the decoder, but we use the ground truth action labels $\hat{a}_m$.

some WaveNet layers. At the end, a final 1-d convolution with kernel size 1 is applied to transform the output to the desired output dimension D'. Overall the temporal modelling module takes as input video features $X \in \mathbb{R}^{T \times D}$ and outputs the hidden video representation $Z \in \mathbb{R}^{T' \times D'}$.

4.2. Frame Classification Module

On top of the hidden video representation Z, we perform classification of every frame. This corresponds to a single 1-d convolution with kernel size 1 which takes as input $Z \in \mathbb{R}^{T' \times D'}$ and outputs $Y' \in \mathbb{R}^{T' \times N}$ where N is the number of actions in our dataset. Because of temporal pooling in our temporal modelling module, the output Y' has a lower temporal resolution than our input. To compensate for this after the convolution, we linearly interpolate Y' to $Y \in \mathbb{R}^{T \times N}$ along the temporal dimension.

4.3. Segment Generation Module

The second branch on top of Z is the segment generation module which predicts the segments S. Each segment s_m consists of predicted action probabilities a_m and the relative length ℓ_m of that segment. We will discuss in Section 4.4 how the relative length is mapped to the absolute length. The module and the transcript prediction loss $\mathcal{L}_t$ are illustrated in Figure 2.

We employ a conventional sequence to sequence network with attention [1]. Given the hidden video representation Z, we use a bidirectional LSTM encoder to encode it. Our decoder is a LSTM recurrent neural network with MLP attention. Although these networks on their own struggle to learn a temporal model for long input sequences [5, 26], we address this issue by the temporal modelling module that encodes temporal relations in the more compact hidden video representation Z.

The decoding starts with a special starting symbol $\hat{a}_{start}$. At each step of the decoding process, the ground truth action label $\hat{a}_{m-1}$ from the previous step is added to the encoded input sequence by concatenating it to the result of the attention. The concatenated vector is then given as input to the LSTM decoder, which estimates probability scores a_m using a fully connected MLP with two layers. Given these probability scores and the ground truth action label $\hat{a}_m$, we compute per segment the action prediction loss $\mathcal{L}_{t_m}$ using cross-entropy. The final transcript prediction loss is defined as the sum of the action prediction losses, *i.e.*, $\mathcal{L}_t = \sum_{m=1}^{M+1} \mathcal{L}_{t_m}$. Note that we have $M+1$ terms since we add a special end symbol $\hat{a}_{end}$ to the ground-truth transcripts, which needs to be predicted by the network as well. The approach uses teacher forcing for training since we do not sample from the predicted action probabilities to feed the decoder, but we use the ground truth action labels $\hat{a}_m$.

Given the probability scores a_m and the hidden state of the decoder, we use another fully connected MLP with two layers to predict ℓ_m, which corresponds to the logarithm of the relative length of the segment. Notice that the parameters of this MLP are not updated based on the transcript prediction loss, but based on the mutual consistency loss $\mathcal{L}_\mu$.

4.4. Mutual Consistency Loss

Using the frame classification and the segment generation modules, our network produces two redundant representations for the action segmentation. We therefore propose the mutual consistency (MuCon) loss which enforces that the two redundant representations match each other and are mutually consistent. Using differentiable sampling with a bilinear kernel [12], we calculate the mutual consistency loss $\mathcal{L}_\mu$ as a function of Y and L in a differentiable way so that the gradients of this loss can be backpropagated to both the estimated length of the actions L and the action probabilities for each frame Y. An overview of the mutual consistency loss is shown in Figure 3.

Given the relative log length estimates $L_{1:M} = (\ell_1, \ldots, \ell_M)$ and the length of the video T, we first need to generate absolute length values $L'_{1:M} = (\ell'_1, \ldots, \ell'_M)$ such that $\sum_{m=1}^{M} \ell'_m = T$, *i.e.*, the absolute lengths sum up to be

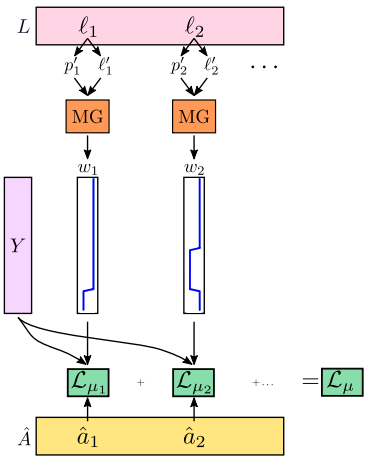

Figure 3. Overview of mutual consistency (MuCon) loss $\mathcal{L}_\mu$. Given the predicted lengths L, we generate a set of masks w_m using differentiable sampling using a bilinear kernel in the mask generation (MG) module. We then calculate the consistency of the estimated framewise probabilities Y for each w_m with the ground-truth action $\hat{a}_m$ using (5). The mutual consistency loss is then given by the sum of $\mathcal{L}_{\mu_m}$ over all segments.

equal to the length of the video. We do so by

$$\ell'_m = T \frac{e^{\ell_m}}{\sum_{k=1}^{M} e^{\ell_k}} \tag{1}$$

which is essentially equal to applying the softmax function on L and multiplying it by T. Notice that we are enforcing the lengths to be positive by predicting the log of the relative lengths. Having the absolute length ℓ'_m of each segment, we can also compute the absolute starting position p'_m for each segment by

$$p'_1 = 0 \tag{2}$$

$$p'_m = \sum_{k=1}^{m-1} \ell'_k. \tag{3}$$

The intuition of the mutual consistency loss is that we want to average the probabilities Y inside each predicted segment and compare it with the ground-truth labels $\hat{A}$. Using the absolute starting positions of the segments and their absolute lengths, we employ a 1-D variant of differentiable sampling [12] termed MG (mask generator) with a bilinear kernel to transform a window template, which is a vector of fixed size filled with ones, to a mask w_m that is one for the frames of a segment and zero otherwise. The MG uses ℓ'_m for scaling and p'_m for translation to generate the masks $W_{1:M} = (w_1, \ldots, w_M)$ where $w_m \in \mathbb{R}^T$. Note that the MG is differentiable, which allows to compute the gradient of the mutual consistency loss with respect to L. An example set of masks generated this way are shown in Figure 4. Notice that the value of the masks at the boundaries can be between 0 and 1 because of sampling artifacts.

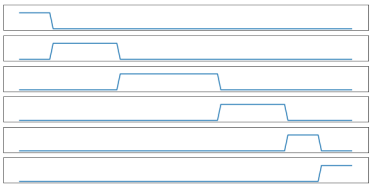

Figure 4. Example masks generated for a sequence with 6 segments. Due to sampling artifacts, the masks are not necessarily sharp at the boundaries.

Using the masks, the average unnormalized probability for each segment m is computed by

$$g(Y, w_m) = \frac{\sum_{t=1}^{T} y_t w_m[t]}{\ell'_m} \tag{4}$$

where $g(Y, w_m) \in \mathbb{R}^N$ and $w_m[t]$ is the value of the mask at frame t. For an estimated segment m, we say it is consistent with Y if the average probability of Y inside the window w_m is high for the action $\hat{a}_m$ and low for all other actions. Therefore we define the loss for each segment as

$$\mathcal{L}_{\mu_m}(Y, w_m, \hat{a}_m) = -\log\left(\frac{e^{g(Y,w_m)[\hat{a}_m]}}{\sum_{n=1}^{N} e^{g(Y,w_m)[n]}}\right) \tag{5}$$

which is essentially equal to applying the softmax function to the average $g(Y, w_m)$ and using the cross entropy loss. The final mutual consistency loss $\mathcal{L}_\mu$ is defined as the sum of all segment losses $\mathcal{L}_\mu = \sum_{m=1}^{M} \mathcal{L}_{\mu_m}$.

The mutual consistency loss is a differential function of the masks W and the frame-wise class probabilities Y. In turn, the masks W are differentiable with respect to the segment lengths L. This means that the gradients of the mutual consistency loss are backpropagated through all three modules of our network.

4.5. Length Regularizer

To prevent degenerate solutions *i.e.* solutions where the length of a segment is almost zero, we add a regularization term for the predicted relative lengths $L_{1:M}$.

$$\mathcal{L}_\ell(L_{1:M}) = \sum_{m=1}^{M} h(\ell_m - w) + h(-\ell_m - w) \tag{6}$$

The proposed length regularizer as defined in (6) prevents predicted lengths to become too large or too small. Here h is the ReLu function and w controls the acceptable range of estimated lengths.

4.6. Inference

While the frame classification module (Section 4.2) always generates the frame-wise probabilities $Y \in \mathbb{R}^{T \times N}$,

the number of segments generated by the segment generation module (Section 4.3) varies during inference. To obtain S, we start with the start symbol and the decoder generates new segments until the special end symbol $\hat{a}_{end}$ is predicted. The estimated relative segment lengths are then converted into absolute lengths using (1). For simplicity, we use L to denote the absolute lengths of the segments.

Since we obtain two redundant representations Y and S from the two heads of our network during inference, we can fuse them to obtain a single result for the input video features X. To this end, we keep the inferred transcript A, but adapt the lengths of the segments L using Y:

$$L^* = \underset{\tilde{L}}{\operatorname{argmax}}\, p(\tilde{L}|Y, S). \tag{7}$$

Similar to [21, 23], we can factorize the term $p(\tilde{L}|Y, S)$ yielding

$$L^* = \underset{\tilde{L}}{\operatorname{argmax}} \prod_{t=1}^{T} \frac{e^{y_t[a_{\alpha(t,\tilde{L})}]}}{\sum_{n=1}^{N} e^{y_t[n]}} \prod_{m=1}^{M} p(\tilde{\ell}_m|\ell_m). \tag{8}$$

The first term uses the softmax as in (5) to convert Y into probabilities. Depending on $\tilde{L}$ the segment number changes for a frame t and it is denoted by $\alpha(t, L)$. The second term penalizes deviations from the estimated absolute segment lengths using a Poisson distribution with mean ℓ_m:

$$p(\tilde{\ell}_m|\ell_m) = P_{\ell_m}(\tilde{\ell}_m) = \frac{(\ell_m)^{\tilde{\ell}_m} e^{-\ell_m}}{\tilde{\ell}_m!}. \tag{9}$$

The best possible lengths according to (8) is then obtained by dynamic programming [23]. Although we are using dynamic programming at test time, we want to emphasis that we only perform dynamic programming using the predicted transcript. Other methods like [20, 23] have to perform the dynamic programming for every transcript observed during training which results in testing time two orders of magnitude slower than our method.

5. Experiments

5.1. Evaluation Metric

Our evaluation of previously published weakly supervised action segmentation methods with open-source code [7, 23, 20] and our own method shows that depending on the network initialization, ordering of training data in each epoch and other randomization factors in training (e.g. CuDNN randomness) the accuracy can vary largely between different runs. We therefore recommend to report the average and standard deviation over 5 runs as best practice.

We reproduce the results of [7, 23, 20] using the official open-source code of these methods, by training and testing them on the Breakfast dataset 5 times each and reporting the average and standard deviation. We also do the same for our proposed method and baselines.

5.2. Benchmark Datasets

We evaluate our method on two popular action segmentation datasets, namely the Breakfast dataset [15] and the Hollywood extended dataset [2]. The Breakfast dataset contains more than 1.7k videos of different cooking activities. The videos belong to 10 different types of breakfast activities like *cereal* or *coffee* which consists of 48 different fine-grained actions. We report the average frame accuracy (MoF) metric over the predefined train/test splits following [22, 23, 7, 20]. The Hollywood extended dataset [2] contains 937 video sequences taken from Hollywood movies. The videos contain 16 different action classes. We follow the train/test split strategy of [22, 23, 20]. We report action segmentation and action alignment performance on this dataset using MoF and the conventional intersection over detection (IoD) metric for each task respectively.

5.3. Implementation Details

We train all three modules of our network together after initializing them with Gaussian random weights. The hidden size of the temporal modeling module is 128. Unless otherwise stated, we apply temporal max pooling with kernel size 2 after the convolutional layers $f_{w_1}, f_{w_2}, f_{w_4}, f_{w_8}$ for all experiments on the Breakfast dataset. For experiments on the Hollywood Extended dataset, we only apply 3 temporal max pooling operations after convolutional layers $f_{w_1}, f_{w_2}, f_{w_4}$ since the videos are shorter. The hidden size and embedding size of the bidirectional LSTM encoder and the LSTM decoder in our segment generation module is set to 128. We also employ an input embedding for the LSTM decoder of size 128 with 0.2 dropout.

Our final training loss is defined as $\mathcal{L} = \mathcal{L}_\mu + \mathcal{L}_t + \alpha * \mathcal{L}_\ell$ which is then optimized using SGD with weight decay of 0.005. The width of the length regularizer is set to 2 and α is set to 0.1 for experiments on the Breakfast dataset and 1.0 for the experiments on the Hollywood extended dataset. The initial learning rate is set to 0.01 and is lowered by a factor of 10 after 70 epochs for the Breakfast dataset and after 60 epochs for the Hollywood extended dataset. We train our network for 150 epochs for all of the experiments. As input features for the Breakfast and Hollywood extended datasets, we use RGB+flow I3D [3] features extracted from a network which was pretrained on the Kinetics400 dataset [14]. We perform all our experiments 5 times and report the average and standard deviation of the performance measure. More details are provided in the supplementary material. We will release the source code for reproducibility upon acceptance.

5.4. Ablation Experiments

In this section we quantitatively examine different components in our method. In this section we report the results on split 1 of the Breakfast dataset [15].

Model	Matching score
Ours - f_t - $\mathcal{L}_\mu$	0.621 ($\pm$ 0.023)
Ours - $\mathcal{L}_\mu$	0.705 ($\pm$ 0.017)
Ours	0.780 ($\pm$ 0.012)

(a) **Transcript prediction performance**: We observe that our temporal modelling module improves with transcript prediction performance. We also see that the MuCon loss improves the quality of the transcript prediction significantly.

Fusion type	MoF
Y	42.1 ($\pm$ 1.6)
S	44.0 ($\pm$ 2.2)
Average fusion	45.0 ($\pm$ 2.3)
DP fusion	47.1 ($\pm$ 2.1)

(b) **Fusion method of Y and S**: The proposed dynamic programming fusion approach (DP) outperforms average fusion thanks to using the predicted transcript from S and solving the over-segmentation problem of Y.

Table 1. Ablation experiments on split 1 of the Breakfast dataset. All of our results are the average over 5 runs with different random seeds.

Transcript prediction performance To see how effective our temporal modeling module f_t is, we train a segment generation module without f_t. To this end, we modify our network so that we only generate the segment S and train the network using only the transcript prediction loss $\mathcal{L}_t$ in two settings: without temporal modeling module (Ours - f_t - $\mathcal{L}_\mu$) and with temporal modeling module (Ours - $\mathcal{L}_\mu$). We evaluate the quality of the segments generated by calculating the matching score between the predicted transcript and the ground truth transcript and report the results in Table 1a. We can observe that not having the temporal module degrades the quality of the segments generated. Furthermore we can observe that the mutual consistency loss $\mathcal{L}_\mu$ helps to improve the accuracy of the inferred transcripts substantially.

Effect of fusion type for Y and S As mentioned in Section 4.6, we obtain two redundant representations Y and S from the two heads of our network during inference. We use a dynamic programming based fusion for our method (Section 4.6). To quantify the effect of the dynamic programming fusion, we also evaluate our method at inference using average fusion and report the results in Table 1b. We observe that Y achieves a lower accuracy than S. While the differences in MoF do not seem to be very drastic, the qualitative results show that Y tends to oversegment while S does not. Nevertheless, fusing the two representations improves the results as shown in Table 1b and the proposed fusion approach outperforms average fusion.

5.5. Comparison to State-of-the-Art

We compare our method, which we denote by MuCon, to state-of-the-art methods on the Breakfast dataset for weakly supervised action segmentation in Table 2 and on Hollywood extended dataset for weakly supervised action segmentation and alignment in Table 3 and Table 4. For action segmentation, we observe that our approach outperforms the methods [11, 22, 7, 23, 4] although our approach does not perform any dynamic programming or Viterbi decoding during training, is fully differentiable and much faster during training and testing. The MoF results are comparable to CDFL [20]. Similar to [22, 7, 23, 4], CDFL requires to perform costly dynamic time warping operations during training and it cannot generate an action order that has not been observed during training. This has been highlighted by the matching score metric that shows the quality of the prediction at segment level. On this metric, our method that is able to generate unseen transcript is able to achieve the best result. The results on the Hollywood extended dataset are similar. On both action segmentation and alignment on this dataset, we have competitive or better results.

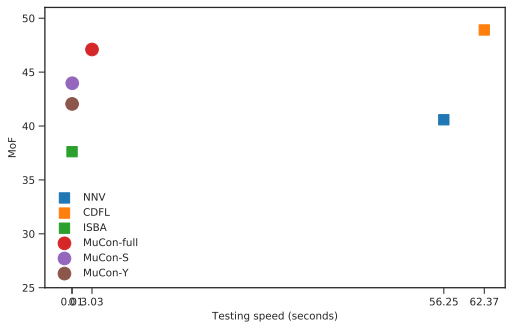

Figure 5. Testing speed vs MoF accuracy.

5.6. Training and Testing Speed

As mentioned before one of the advantages of our method is that during training we do not need to perform costly dynamic programming decoding. Because of this our method is 14 times faster to train compared to CDFL [20]. Also because our network is trained to predict the transcript, we only have to perform the dynamic programming decoding only using the predicted transcript while approaches like CDFL and NNV [23] have to perform the decoding for all transcripts seen during training. This makes our approach 20 times faster during inference.

5.7. Effect of Feature Type

As noted in Section 5.3, our method uses features extracted from the I3D network [3] pretrained on the Kinetics dataset [14] while other approaches like NNV and CDFL

Model	MoF Reported	MoF Avg (± Std)	MoF Max	MoF Min	Avg Matching Score (± Std)	Training Duration (hours)	Testing Duration (seconds)
ECTC [11]	27.7	-	-	-	-	-	-
HMM/RNN [22]	33.3	-	-	-	-	-	-
ISBA [7]	38.4	36.4 (± 1.0)	37.6	35.1	0.279 (± 0.011)	12.75	0.01
NNV [23]	43.0	39.7 (± 2.4)	43.5	37.5	0.686 (± 0.009)	11.23	56.25
D3TW [4]	45.7	-	-	-	-	-	-
CDFL [20]	50.2	48.1 (± 2.5)	50.9	44.6	0.712 (± 0.009)	66.73	62.37
MuCon - Y		42.2 (± 1.6)	44.2	40.2		4.57	0.02
MuCon - S		43.5 (± 1.7)	45.8	41.5	0.788 (± 0.009)	4.57	0.04
MuCon - full		47.1 (± 1.9)	49.7	44.8	0.788 (± 0.009)	4.57	3.03

Table 2. Weakly supervised action segmentation performance on the Breakfast dataset. The mean over frame (MoF) and matching score numbers are averaged over all dataset splits and 5 different runs. Training and Testing durations are reported only for split 1. Testing duration is divided by the number of videos in the test set of split 1. Note that our method *MuCon - full* achieves statistically the same MoF as the state-of-the-art *CDFL* [20] method and also performs better on the matching score metric that indicates the quality of the predicted transcripts. Also our method is 14 times faster to train and 20 times faster during testing.

Model	MoF Reported	MoF Avg (± Std)	MoF Max	MoF Min
ISBA [7]	28.7	-	-	-
D3TW [4]	33.6	-	-	-
CDFL [20]	40.6	-	-	-
MuCon		41.6 (± 1.4)	43.2	39.7

Table 3. Weakly supervised action segmentation performance on the Hollywood dataset. We report the mean over frames (MoF) accuracy averaged over all dataset splits. Note that for this dataset, the background class is not counted towards MoF as in [7, 4, 20]

Model	IoD Reported	IoD Avg (± Std)	IoD Max	IoD Min
HMM/RNN [22]	46.3	-	-	-
ISBA [7]	39.6	-	-	-
NNV [23]	48.7	-	-	-
D3TW [4]	50.9	-	-	-
CDFL [20]	52.9	-	-	-
MuCon		52.3 (± 1.7)	54.5	50.5

Table 4. Weakly supervised action alignment performance on the Hollywood dataset. We report the average intersection over detection metric (IoD) averaged over all dataset splits.

use IDT features. For a fair comparison, we have performed experiments using 64 dimensional IDT features and 2048 dimensional I3D features using NNV, CDFL and the proposed MuCon. The results are presented in Figure 6. We observe that NNV and CDFL show a significant performance drop using I3D feature. To show that this drop in performance is not a result of the I3D features being of higher dimension, we perform PCA dimension reduction on the I3D features and convert I3D features to 64 dimensional PCA-I3D features. We observe that while our model achieves reasonable results for all types of features and gets higher performance by using higher level features, the results of NNV and CDFL drop as we move from IDT to

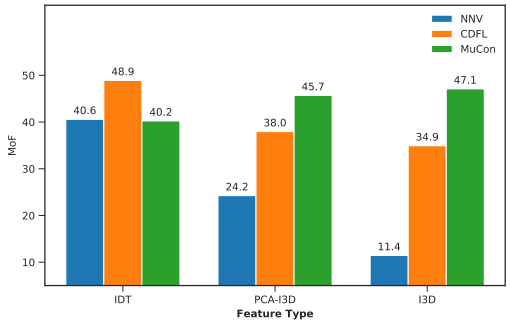

Figure 6. Performance using different features on split 1 of the Breakfast dataset.

PCA-I3D and I3D features.

6. Conclusion

We proposed a novel approach for weakly supervised action segmentation from transcript. Our proposed method consists of a two branch network that predicts two redundant but different representations for action segmentation at training time. We introduced a new mutual consistency loss (MuCon) that enforces these two representations to be consistent during training. Using MuCon and a transcript prediction loss, we are able to train our network end-to-end without the need for costly dynamic programming or iterative processes and achieve state-of-the-art results on the Breakfast and Hollywood extended datasets. Our proposed method is able to perform action segmentation and alignment competitively or better than state-of-the-art while being 14 times faster to train and 20 times faster at test time. We think that mutual consistency learning based on redundant representations can also be applied in other domains.

References

[1] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In *ICLR*, 2015. 2, 3, 4

[2] 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, 6

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

[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. *CVPR*, 2019. 1, 7, 8

[5] Kyunghyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: Encoder–decoder approaches. In *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*, 2014. 4

[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. 1

[7] Li Ding and Chenliang Xu. Weakly-supervised action segmentation with iterative soft boundary assignment. In *CVPR*, 2018. 1, 2, 6, 7, 8

[8] Yazan Abu Farha and Juergen Gall. Ms-tcn: Multi-stage temporal convolutional network for action segmentation. *CVPR*, 2019. 1, 2, 3

[9] Christoph Feichtenhofer, Haoqi Fan, Jitendra Malik, and Kaiming He. Slowfast networks for video recognition. *ICCV*, 2019. 1

[10] Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N Dauphin. Convolutional sequence to sequence learning. In *ICML*, 2017. 2

[11] De-An Huang, Li Fei-Fei, and Juan Carlos Niebles. Connectionist temporal modeling for weakly supervised action labeling. In *ECCV*, 2016. 1, 2, 7, 8

[12] Max Jaderberg, Karen Simonyan, Andrew Zisserman, et al. Spatial transformer networks. In *NIPS*, 2015. 4, 5

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

[14] Will Kay, Joao Carreira, Karen Simonyan, Brian Zhang, Chloe Hillier, Sudheendra Vijayanarasimhan, Fabio Viola, Tim Green, Trevor Back, Paul Natsev, et al. The kinetics human action video dataset. *arXiv*, 2017. 6, 7

[15] Hilde Kuehne, Ali Arslan, and Thomas Serre. The language of actions: Recovering the syntax and semantics of goal-directed human activities. In *CVPR*, 2014. 2, 6

[16] Hilde Kuehne, Juergen Gall, and Thomas Serre. An end-to-end generative framework for video segmentation and recognition. In *WACV*, 2016. 1, 2

[17] Hilde Kuehne, Alexander Richard, and Juergen Gall. Weakly supervised learning of actions from transcripts. *CVIU*, 2017. 1, 2

[18] Colin Lea, Michael D Flynn, Rene Vidal, Austin Reiter, and Gregory D Hager. Temporal convolutional networks for action segmentation and detection. In *CVPR*, 2017. 1, 2

[19] Colin Lea, Austin Reiter, René Vidal, and Gregory D Hager. Segmental spatiotemporal cnns for fine-grained action segmentation. In *ECCV*, 2016. 2

[20] Jun Li, Peng Lei, and Sinisa Todorovic. Weakly supervised energy-based learning for action segmentation. In *ICCV*, 2019. 1, 2, 6, 7, 8

[21] Alexander Richard and Juergen Gall. Temporal action detection using a statistical language model. In *CVPR*, 2016. 6

[22] Alexander Richard, Hilde Kuehne, and Juergen Gall. Weakly supervised action learning with rnn based fine-to-coarse modeling. In *CVPR*, 2017. 1, 2, 6, 7, 8

[23] Alexander Richard, Hilde Kuehne, Ahsan Iqbal, and Juergen Gall. Neuralnetwork-viterbi: A framework for weakly supervised video learning. In *CVPR*, 2018. 1, 2, 6, 7, 8

[24] Marcus Rohrbach, Sikandar Amin, Mykhaylo Andriluka, and Bernt Schiele. A database for fine grained activity detection of cooking activities. In *CVPR*, 2012. 2

[25] Karen Simonyan and Andrew Zisserman. Two-stream convolutional networks for action recognition in videos. In *NIPS*, 2014. 1

[26] 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. In *CVPR*, 2016. 4

[27] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *NIPS*, 2014. 2

[28] 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. 2, 3

[29] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NIPS*, 2017. 2

[30] Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. In *CVPR*, 2018. 1

[31] Zijun Wei, Boyu Wang, Minh Hoai Nguyen, Jianming Zhang, Zhe Lin, Xiaohui Shen, Radomir Mech, and Dimitris Samaras. Sequence-to-segment networks for segment detection. In *NIPS*. 2018. 2

[32] Yue Zhao, Yuanjun Xiong, Limin Wang, Zhirong Wu, Xiaoou Tang, and Dahua Lin. Temporal action detection with structured segment networks. In *ICCV*, 2017. 1, 2