Weakly supervised learning of actions from transcripts

Hilde Kuehne, Alexander Richard, Juergen Gall

University of Bonn, Germany

Abstract

We present an approach for weakly supervised learning of human actions from video transcriptions. Our system is based on the idea that, given a sequence of input data and a transcript, i.e. a list of the order the actions occur in the video, it is possible to infer the actions within the video stream, and thus, learn the related action models without the need for any frame-based annotation. Starting from the transcript information at hand, we split the given data sequences uniformly based on the number of expected actions. We then learn action models for each class by maximizing the probability that the training video sequences are generated by the action models given the sequence order as defined by the transcripts. The learned model can be used to temporally segment an unseen video with or without transcript. We evaluate our approach on four distinct activity datasets, namely Hollywood Extended, MPII Cooking, Breakfast and CRIM13. We show that our system is able to align the scripted actions with the video data and that the learned models localize and classify actions in the dataset competitively in comparison to models trained with full supervision, i.e. with frame level annotations, and that they outperform any current state-of-the-art approach for aligning transcripts with video data.

Keywords: Weak learning, Action recognition, Action classification, Temporal segmentation

1. Introduction

Weakly supervised learning has become of more and more interest in recent years, also in the field of action recognition (see e.g. [1, 2, 3, 4]). Here, especially the large amount of available data on open video platforms such as Youtube or Vimeo, but also in the context of surveillance, smart-homes or behavior analysis, constitutes a high demand for weakly supervised methods.

So far, most methods for action classification and detection rely on supervised learning. Thus for any new action to be trained, labels with temporal information are needed. But video annotation is very cost and time consuming, as it usually requires more than one label as well as corresponding time stamps to capture its relevant content. Relying on hand-annotation alone will make it difficult to cover larger amounts of video data and to develop systems to work on

Figure 1: Overview of weakly supervised learning from transcripts. The training data consists of video data and transcripts describing the order of the actions but not providing a labeling at frame level. Each action class is represented by a model that is initialized with uniform segments of the respective videos. The initialized model is used to infer the segments a video given a transcription of it. The new segmentation is used to reestimate and update the model parameters. The last two steps can be repeated until convergence.

large-scale domains outside existing presegmented datasets. Weakly supervised methods, such as the here presented one, might provide a first step towards a remedy in this case.

In this work, we propose a framework for weakly supervised learning of temporal action models from video data as shown in Figure 1. The approach is based on a combination of video data and their transcripts as input for training. The transcripts solely describe the order in which the actions occur in the video without the need of a temporal labeling of frames. Based on this input, we lend on the concept of flat models in speech recognition to infer the related actions from the video input. We model each action in form of an HMM and learn the model parameters based on the transcripts. First, we initialize each model by generating uniform splits of all videos, based on the transcript information. During training, we maximize the probability that the training video sequences are generated by the HMMs given the sequence order as defined by the transcripts. Therefore, we infer the segmentation boundaries for each video and use the new segmentation to reestimate the model. This process can be repeated until convergence. The learned models can then be used to convert a video transcription into temporal segmentation of a video or to segment and classify a video without any annotations. In the latter case, we additionally

learn a grammar from the transcripts of the training data which describes a valid order of actions.

We evaluate our approach on four distinct large-scale datasets for the segmentation of actions in video data, namely the Hollywood Extended [3], MPII Cooking [5], the Breakfast [6] and the CRIM13 [7] dataset with overall more than 100h of video data [1]. The proposed system relies on an appropriate amount of data to build robust probabilistic models for initialization and training. We compare our results against the baseline approach by [3] as well as against a supervised system. It shows that our approach does well compared to the baseline and that it is able to detect actions in unseen sequences based on the weakly trained models.

2. Related Work

The problem of action classification on presegmented video clips has been widely studied and considerable advances have been made. Mainly recent approaches using CNNs in combination with improved dense trajectories show promising results [8, 9, 10] compared to the long time de facto approach for action recognition, which is the combination of Fisher vectors and improved dense trajectories [11].

Based on the advances in action classification on presegmented videos, research starts to focus on the task of action detection in unsegmented videos. Here, the aim is to localize and classify a set of actions in temporally unrestricted videos. Some approach the problem as a localization task in the spatiotemporal domain [12, 13, 14] and cluster trajectories or apply dynamic programming to enable efficient search for actions in the input space. Most works, however, focus on finding the temporal location of actions, which can be seen as equivalent to a temporal video segmentation task. E.g. Shi et al. [15] use a semi-Markov model and find the best segmentation based on support vector machine scores. In [5], a sliding window approach with non-maximum suppression is applied and pose features are added to the dense trajectories. Ni et al. [16] analyze the video on multiple levels of granularity and track hands and objects to exploit the most relevant regions for finding actions. In order to adequately model long term dependencies of action segments in videos, a nonparametric hierarchical Bayesian model is used in the sequence memorizer in [17]. Others try to model temporal dynamics with mid-level concept detectors [18] or encounter the relation to speech processing and extract features comparable to acoustic features from a speech signal which are then fed into a classical hidden HMM based speech recognizer [19].

Additionally, e.g. in the context of hierarchical action detection, where long activities can be subdivided into smaller action units, several works make use of grammars [6, 20, 21, 22]. While the method of [21] is particularly designed for

[1] Code and data available under:
http://pages.iai.uni-bonn.de/kuehne_hilde/projects/weakLearning/index.html

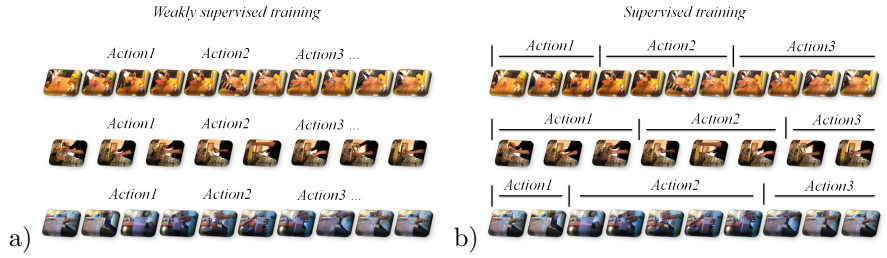


Figure 2: Problem statement of weakly supervised vs. supervised training. In case of weakly supervised training transcript annotations define only the order in which the actions occur is given without a temporal labeling of frames (a). In case of supervised training segmentation information is provided with temporal information (b).

hierarchical action detection and makes use of a forward-backward algorithm over the grammar's syntax tree, our method builds on [22], where a HMM is used in combination with a context free grammar to parse the most likely video segmentation and corresponding action classification. While all previously mentioned methods use fully supervised training and rely on annotation of all actions and their exact segment boundaries, our method only requires the sequence of actions that occur in the video but no segment boundaries at all.

In terms of weakly supervised video segmentation and action detection, first attempts have been made by Laptev et al. [23] using movie scripts for an automatic annotation of human actions in video. They align movie scripts with movie subtitles and use the temporal information associated with the subtitles to infer time intervals for a specific action. Building upon this work, Duchenne et al. [1] also use an automatic script alignment to provide coarse temporal localization of human actions and point out that the textual based detection usually leads to temporal misalignment. Another approach has been made in [2]. Here, the authors use only video-level annotation, e.g. the main action class in the video, and try to find appropriate localizations of the actual actions using web images found under the name of the action class. Most relevant to our method is the work [3] using the same kind of weak annotations that is also used in our approach and applying a discriminative clustering exploiting ordered action labels. In [4], Wu et al. propose an algorithm to model long-term dependencies and co-occurrences of actions in a long activity via topic priors and time distributions between action-topic pairs. They feature a k-means based clustering to build action-words and model activities as sequences of these words.

3. Transcript Annotation vs. Segment Annotation

Before we start with the actual system description, we describe the different types of annotation. We refer to the term *transcripts* if the annotation contains only the actions within a video and the order in which they occur (Figure 2 a), e.g.

`background, take_cup, pour_coffee, pour_milk, background`.

Opposed to that, a fully segmented annotation, also referred to as segmentation, explicitly requires the start and end times or frames for each action (Figure 2 b), e.g.

```
  0 -  61:  background
 62 - 197:  take_cup
198 - 376:  pour_coffee
277 - 753:  pour_milk
753 - 928:  background .
```

To analyze the cost of the different annotation techniques, we let annotators label both types and compare the overall annotation time. We chose 10 videos from the Breakfast dataset with the activity 'making coffee'. For this activity, there are seven possible action classes occurring within the videos. We let four test persons create a segmented annotation of the videos and measured the time they needed. Similarly, we let four other test persons create transcripts for the same videos and again measured the time. For both groups, we subtract the overall duration of the videos, and measure the additional time as annotation overhead. It shows that for a fully annotated segmentation annotators needed an overhead of 112.7 seconds on average to create the annotations for 10 videos. For the transcripts, annotators needed an overhead of only 14.1 seconds on average. Note that the difference can be expected to be even larger when the action segments in the video are shorter, e.g. as in the case of the fine-grained MPII dataset. Thus, being able to use weakly supervised data (i.e. transcripts) is beneficial since the annotation is much cheaper. Additionally, transcripts may even be directly extracted from subtitles [23, 24] without any annotation cost at all.

4. System Description

4.1. Feature Computation

For the proposed system, we use a combination of dense trajectories and reduced Fisher vectors. We compute dense trajectory features [11] and reduce the dimensionality of the descriptor from 426 dimensions to 64 dimensions by PCA, as described in [25]. To compute the Fisher vectors, we sample 100,000 random features to learn a Gaussian mixture model with 256 GMMs. The Fisher vector representation is computed for each frame over a sliding window of 20 frames using the implementation [26]. The dimensionality of the resulting vector is then reduced to 64 dimensions using PCA again. Thus, each frame is represented by a 64-dimensional reduced Fisher vector. We further apply an L2-normalization to each feature dimension separately for each video clip to further normalize the features. We denote the input sequence of each video as $\mathbf{x}_1^T = \{x_1, \ldots, x_T\}$ and refer to $x_t \in \mathbb{R}^m$ ($m = 64$ in our case) for the feature vector at frame t.

4.2. Model Initialization

Based on the work of [6], we define a HMM for each action class and model a video input as a concatenation of HMMs that are defined by the set of states $S = \{s_1, \ldots, s_n\}$, the state transition probability matrix $A \in \mathbb{R}^{n \times n}$ with the elements $a_{i,j}$ defining the transition from state s_i to state s_j, and the observation probabilities $b_j(x_t) = p(x_t|s_j)$. The $b_j(x_t)$ are probabilities defined by a Gaussian mixture model for state j. In our approach, HMMs are defined by a strict left-to-right feed forward topology, thus, only self-transitions and transitions to the next state are allowed.

The number of states for each HMM is set to a fraction of the mean action length in frames. The mean action length is computed based on the transcripts, i.e. it is obtained by dividing the number of frames by the number of action instances in the transcripts. We chose the number of states such that each state captures 10 video frames on average, respectively.

The state transition probabilities $a_{i,j}$ of each HMM are set initialized based on the average number of frames per state. As a state captures 10 frames on average, we set the transition probability $a_{i,i+1}$ to $0.1(= 1/10)$ and the self-transition probability $a_{i,i}$ to 0.9. As we feature a left-to-right feed forward model, all other probabilities are zero.

The observation probabilities b_j are initialized using a linear alignment of the action transcript to the video frames. Each video sequence is split to k segments of equal size, where k is the number of action instances in the transcript (see e.g. Figure 5 a)). This way, we obtain an initial alignment of video frames to HMMs that can be used to train the Gaussian mixture models that specify the distributions $b_j(x) = p(x|s_j)$. Since we model the observation probability by a single component Gaussian only in our system, the observation probability is parametrized only by the mean μ and the standard deviation σ of the related feature distribution.

4.3. Inference

In this section, we describe how to infer the segmentation of a video for two different cases: First for the case that the transcripts are provided and we are interested in the segmentation boundaries (Figure 3 a). This case is used in training as well as for the segmentation, when given a list of transcripts. Second, we describe inference for the case that no transcripts are provided and we need to estimate both, the sequence of occurring actions as well as the segment boundaries for each hypothesized action (Figure 3 b). This case is used for the combined classification and segmentation of video sequences.

Video Segmentation given the Action Transcripts. Given the action transcripts, a large sequence-HMM can be build that is a concatenation of the HMMs for each action class in the order they occur in the transcript for this sequence. Video segmentation can then be tackled by finding the best alignment of video frames to the sequence-HMM. The most likely alignment of the input frames to

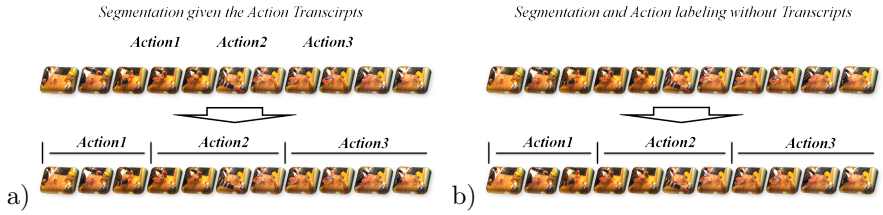


Figure 3: Example for video segmentation given the action transcripts compared to combined segmentation and action labeling without transcripts. In the first case the order of actions is given by the transcripts and only segmentation boundaries need to be inferred (a). In the second case no transcripts are provided and both, the sequence of occurring actions and the segment boundaries, are hypothesized for each action (b).

the sequence-HMM states is

$$\arg\max_{s_1,\ldots,s_T} p(s_1,\ldots,s_T|\mathbf{x}_1^T) = \prod_{t=1}^{T} p(x_t|s_t)\cdot p(s_t|s_{t-1}), \tag{1}$$

where each s_t is a state index of one of the original action class HMMs. The above maximization can be solved efficiently using the Viterbi algorithm. Once the alignment from frames to HMM states is computed, it is straightforward to infer the segment boundaries by looking at which points t there is a transition from the HMM of one action class to the HMM of another action class. An example for the boundary information obtained by applying the initialized models to the video sequence input is e.g. shown in Figure 5 b).

Video Segmentation and Action Labeling without Transcripts. If only the video but not its action transcripts are given, we need to infer both, the segment boundaries and the actual actions that occur in the video. A general approach is to define all valid action label sequences with a context free grammar and evaluate all possible paths. Since the typical datasets are frequently rather short and there is only a limited amount of training sequences, we use a minimalist grammar by just considering all possible paths that occur in the training data.

4.4. Weakly Supervised Learning

So far, we have introduced the model and showed how to use it for inference. It remains to show how it can be trained using the action transcripts only.

To this end, we compute the probability of an alignment going through state j at time t, $p(s_t = j|\mathbf{x}_1^T)$, which can be computed as the product of a forward and backward probability $\alpha_j(t)$ and $\beta_j(t)$,

$$\alpha_j(t) = p(s_t = j|\mathbf{x}_1^t), \tag{2}$$

$$\beta_j(t) = p(s_t = j|\mathbf{x}_t^T). \tag{3}$$

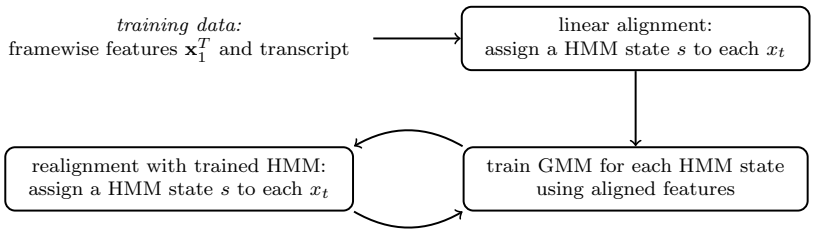


Figure 4: Weakly supervised training procedure with action transcripts. The model learns the alignment of frames to action classes, i.e. the segmentation, without supervision. At the beginning, the HMMs of each action instance in the video are concatenated and the frames are linearly distributed over the HMM states. This initial alignment is used to train the GMMs and obtain a HMM that better fits the training data. Using this HMM, a realignment of frames to HMM states is computed. The procedure is iterated until convergence.

The probabilities $\alpha_j(t)$ and $\beta_j(t)$ are computed using the Baum-Welch algorithm. This information is then used to update the single HMM models independently. We calculate the forward probabilities $\alpha_j(t)$ and backward probabilities $\beta_j(t)$ for all states j and times t. For each state j and time t, we use the product of the two probabilities and the current observation vector to update the observation probability for this state. Each observation is assigned to every state in proportion to the probability of the model being in that state when the vector was observed. The resulting new estimates are

$$\mu'_j = \frac{\sum_{t=1}^{T} \alpha_j(t)\beta_j(t)x_t}{\sum_{t=1}^{T} \alpha_j(t)\beta_j(t)}, \tag{4}$$

$$\sigma'_j = \frac{\sum_{t=1}^{T} \alpha_j(t)\beta_j(t)(x_t - \mu_j)(x_t - \mu_j)^T}{\sum_{t=1}^{T} \alpha_j(t)\beta_j(t)}. \tag{5}$$

With the updated model parameters, a new alignment can be computed as described in Section 4.3. With this new alignment, the model parameters can again be updated. This process is iterated until convergence. An overview of the process is shown in Figure 4. An example for the segmentation results after each step is shown in Figure 5.

Note that the initialization can be seen as a crucial factor for the overall system. If the linear alignment of frames to HMM states hardly matches the true action segments at all, e.g. if the segment lengths are very unbalanced, the system might learn unrealistic segmentation. However, if at least a certain amount of frames of the desired action is included in each initial segment, the algorithm usually converges quickly and achieves satisfying results. Also note that not all actions need to have a good initial model, as long as the overall parameters can be inferred by the subsequent Viterbi decoding.

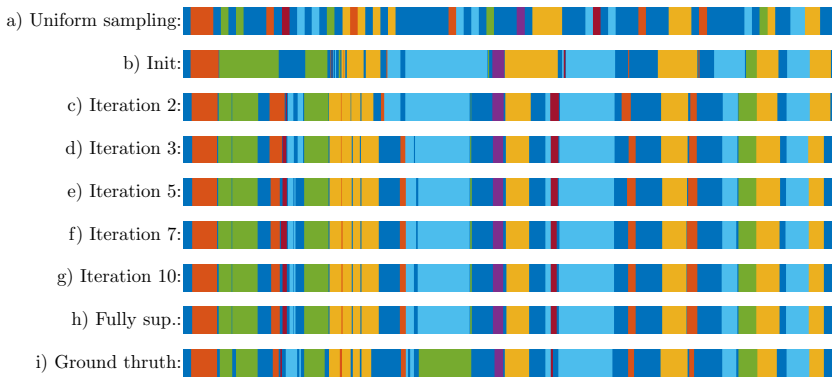


Figure 5: This example segmentation of one video from the MPII Cooking dataset shows how the segmentation evolves during the training. We start with a uniform sampling to initialize the model (a) and apply the initialized models to the overall sequence (b). The new boundaries are then used to train the new model parameters and the procedure is repeated until convergence (c-g). After three to five iterations, the inferred segmentation is fairly similar to the fully supervised segmentation (h) as well as to the annotated ground truth (i).

5. Experiments

In this section, we evaluate our method empirically for the two setups described in Section 4.3, the segmentation task, where we only infer the segmentation of a video given the transcripts, as well as the combined segmentation and classification, where the segmentation and action labels are inferred simultaneously without transcript information. In this case, the transcripts are only used during training to learn the models and build a grammar for the recognition system.

5.1. Experimental Setup

Our method is evaluated on the following four action detection datasets:

Hollywood Extended [3] is a dataset of 937 clips extracted from Hollywood movies. It features 15 action classes with a mean of 2.5 segments per video. For evaluation, we use ten-fold cross-validation, using the last digit in the file name as respective split index.

MPII Cooking [5] is a large database for the detection of fine grained cooking activities. It comprises 8h of video data with recordings of 12 different individuals. Including background, the dataset consists of 65 action classes with a mean of 86.1 segments per video. We use seven-fold cross-validation with the same splits that are also used in [5].

Breakfast [6] is a large scale database allowing for hierarchical activity recognition and detection and comprises roughly 77h of video and about 4 million frames. A set of 10 breakfast related activities is divided into 48 smaller action classes with 4.9 segments per video. We use the fine grained action annotation to evaluate the segmentation accuracy of our method. Recognition

results on the 10 coarse activity classes are also reported. Following [6], we use four splits for evaluation.

CRIM13 [7] is a large-scale mice behavior dataset, featuring about 50h of annotated mice activities, capturing the interactions of two mice as well as isolated behaviour. Overall 13 different action classes are annotated, with about 140 action segments within ten minutes of footage. The dataset comprises the side as well as the top view of a transparent cage. For the following evaluation, we only consider the side view of all activities. Additionally, as mice behaviour is clearly different from goal directed human behaviour in the other datasets, we favour a bi-gram model for sequence parsing instead of a fixed path grammar.

5.2. Inferring Video Segmentation from Action Transcripts

We first look at the segmentation capabilities of our model. In this case transcripts are provided and the aim is to infer the start and end frames of the respective action units. We investigate the quality of the hypothesized segments on both, training data and test data. In both cases the action transcripts are given, but we only use the training data to learn the parameters of the model.

We provide three performance measures for all experiments: We report mean over frames (MoF), which is the mean frame wise accuracy of the hypothesized video segmentation, and mean over classes (MoC), computed by taking the mean over frames for each class independently and computing the average over all classes. We also report the Jaccard index (Jacc) defined by intersection over union of ground truth and recognized action segment, also computed as mean over all classes. Looking at the characteristics of the provided measures, mean over frames weights each frame equally, so frequent or long action instances have a higher impact on the result. This is particularly important if one class dominates a dataset, which is often the case for the background class. Recognizing this class correctly can already lead to high MoF rates. For mean over classes, in contrast, this effect is averaged out. In exchange, rare classes may have an unreasonable large influence to the MoC score since each class contributes equally.

In Table 1 and 2, we evaluate the segmentation performance on the training as well as on the test data. First column, *naive*, is the naive baseline of each dataset. Here, the video is simply split into k segments of equal size, where k is the number of action instances in the transcripts. We assign the labels of the action transcripts to each corresponding segment as shown in Fig. 5 a) and compute the segmentation quality with respect to the ground truth. We can see that for all datasets the naive baseline already classifies 20%-30% of all frames correctly. As this splitting is used to initialize the related action models, this is also the amount of correct frames the system starts with.

The second column, *init*, shows the result of the initialized HMMs applied for segmentation before the first realignment (see also Figure 5 b)). One can see that this first inference based of the initialized models and transcripts leads to a significant improvement in segmentation accuracy. This shows that the linear alignment is enough to estimate the model parameters and can be seen as a

Dataset		train					
		naive	init	3rd	5th	10th	fully
HwdExt	*MoF*	0.472	0.451	0.494	0.508	**0.516**	0.691
	MoC	0.367	0.426	**0.483**	0.471	0.466	0.723
	Jacc	0.224	0.263	**0.291**	0.290	0.290	0.544
MPII	*MoF*	0.192	0.455	0.577	0.584	**0.590**	0.863
	MoC	0.128	0.385	0.492	0.498	**0.499**	0.910
	Jacc	0.068	0.228	0.318	0.324	**0.327**	0.792
Breakfast	*MoF*	0.312	0.393	0.439	**0.440**	0.434	0.891
	MoC	0.281	0.355	**0.404**	0.401	0.392	0.865
	Jacc	0.174	0.228	**0.266**	0.265	0.261	0.773
CRIM13	*MoF*	0.306	0.298	0.380	0.394	**0.401**	0.665
	MoC	0.155	0.289	0.361	0.377	**0.383**	0.692
	Jacc	0.083	0.131	0.179	0.189	**0.193**	0.454

Table 1: Comparison of the video segmentation quality on the training data for naive uniform segmentation (*naive*), initialization (*init*), weakly supervised training (*3rd, 5th and 10th iteration*), and fully supervised training (*fully*). Results are reported as mean over frames (*MoF*), mean over classes (*MoC*) and Jaccard index (*Jacc*).

hint that the initialization is successful. The new boundaries are then used for further reestimation.

For the *weakly* supervised results, we report results after three, five and ten iterations (Fig. 5 d), e) and g)). It shows that the accuracy increases significantly compared to the initial model especially for the first iterations. Comparing the results for the train and the test data, it can be observed that the systems starts to converge after three to five iterations and begins to overfit on the training data leading to a decreased accuracy in the test data.

To further analyse the effect of the number of iterations we plot segmentation results for all four datasets for up to 10 iterations in Figure 6. On all datasets, the obtained segmentation on both train and test set improves until the third iteration. After that, a degradation of the performance can be observed on the test data. Looking at the confusion matrix after ten iterations as shown in Figure 7 for MPII it shows two main reasons for this overfitting. First, successive classes that consistently appear in the same order are aggregated, here e.g. "take out from spice holder", "smell", "spice", "take put in spice holder". This is based on the fact that, if a certain combination of actions is always executed in the same order and the classes do not appear in other context, segment boundaries can be set at any point within those segments without decreasing the probability of the overall sequence. Second, accuracy for classes with few instances, such as "pull out" decreases. All datasets used for evaluation have a highly imbalanced distribution of class instances, e.g. for MPII the number of instances of the largest and the smallest differ by a factor of 42.8. Thus, classes with few samples will be initialized with very few data, and in the following only be detected for a few frames. Therefore, those models tend to degenerate during training. Accordingly, classes with more training samples

Dataset		test					
		naive	init	3rd	5th	10th	fully
HwdExt	*MoF*	0.473	0.441	0.491	0.505	**0.513**	0.645
	MoC	0.350	0.419	**0.476**	0.465	0.457	0.617
	Jacc	0.217	0.267	0.302	**0.305**	0.298	0.457
MPII	*MoF*	0.182	0.463	**0.598**	0.596	0.560	0.662
	MoC	0.109	0.354	**0.436**	0.426	0.427	0.503
	Jacc	0.057	0.220	0.309	0.312	**0.314**	0.408
Breakfast	*MoF*	0.304	0.374	**0.400**	0.386	0.376	0.776
	MoC	0.245	0.330	**0.347**	0.319	0.302	0.638
	Jacc	0.165	0.223	**0.244**	0.230	0.221	0.532
CRIM13	*MoF*	0.304	0.298	0.383	0.391	**0.392**	0.646
	MoC	0.153	0.287	**0.357**	0.352	0.342	0.643
	Jacc	0.081	0.137	**0.187**	0.186	0.181	0.436

Table 2: Comparison of the video segmentation quality on the test data for naive uniform segmentation (*naive*), initialization (*init*), weakly supervised training (*3rd, 5th and 10th iteration*), and fully supervised training (*fully*). *Test* refers to the segmentation results on unseen data.

will be recognized more often and thus result in more general models such as e.g. background classes.

We also compare the results to the *fully* supervised setup (Table 1 and 2). Here we just use the segmentation ground truth for the initial assignment of frames to HMM states (Fig. 5 h)) and do not apply any boundary reestimation. It shows that on test data the weakly learned models catch up e.g. to 5%-10% on Hollywood Extended and MPII compared to the supervised segmentation. Further, it achieves at least 50%, and usually more accuracy compared to the fully supervised models.

Figure 8 shows qualitative segmentation results for videos from MPII Cooking and Breakfast dataset. The first color bar shows the *naive* segmentation. The second color bar, *init* refers to the inference results of initial models. For both datasets, the first examples show a good matching of the weakly supervised segmentation compared to the ground truth with only slightly shifted segment boundaries. The second images show cases where larger segments have been detected, but some segments are not aligned correctly.

5.3. Comparison to State-of-the-art

We compare the proposed framework to the performance of another state-of-the-art system of Bojanowski et al. [3], which is also able to infer video segmentation boundaries given weak annotations similar to ours. We apply the source code as provided by the authors to all four datasets for a comparative baseline shown in Tab. 3. For our system, we fix the number of iterations for all dataset to three here as well as for the following experiments. We again report mean over frames (*MoF*), mean over classes (*MoC*) and Jaccard index (*Jacc*). Note that we compute the Jaccard index as intersection over union, which is

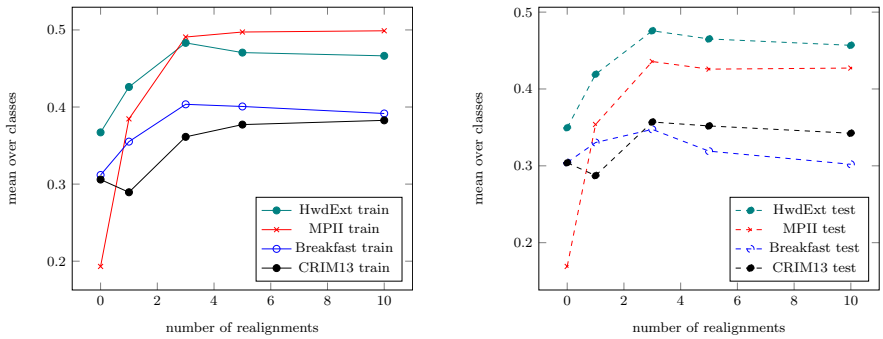


Figure 6: Effect of the number of realignments on the segmentation results on train and test set of Hollywood Extended, MPII, Breakfast, and CRIM13 (MoC).

different from the Jaccard index used in [3] that is computed as intersection over detection (IoD). Overall, it shows that especially for larger datasets the proposed approach outperforms the method of [3].

More recent approaches based on connectionist temporal models [27] report a best MoF accuracy of 27.7% for the Breakfast dataset, which is still below the performance of the proposed model. They also plot an average Jaccard index of approximately 0.41 for the Hollywood Extended dataset following the protocol of [3] which for our case would be 0.472 on the training data and 0.423 on the test data.

5.4. Combined Classification and Segmentation from Video Data Only

In this section, we evaluate the resulting models of the weak learning for the combined classification and segmentation of a video sequence, i.e. localizing and classifying the actions. Thus only video features of the test set are provided without any further information. The task is then to localize the respective action classes and to segment them just based on the models gained by weak learning. To this end, we again compare three different training modalities: (a) for the initial model based on linear segmentation only, (b) after the weakly supervised training, and (c) for the fully supervised training. Note that in this case, no annotations are used at all for the parsing of the test input. This is different from the segmentation task in Section 5.2, where we infer the segmentation given the action transcripts for the test data. In order to model the temporal dependencies between the hypothesized actions, a context free grammar is used, cf. Section 4.3. We again report mean over frames (MoF), mean over classes (MoC) and Jaccard index (Jacc). For the Breakfast dataset, we also include *Activity*, which is the high level activity classification accuracy, using the context free grammar to look up which high level activity the recognized action sequence belongs to.

Naturally, applying fully supervised training with all data achieves better results than just weakly supervised training. Still, it becomes clear that the

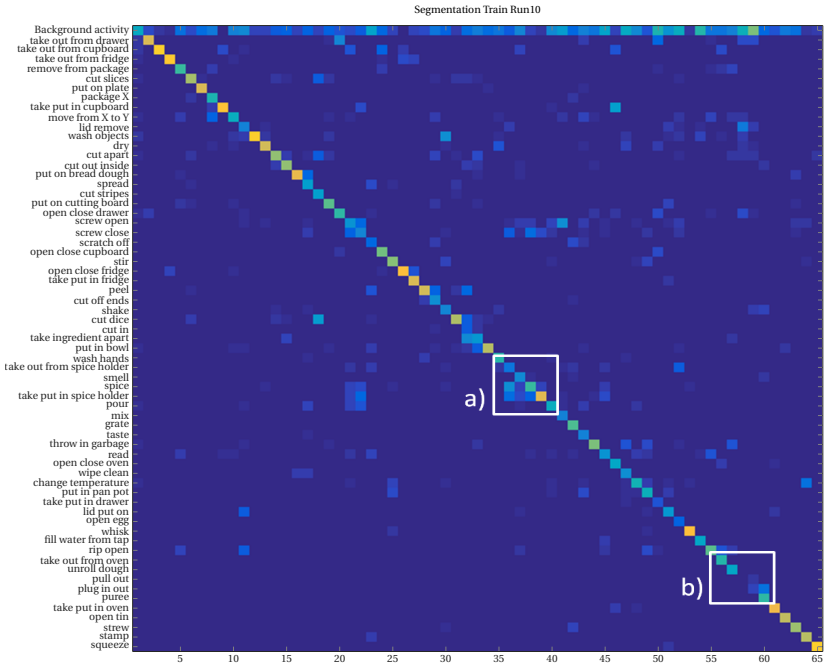


Figure 7: Confusion matrix for segmentation on MPII test set after ten iterations. Especially classes that often appear in successive order tend to be aggregated, such as "take out from spice holder", "smell", "spice", "take put in spice holder" (a). Additionally, classes with minor instances tend to be suppressed (b).

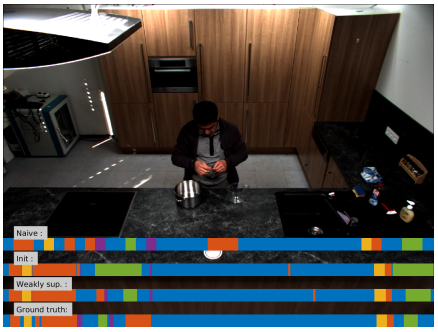
gap between both training methods is comparably small. While the mean over classes decreases on MPII Cooking for fully supervised training, the result is nearly the same for weakly supervised training. Also for the task of activity recognition on the Breakfast dataset, weakly supervised training achieves competitive results compared to fully supervised training.

Figure 9 shows example results for the combined segmentation and classification on MPII Cooking and Breakfast. Since we do not use the transcripts in this setup, the result deviates more from the from the ground truth compared to the results of the segmentation task shown in Figure 8.

5.5. Runtime

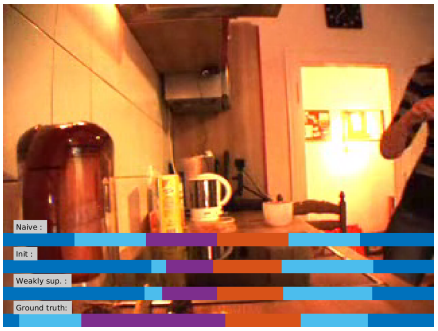
Finally, we look at the runtime of the weak learning process. We report the runtime for training with three iterations, for the segmentation on the test data as well as for combined classification and segmentation. We process the first split of each dataset on one core of an Intel Core i7 CPU with 3.30GHz and compared it with computation time of [3] under same conditions. The reported time refers only to the inference resp. optimization process and does not include feature computation or loading. Note that classification here refers to a

(a) Segmentation of the video `s16-d11-cam002`.



(b) Segmentation of the video `s16-d06-cam002`.

(c) Segmentation of the video `P15_cereal`.

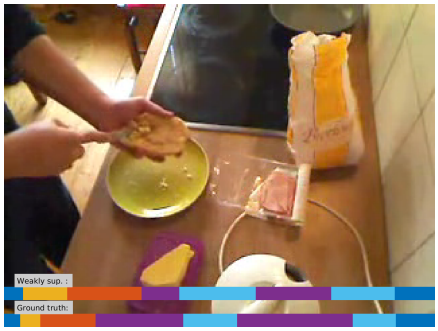


(d) Segmentation of the video `P13_cereal`.

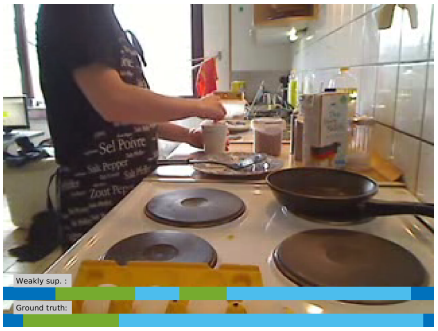
Figure 8: Example for video segmentation from action transcripts, showing first the naive baseline, the alignment after the initialization, after three iterations as well as the ground truth segmentation. It shows that the first inference after the initialization already leads to an acceptable segmentation, that is further refined during training.

(a) Classification and segmentation result of the video `s16-d01-cam002`.

(b) Classification and segmentation result of the video `s16-d09-cam002`.



(c) Classification and segmentation result of the video `P12_sandwich`.



(d) Classification and segmentation result of the video `P08_milk`.

Figure 9: Example for the combined classification and segmentation without transcript information. It shows that the inferred results deviate more from the from the ground truth compared to the results of the segmentation task.

Dataset		train		test	
		ours	[3]	ours	[3]
HwdExt	*MoF*	**0.494**	0.218	**0.491**	0.220
	MoC	**0.483**	0.360	**0.476**	0.370
	Jacc	**0.291**	0.186	**0.302**	0.171
MPII	*MoF*	**0.581**	0.048	**0.602**	0.048
	MoC	**0.491**	0.016	**0.436**	0.025
	Jacc	**0.319**	0.001	**0.319**	0.002
Breakfast	*MoF*	**0.439**	0.210	**0.400**	0.211
	MoC	**0.404**	0.188	**0.347**	0.182
	Jacc	**0.266**	0.086	**0.244**	0.080

Table 3: Comparison of the segmentation results obtained with the method from [3] with the results of our method. Results are reported in mean over frames (*MoF*), mean over classes (*MoC*) and Jaccard index (*Jacc*).

combined classification and segmentation without any transcripts as described in Section 5.4 as opposed to a simple segmentation task with given transcripts (Section 5.2). Classification and segmentation thus not only infer a single sequence, but, theoretically needs to compute all possible paths to maximize the probability for a given observation sequence, leading thus to a longer runtime than a simple segmentation task.

Overall the proposed method is roughly one order of magnitude faster than [3]. This is mainly based on the different number of iterations used for both methods. As discussed in Section 5.2, we fixed the number of iterations to three, as the proposed method usually converges and starts to overfit at this point. For [3], the authors recommend 200 iterations resulting thus in a longer overall runtime.

6. Conclusions

We proposed an approach for weakly supervised learning of a temporal action model. For training we use a combination of a frame-based video representation and the corresponding transcripts to infer the scripted actions, and thus, learn the related action models without the need of a frame level annotation. To do this, we model each action by a HMM and iterate sequence decoding and model reestimation to adapt and train the related models, based on the transcribed input. We evaluated our approach on four challenging activity segmentation datasets and showed that the process iteratively improves the estimation of the segment boundaries and the action classification. The weakly supervised learned action models are competitive in comparison to the models learned with full supervision, showing that weakly supervised learning for temporal semantic video segmentation is also feasible for large-scale video datasets.

Dataset		init	weakly sup.	fully sup.
HwdExt	*MoF*	0.259	0.330	0.395
	MoC	0.109	0.186	0.177
	Jacc	0.042	0.086	0.084
MPII	*MoF*	0.443	0.597	0.720
	MoC	0.329	0.432	0.572
	Jacc	0.196	0.297	0.455
Breakfast	*MoF*	0.163	0.259	0.685
	MoC	0.123	0.167	0.469
	Jacc	0.052	0.098	0.361
	Activity	0.405	0.566	0.664
CRIM13	*MoF*	0.113	0.238	0.328
	MoC	0.191	0.287	0.530
	Jacc	0.053	0.108	0.187

Table 4: Comparison of the combined video classification and segmentation quality for the initial model as well as the weakly supervised model (three iterations) and fully supervised model, respectively. Results are reported in mean over frames (*MoF*), mean over classes (*MoC*) and Jaccard index (Jacc). *Activity* is the high level activity classification accuracy on Breakfast.

Dataset	Train	Segmentation	Classification	[3] -Train
HwdExt	0.47 min	0.06 min	1.61 min	8.2 min
MPII	5.1 min	0.93 min	13.7 min	42.6 min
Breakfast	3.9 min	0.35 min	23.7 min	115.4 min

Table 5: Evaluation of runtime for the weak learning with three iterations, segmentation on the test data and combined classification and segmentation.

7. Acknowledgment

This work was supported by the DFG Emmy Noether program (GA 1927/1-1) and the DFG grant KU 3396/2-1.

[1] O. Duchenne, I. Laptev, J. Sivic, F. Bach, J. Ponce, Automatic annotation of human actions in video, in: Int. Conf. on Computer Vision, 2009, pp. 1491–1498.

[2] C. Sun, S. Shetty, R. Sukthankar, R. Nevatia, Temporal localization of fine-grained actions in videos by domain transfer from web images, in: ACM Conf. on Multimedia, 2015, pp. 371–380.

[3] P. Bojanowski, R. Lajugie, F. Bach, I. Laptev, J. Ponce, C. Schmid, J. Sivic,

Weakly supervised action labeling in videos under ordering constraints, in: European Conf. on Computer Vision, 2014.

[4] C. Wu, J. Zhang, S. Savarese, A. Saxena, Watch-n-patch: Unsupervised understanding of actions and relations, in: IEEE Conf. on Computer Vision and Pattern Recognition, 2015, pp. 4362–4370.

[5] M. Rohrbach, S. Amin, M. Andriluka, B. Schiele, A database for fine grained activity detection of cooking activities, in: IEEE Conf. on Computer Vision and Pattern Recognition, 2012, pp. 1194–1201.

[6] H. Kuehne, A. Arslan, T. Serre, The language of actions: Recovering the syntax and semantics of goal-directed human activities, in: IEEE Conf. on Computer Vision and Pattern Recognition, 2014, pp. 780–787.

[7] X. Burgos-Artizzu, P. Dollár, D. Lin, D. Anderson, P. Perona, Social behavior recognition in continuous videos, in: IEEE Conf. on Computer Vision and Pattern Recognition, 2012.

[8] M. Jain, J. C. van Gemert, C. G. Snoek, What do 15,000 object categories tell us about classifying and localizing actions?, in: IEEE Conf. on Computer Vision and Pattern Recognition, 2015, pp. 46–55.

[9] K. Simonyan, A. Zisserman, Two-stream convolutional networks for action recognition in videos, in: Advances in Neural Information Processing Systems, 2014, pp. 568–576.

[10] C. Feichtenhofer, A. Pinz, A. Zisserman, Convolutional two-stream network fusion for video action recognition, in: IEEE Conf. on Computer Vision and Pattern Recognition, 2016.

[11] H. Wang, C. Schmid, Action recognition with improved trajectories, in: Int. Conf. on Computer Vision, 2013, pp. 3551–3558.

[12] J. C. van Gemert, M. Jain, E. Gati, C. G. Snoek, Apt: Action localization proposals from dense trajectories, in: British Machine Vision Conference, 2015.

[13] M. Jain, J. Van Gemert, H. Jégou, P. Bouthemy, C. G. Snoek, Action localization with tubelets from motion, in: IEEE Conf. on Computer Vision and Pattern Recognition, 2014, pp. 740–747.

[14] J. Yuan, Z. Liu, Y. Wu, Discriminative video pattern search for efficient action detection, IEEE Transactions on Pattern Analysis and Machine Intelligence 33 (2011) 1728–1743.

[15] Q. Shi, L. Wang, L. Cheng, A. Smola, Discriminative human action segmentation and recognition using semi-markov model, in: IEEE Conf. on Computer Vision and Pattern Recognition, 2008.

[16] B. Ni, V. R. Paramathayalan, P. Moulin, Multiple granularity analysis for fine-grained action detection, in: IEEE Conf. on Computer Vision and Pattern Recognition, 2014, pp. 756–763.

[17] Y. Cheng, Q. Fan, S. Pankanti, A. Choudhary, Temporal sequence modeling for video event detection, in: IEEE Conf. on Computer Vision and Pattern Recognition, 2014, pp. 2235–2242.

[18] S. Bhattacharya, M. M. Kalayeh, R. Sukthankar, M. Shah, Recognition of complex events: Exploiting temporal dynamics between underlying concepts, in: IEEE Conf. on Computer Vision and Pattern Recognition, 2014, pp. 2243–2250.

[19] C.-C. Chen, J. Aggarwal, Modeling human activities as speech, in: IEEE Conf. on Computer Vision and Pattern Recognition, IEEE, 2011, pp. 3425–3432.

[20] H. Pirsiavash, D. Ramanan, Parsing videos of actions with segmental grammars, in: IEEE Conf. on Computer Vision and Pattern Recognition, 2014, pp. 612–619.

[21] N. N. Vo, A. F. Bobick, From stochastic grammar to bayes network: Probabilistic parsing of complex activity, in: IEEE Conf. on Computer Vision and Pattern Recognition, 2014, pp. 2641–2648.

[22] H. Kuehne, J. Gall, T. Serre, An end-to-end generative framework for video segmentation and recognition, in: Proc. IEEE Winter Applications of Computer Vision Conference (WACV 16), 2016.

[23] I. Laptev, M. Marszalek, C. Schmid, B. Rozenfeld, Learning realistic human actions from movies, in: IEEE Conf. on Computer Vision and Pattern Recognition, 2008.

[24] J. Alayrac, P. Bojanowski, N. Agrawal, J. Sivic, I. Laptev, S. Lacoste-Julien, Learning from narrated instruction videos, in: IEEE Conf. on Computer Vision and Pattern Recognition, 2016.

[25] D. Oneata, J. Verbeek, C. Schmid, Action and event recognition with fisher vectors on a compact feature set, in: Int. Conf. on Computer Vision, 2013, pp. 1817–1824.

[26] A. Vedaldi, B. Fulkerson, Vlfeat library (2008).

[27] D.-A. Huang, L. Fei-Fei, J. C. Niebles, Connectionist temporal modeling for weakly supervised action labeling, in: IEEE Conf. on Computer Vision and Pattern Recognition, 2016.