

# Weakly Supervised Action Learning with RNN based Fine-to-coarse Modeling

Alexander Richard, Hilde Kuehne, Juergen Gall  
University of Bonn, Germany

{richard,kuehne,gall}@iai.uni-bonn.de

## Abstract

We present an approach for weakly supervised learning of human actions. Given a set of videos and an ordered list of the occurring actions, the goal is to infer start and end frames of the related action classes within the video and to train the respective action classifiers without any need for hand labeled frame boundaries. To address this task, we propose a combination of a discriminative representation of subactions, modeled by a recurrent neural network, and a coarse probabilistic model to allow for a temporal alignment and inference over long sequences. While this system alone already generates good results, we show that the performance can be further improved by approximating the number of subactions to the characteristics of the different action classes. To this end, we adapt the number of subaction classes by iterating realignment and reestimation during training. The proposed system is evaluated on two benchmark datasets, the Breakfast and the Hollywood extended dataset, showing a competitive performance on various weak learning tasks such as temporal action segmentation and action alignment.

## 1. Introduction

Given the large amount of available video data, *e.g.* on Youtube, from movies or even in the context of surveillance, methods to automatically find and classify human actions within these videos gained an increased interest within the last years [30, 12, 26, 23, 34].

While there are several successful methods to classify trimmed video clips [30, 26], temporal localization and classification of human actions in untrimmed, long video sequences are still a huge challenge. Most existing approaches in this field rely on fully annotated video data, *i.e.* the exact start and end time of each action in the training set needs to be provided [24, 23, 34]. For real world applications, this requires an enormous effort of creating training data and can be too expensive to realize. Therefore, weakly supervised methods are of particular interest. Such methods usually assume that only an ordered list of actions occurring

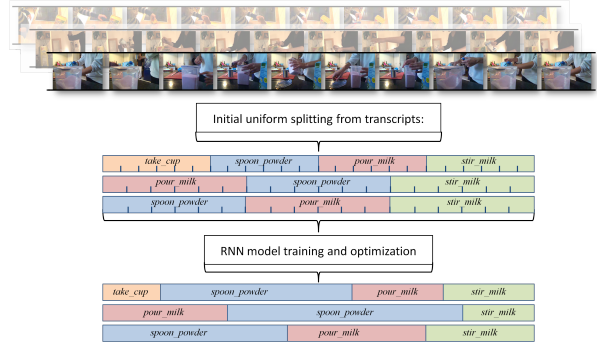


Figure 1. Overview of the proposed weak learning system. Given a list of ordered actions for each video, an initial segmentation is generated by uniform segmentation. Based on this input information we iteratively train an RNN-based fine-to-coarse system to align the frames to the respective action.

in the video is annotated instead of exact framewise start and end points [7, 3, 9]. This information is much easier to generate for human annotators, or can even be automatically derived from scripts [17, 20] or subtitles [1]. The idea that all those approaches share is that - given a set of videos and a respective list of the actions that occur in the video - it is possible to learn the characteristics of the related action classes, to infer their start and end frames within the video, and to build the corresponding action models without any need for hand labeled frame boundaries (see Figure 1).

In this work, we address the task of weak learning of human actions by a fine-to-coarse model. On the fine grained level, we use a discriminative representation of subactions, modeled by a recurrent neural network as *e.g.* used by [6, 35, 27, 33]. In our case, the RNN is used as basic recognition model as it provides robust classification of small temporal chunks. This allows to capture local temporal information. The RNN is supplemented by a coarse probabilistic model to allow for temporal alignment and inference over long sequences.

Further, to bypass the difficulty of modeling long and complex action classes, we divide all actions into smaller building blocks. Those subactions are eventually modeled within the RNN and later combined by the inference process. The usage of subactions allows to distribute hetero-

geneous information of one action class over many subclasses and to capture characteristics such as the length of the overall action class. Additionally, we show that automatically learning the number of subactions for each action class leads to a notably improved performance.

Our model is trained with an iterative procedure. Given the weakly supervised training data, an initial segmentation is generated by uniformly distributing all actions among the video. For each obtained action segment, all subactions are then also uniformly distributed among the part of the video belonging to the corresponding action. This way, an initial alignment between video frames and subactions is defined. In an iterative phase, the RNN is then trained on this alignment and used in combination with the coarse model to infer new action segment boundaries. From those boundaries, we recompute the number of subactions needed for each action class, distribute them again among the frames aligned to the respective action, and repeat the training process until convergence.

We evaluate our approach on two common benchmark datasets, the Breakfast dataset [14] and the Hollywood extended dataset [3], regarding two different tasks. The first task is temporal action segmentation, which refers to a combined segmentation and classification, where the test video is given without any further annotation. The second task is aligning a test video to a given order of actions, as proposed by Bojanowski *et al.* [3]. Our approach is able to outperform current state-of-the-art methods on both tasks.

## 2. Related Work

For the case of fully supervised learning of actions, well-studied deep learning and temporal modeling approaches exist. While the authors of [34] focus on a purely neural network based approach, Tang *et al.* [29] propose to learn the latent temporal structure of videos with a hidden Markov model. Combining deep learning and temporal modeling, the authors of [18] use a segmental CNN and a semi-Markov model to represent temporal transitions between actions. However, these methods are not applicable in a weakly supervised setting.

Addressing the problem of weakly supervised learning of actions, a variety of different approaches have been explored. First works, proposed by Laptev *et al.* [17] and Marszalek *et al.* [20], focus on mining training samples from movie scripts. They extract class samples based on the respective text passages and use those snippets for training without applying a dedicated temporal alignment of the action within the extracted clips. First attempts for learning action classes including temporal alignment on weakly annotated data are made by Duchenne *et al.* [7]. Here, it is assumed that all snippets contain only one class and the task is to temporally segment frames containing the relevant action from the background activities. The temporal align-

ment is thus interpreted as a binary clustering problem, separating temporal snippets containing the action class from the background segments. The clustering problem is formulated as a minimization of a discriminative cost function. This problem formulation is extended by Bojanowski *et al.* [3] also introducing the Hollywood extended dataset. Here, the weak learning is formulated as a temporal assignment problem. Given a set of videos and the action order of each video, the task is to assign the respective class to each frame, thus to infer the respective action boundaries. The authors propose a discriminative clustering model using the temporal ordering constraints to combine classification of each action and their temporal localization in each video clip. They propose the usage of the Frank-Wolfe algorithm to solve the convex minimization problem. This method has been adopted by Alayrac *et al.* [1] for unsupervised learning of task and story lines from instructional video. Another approach for weakly supervised learning from temporally ordered action lists is introduced by Huang *et al.* [9]. They feature extended connectionist temporal classification and propose the induction of visual similarity measures to prevent the CTC framework from degeneration and to enforce visually consistent paths. On the other hand, Kuehne *et al.* [16] borrow on the concept of flat models in speech recognition. They model actions by hidden Markov models (HMMs) and aim to maximize the probability of training sequences being generated by the HMMs by iteratively inferring the segmentation boundaries for each video and using the new segmentation to reestimate the model. The last two approaches were both evaluated on the Hollywood extended as well as on the Breakfast dataset, thus, these two datasets are also used for the evaluation of the here proposed framework.

Beside the approaches focusing on weak learning of human actions based on temporally ordered labels, also other weak learning scenarios have been explored. A closely related approach comes from the field of sign language recognition. Here, Koller *et al.* [13] integrate CNNs with hidden Markov models to learn sign language hand shapes based on a single frame CNN model from weakly annotated data. They evaluate their approach on various large scale sign language corpora, *e.g.* for Danish and New Zealand sign language. Gan *et al.* [8] show an approach to learn action classes from web images and videos retrieved by specific search queries. They feature a pairwise match of images and video frames and combine this with a regularization over the selected video frames to balance the matching procedure. The approach is evaluated on standard action classification datasets such as UCF101 and Trecvid. Also learning from web videos and images is the approach of [28]. Weak video labels and noisy image labels are taken as input, and localized action frames are generated as output. The localized action frames are used to train action recognition

models with long short-term memory networks. Results are reported, among others, for temporal detection on the THU-MOS 2014 dataset. Another idea is proposed by Misra *et al.* [21], aiming to learn a temporal order verification for human actions in an unsupervised way by training a CNN with correct vs. shuffled video snippets and thus capturing temporal information. The system can be used for pre-training feature extractors on small datasets as well as in combination with other supervised methods. A more speech related task is also proposed by Malmaud *et al.* [19], trying to align recipe steps to automatically generated speech transcripts from cooking videos. They use an hybrid HMM model in combination with a CNN based visual food detector to align a sequence of instructions, e.g. from textual recipes, to a video of someone carrying out a task. Finally, [32] propose an unsupervised technique to derive action classes from RGB-D videos, respectively human skeleton representations, also considering an activity as a sequence of short-term action clips. They propose Gibbs sampling for learning and inference of long activities from basic action words and evaluate their approach on an RGB-D activity video dataset.

### 3. Technical Details

In the following, we describe the proposed framework in detail, starting with a short definition of the weak learning task and the related training data. We then define our model and describe the overall training procedure as well as how it can be used for inference.

#### 3.1. Weakly Supervised Learning from Action Sequences

In contrast to fully supervised action detection or segmentation approaches, where frame based ground truth data is available, in weakly supervised learning only an ordered list of the actions occurring in the video is provided for training. A video of making tea, for instance, might consist of taking a cup, putting the teabag in it, and pouring water into the cup. While fully supervised tasks would provide a temporal annotation of each action start and end time, in our weakly supervised setup, all given information is the ordered action sequence

`take_cup, add_teabag, pour_water.`

More formally, we assume the training data is a set of tuples  $(\mathbf{x}_1^T, \mathbf{a}_1^N)$ , where  $\mathbf{x}_1^T$  are framewise features of a video with  $T$  frames and  $\mathbf{a}_1^N$  is an ordered sequence  $(a_1, \dots, a_N)$  of actions occurring in the video. The segmentation of the video is defined by the mapping

$$n(t) : \{1, \dots, T\} \mapsto \{1, \dots, N\} \quad (1)$$

that assigns an action segment index to each frame. Since our model iteratively optimizes the action segmentation, initially, this can simply be a linear segmentation of the provided actions, see Figure 4a. The likelihood of the video frames  $\mathbf{x}_1^T$  given the action transcripts  $\mathbf{a}_1^N$  is then defined as

$$p(\mathbf{x}_1^T | \mathbf{a}_1^N) := \prod_{t=1}^T p(x_t | a_{n(t)}), \quad (2)$$

where  $p(x_t | a_{n(t)})$  is the probability of frame  $x_t$  being generated by the action  $a_{n(t)}$ .

The action classes given for training usually describe longer, task-oriented procedures that naturally consist of more than one significant motion, e.g. *take\_cup* can involve moving a hand towards a cupboard, opening the cupboard, grabbing the cup and placing it on the countertop. This makes it difficult to train long, heterogeneous actions as a whole. To efficiently capture those characteristics, we propose to model each action as a sequential combination of subactions. Therefore, for each action class  $a$ , a set of subactions  $s_1^{(a)}, \dots, s_{K_a}^{(a)}$  is defined. The number  $K_a$  is initially estimated by a heuristic and refined during the optimization process. Practically, this means that we subdivide the original long action classes into a set of smaller subactions. As subactions are obviously not defined by the given ordered action sequences, we treat them as latent variables that need to be learned by the model. In the following system description, we assume that the subaction frame boundaries are known, e.g. from previous iterations or from an initial uniform segmentation (see Figure 4b), and discuss the inference of more accurate boundaries in Section 3.4.

#### 3.2. Coarse Action Model

In order to combine the fine grained subactions to action sequences, a hidden Markov model  $\mathcal{H}_a$  for each action  $a$  is defined. The HMM ensures that subactions only occur in the correct ordering, i.e. that  $s_i^{(a)} \prec s_j^{(a)}$  for  $i \leq j$ . More precisely, let

$$s(t) : \{1, \dots, T\} \mapsto \{s_1^{(a_1)}, \dots, s_{K_{a_N}}^{(a_N)}\} \quad (3)$$

be the known mapping from video frames to the subactions of the ordered action sequence  $\mathbf{a}_1^N$ . This is basically the same mapping as the one in Equation (1) but on subaction level rather than on action level. When going from one frame to the next, we only allow to assign either the same subaction or the next subaction, so if at frame  $t$ , the assigned subaction is  $s(t) = s_i^{(a)}$ , then at frame  $t+1$ , either  $s(t+1) = s_i^{(a)}$  or  $s(t+1) = s_{i+1}^{(a)}$ . The likelihood of the video frames  $\mathbf{x}_1^T$  given the action transcripts  $\mathbf{a}_1^N$  is then

$$p(\mathbf{x}_1^T | \mathbf{a}_1^N) := \prod_{t=1}^T p(x_t | s(t)) \cdot p(s(t) | s(t-1)), \quad (4)$$

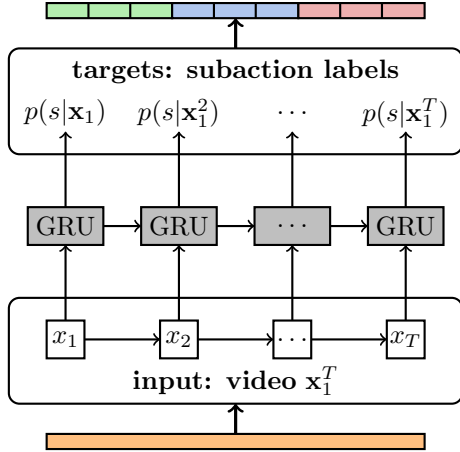


Figure 2. RNN using gated recurrent units with framewise video features as input. At each frame, the network outputs a probability for each possible subaction while considering the context of the video.

where  $p(x_t|s)$  are probabilities computed by the fine-grained model, see Section 3.3. The transition probabilities  $p(s|s')$  from subaction  $s'$  to subaction  $s$  are relative frequencies of how often the transition  $s' \rightarrow s$  occurs in the  $s(t)$ -mappings of all training videos.

### 3.3. Fine-grained Subaction Model

For the classification of fine-grained subactions, we use an RNN with a single hidden layer of gated recurrent units (GRUs) [4]. It is a simplified version of LSTMs that shows comparable performance [11, 5] also in case of video classification [2]. The network is shown in Figure 2.

For each frame, it predicts a probability distribution over all subactions, while the recurrent structure of the network allows to incorporate local temporal context. Since the RNN generates a posterior distribution  $p(s|x_t)$  but our coarse model deals with subaction-conditional probabilities, we use Bayes' rule to transform the network output to

$$p(x_t|s) = \text{const} \cdot \frac{p(s|x_t)}{p(s)}. \quad (5)$$

**Solving Efficiency Issues.** Recurrent neural networks are usually trained using backpropagation through time (BPTT) [31], which requires to process the whole sequence in a forward and backward pass. As videos can be very long and may easily exceed 10,000 frames, the computation time per minibatch can be extremely high. Even worse, long videos may not fit the memory of high-end GPUs, since during training the output of all network layers needs to be stored for each frame of the video in order to compute the gradient.

We tackle this problem by using small chunks around each video frame that can be processed efficiently and with a reasonably large minibatch size in order to enable efficient RNN training on long videos. For each frame  $t$ , we

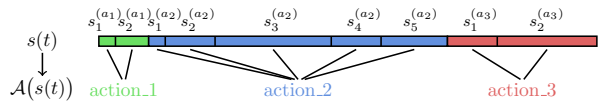


Figure 3. The extractor function  $\mathcal{A}$  computes the unique action sequence induced by the frame-to-subaction alignment  $s(t)$ .

create a chunk over  $\mathbf{x}[t-20, t]$  and forward it through the RNN. While this practically increases the amount of data that needs to be processed by a factor of 20, only short sequences need to be forwarded at once and we benefit from a high parallelization degree and comparable large minibatch size.

Additionally, one has to note that even LSTMs and GRUs can only capture a limited amount of temporal context. For instance, studies from machine translation suggest that 20 frames is a range that can be well captured by these architectures [4]. This finding is confirmed for video data in [27]. Also, humans usually do not need much more context to accurately classify a part of an action. Hence, storing the information of *e.g.* frame 10 while computing the output of frame 500 is not necessary. Thus, it can be appropriate to limit the temporal scope in favor of a faster, more feasible training.

### 3.4. Inference

Based on the observation probabilities of the fine-grained subaction model and the coarse model for overall actions, we will now discuss the combined inference of both models on video level.

Given a video  $\mathbf{x}_1^T$ , the most likely action sequence

$$\hat{\mathbf{a}}_1^N = \arg \max_{\mathbf{a}_1^N} \{p(\mathbf{x}_1^T | \mathbf{a}_1^N) \cdot p(\mathbf{a}_1^N)\} \quad (6)$$

and the corresponding frame alignment is to be found. In order to limit the amount of action sequences to optimize over, a context-free grammar  $\mathcal{G}$  is created from the training set as in [16]. We set  $p(\mathbf{a}_1^N) = 1$  if  $\mathbf{a}_1^N$  is generated by  $\mathcal{G}$  and  $p(\mathbf{a}_1^N) = 0$  otherwise. Thus, in Equation (6), the  $\arg \max$  only needs to be taken over action sequences generated by  $\mathcal{G}$  and the factor  $p(\mathbf{a}_1^N)$  can be omitted. Instead of finding the optimal action sequence directly, the inference can equivalently be performed over all possible frame-to-subaction alignments  $s(t)$  that are consistent with  $\mathcal{G}$ . Consistent means that the unique action sequence defined by  $s(t)$  is generated by  $\mathcal{G}$ . Formally, we define an extractor function  $\mathcal{A} : s(t) \mapsto \mathbf{a}_1^N$  that maps the frame-to-subaction alignment  $s(t)$  to its action sequence, see Figure 3 for an illustration. Equation (6) can then be rewritten as

$$\hat{\mathbf{a}}_1^N = \arg \max_{s(t): \mathcal{A}(s(t)) \in \mathcal{L}(\mathcal{G})} \left\{ \prod_{t=1}^T p(x_t | s(t)) \cdot p(s(t) | s(t-1)) \right\}, \quad (7)$$



where  $\mathcal{L}(\mathcal{G})$  is the set of all possible action sequences that can be generated by  $\mathcal{G}$ . Note that Equation (7) can be solved efficiently using a Viterbi algorithm if the grammar is context-free, see *e.g.* [10].

For training, as well as for the task of aligning videos to a given ordered action sequence  $\mathbf{a}_1^N$ , the best frame alignment to a single sequence needs to be inferred. By defining a grammar that generates only the given action sequence  $\mathbf{a}_1^N$ , this alignment task can be solved using Equation (7). For the task of temporal action segmentation, *i.e.* when no action sequence is provided for inference, the context-free grammar can be derived from the ordered action sequences given in the training samples.

### 3.5. Training

Training of the model is an iterative process, altering between both, the recurrent neural network and the HMM training, and the alignment of frames to subaction units via the HMM. The whole process is illustrated in Figure 4.

**Initialization.** The video is divided into  $N$  segments of equal size, where  $N$  is the number of action instances in the transcript (Figure 4a). Each action segment is further subdivided equally across the subactions (Figure 4b). Note that this defines the mapping  $s(t)$  from frames to subactions. Each subaction should cover  $m$  frames of an action on average. Thus, the initial number of subactions for each action is

$$\frac{\text{number of frames}}{\text{number of action instances} \cdot m}, \quad (8)$$

where we usually choose  $m = 10$  as proposed in [15, 16]. Hence, initially each action is modeled with the same number of subactions. This can change during the iterative optimization.

**Iterative Training.** The fine-grained RNN is trained with the current mapping  $s(t)$  as ground truth. Then, the RNN and HMM are applied to the training videos and a new alignment of frames to subactions (Figure 4d) is inferred given the new fine-grained probabilities  $p(x_t|s)$  from the RNN. The new alignment is obtained by finding the subaction mapping  $s(t)$  that best explains the data:

$$\hat{s}(t) = \arg \max_{s(t)} \left\{ p(\mathbf{x}_1^T | \mathbf{a}_1^N) \right\} \quad (9)$$

$$= \arg \max_{s(t)} \left\{ \prod_{t=1}^T p(x_t | s(t)) \cdot p(s(t) | s(t-1)) \right\}. \quad (10)$$

Note that Equation (10) can be efficiently computed using a Viterbi algorithm. Once the realignment is computed for all training videos, the average length of each action is reestimated as

$$\text{len}(a) = \frac{\text{number of frames aligned to } a}{\text{number of } a\text{-instances}} \quad (11)$$

and the number of subactions is reestimated based on the updated average action lengths. Particularly, for action  $a$ , there are now  $\text{len}(a)/m$  subactions, which are again uniformly distributed among the frames assigned to the corresponding action, *cf.* Figure 4e. These steps are iterated until convergence.

**Stop Criterion.** As the system iteratively approximates the optimal action segmentation on the training data, we define a stop criterion based on the overall amount of action boundaries shifted from one iteration to the succeeding one. In iteration  $i$ , let  $\text{change}(i)$  denote the percentage of frames that is labeled differently compared to iteration  $i - 1$ . We stop the optimization if the frame change rate between two iterations is less than a threshold,

$$|\text{change}(i) - \text{change}(i - 1)| < \vartheta \Rightarrow \text{stop}. \quad (12)$$

## 4. Experiments

In this section, we provide a detailed analysis of our method. Code and models are available online<sup>1</sup>.

### 4.1. Setup

**Datasets.** We evaluate the proposed approach on two heterogeneous datasets. The Breakfast dataset is a large scale dataset with 1,712 clips and an overall duration of 66.7 hours. The dataset comprises various kitchen tasks such as making tea but also complex activities such as the preparation of fried egg or pancake. It features 48 action classes with a mean of 4.9 instances per video. We follow the evaluation protocol as proposed by the authors in [14].

The Hollywood extended [3] dataset is an extension of the well known Hollywood dataset, featuring 937 clips from different Hollywood movies. The clips are annotated with two or more action labels resulting in 16 different action classes overall and a mean of 2.5 action instances per clip.

**Features.** For both datasets we follow the feature computation as described in [15] using improved dense trajectories (IDT) and Fisher vectors (FVs). To compute the FV representation, we first reduce the dimensionality of the IDT features from 426 to 64 by PCA and sample 150,000 randomly selected features to build a GMM with 64 Gaussians. The Fisher vector representation [25] for each frame is computed over a sliding window of 20 frames. Following [22], we apply power and l2 normalization to the resulting FV representation. Additionally, we reduce the final FV representation from 8,192 to 64 dimensions via PCA to keep the overall video representation manageable and easier to process.

**Stop Criterion.** For the stop criterion, we fix  $\vartheta = 0.02$ , *i.e.* if the difference of the frame change between two iterations is less than two percent, we stop iterating. Figure 5

<sup>1</sup><https://github.com/alexanderrichard/weakly-sup-action-learning>

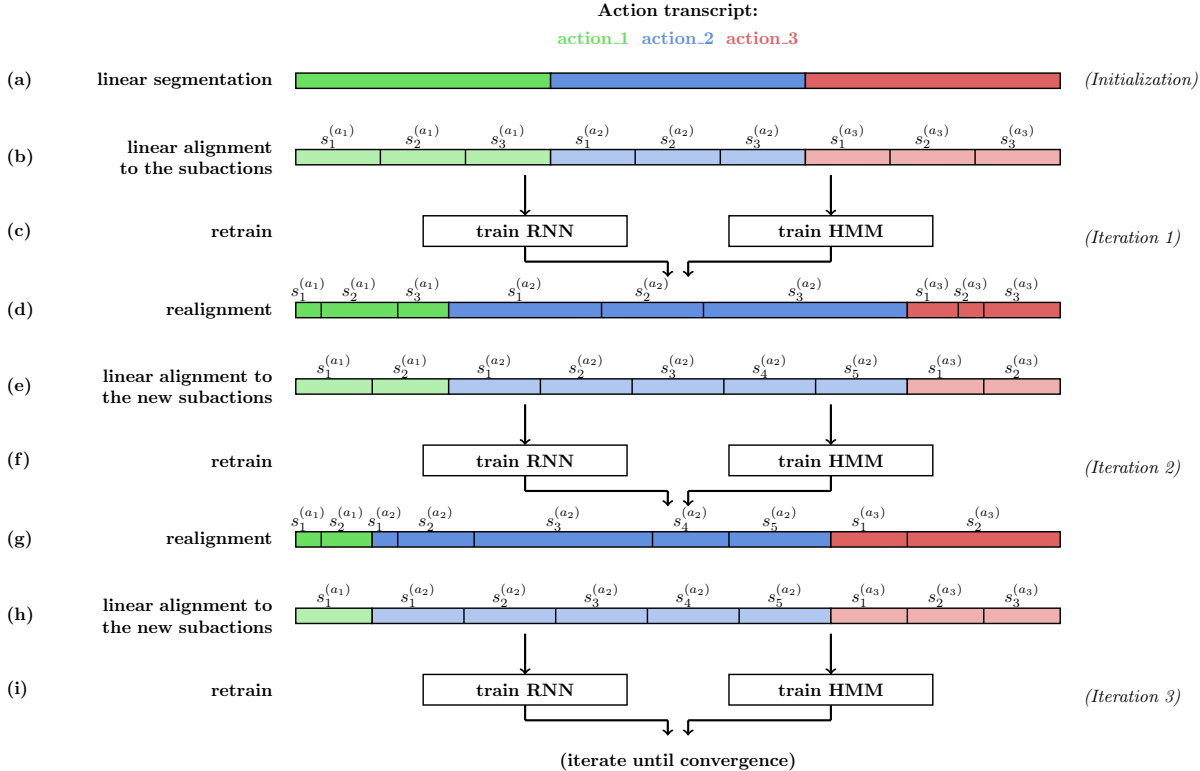


Figure 4. Training process of our model. Initially, each action is modeled with the same number of subactions and the video is linearly aligned to these subactions. Based on this alignment, the RNN is trained and used in combination with the HMMs to realign the video frames to the subactions. Eventually, the number of subactions per action is reestimated and the process is iterated until convergence.

Breakfast	Accuracy (Mof)
<i>GRU no subactions</i>	22.4
<i>GRU w/o reestimation</i>	28.8
<i>GRU + reestimation</i>	33.3
<i>GRU + GT length</i>	51.3

Table 1. Results for temporal segmentation on the Breakfast dataset comparing accuracy of the proposed system (GRU + reestimation) to the accuracy of the same architecture without subactions (GRU no subactions) and to the architecture with subclasses but without reestimation.

illustrates the criterion for two example experiments. The blue curve is the frame accuracy, the red curve is the difference of frame changes between two iterations. It can be seen that after a few iterations, the frame accuracy does not increase anymore but begins to oscillate, see also Table 2. Comparing the frame change rate of the train data is a good indicator of when to stop iterating. In all experiments, we compute results based on the alignment of the last iteration before the threshold  $\vartheta$  is crossed.

## 4.2. Analysis of the Coarse Model

For the following evaluation of our system, we report performance for the task of temporal action segmentation, *i.e.* the combined video segmentation and classification.

Given a video without any further information, the task is to classify all frames according to their related action. This includes to infer which actions occur in the video, in which order they occur, and their respective start and end frames. We evaluate on the test set of the Breakfast dataset and report results as mean accuracy over frames (Mof) (see [14]). We iterate the system until the stop criterion as described in Section 3.5 is reached.

First, we regard the properties of the coarse action modeling. We therefore compare the proposed system to the results of the same setting, but without further subdividing actions into subactions (GRU no subactions, Table 1). Additionally, we regard results of the system without reestimation of subactions during optimization (GRU w/o reestimation, Table 1). For the system without reestimation, we follow the initial steps as shown in Figure 4, thus, we linearly segment the videos according to the number of actions, generate an initial subaction alignment, train the respective subaction classes, and realign the sequence based on the RNN output. But, opposed to the setup with reestimation, we omit the step of reestimating the number of subclasses and the following alignment. Instead we just use the output of the realignment to retrain the classifier and iterate the process of training, alignment, and re-training. Thus,

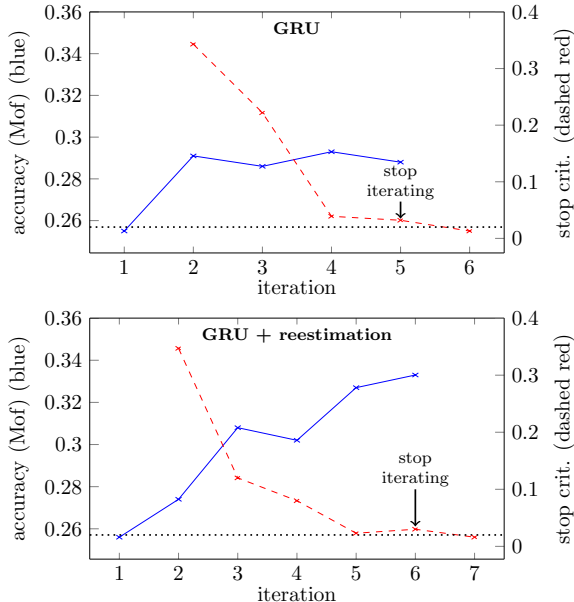


Figure 5. Stop criterion for two experiments, one using the fine-grained model and no subaction reestimation, one using it with subaction reestimation. The blue curve shows the frame accuracy over the iterations, the red curve shows the frame change rate between the current and the preceding iteration. The dashed line represents the threshold  $\vartheta = 0.02$ .

the number of subclasses is constant and the coarse model is not adapted to the overall estimated length of the action class. Finally, we compare against a system in which we use the ground truth boundaries to compute the mean length of an action class and set the number of subactions based on the mean ground truth length (GRU + GT length, Table 1). Here, all action classes are still uniformly initialized, but longer action classes are divided into more subactions than shorter ones. We include this artificial scenario as it models the performance in case that the optimal number of subaction classes would be found.

One can see in Table 1 that recognition performance without subactions is significantly below all other configurations, supporting the idea that subaction modeling in general helps recognition in this scenario. The model with subactions, but without reestimation, improves over the single class model, but is still below the system with subaction reestimation. Compared to that, the model with subaction reestimation performs 5% better. We ascribe the performance increase of the reestimated model to the fact that a good performance is highly related to the correct number of subactions, thus to a good length representation of the single actions. By reestimating the overall number of subactions after each iteration, this ground truth length is approximated. The impact of the number of subactions becomes clear, when considering the results when the ground truth action lengths are used. Here the performance of the

Breakfast	Iter 1	Iter 2	Iter 3	Iter 4	Iter 5
<i>GMM w/o reest.</i>	15.3	23.3	26.3	27.0	26.5
<i>MLP w/o reest.</i>	22.4	24.0	23.7	23.1	20.3
<i>GRU w/o reest.</i>	25.5	29.1	28.6	29.3	28.8
<i>GRU w/o HMM</i>	21.3	20.1	23.8	21.8	22.4

Table 2. Results for low level recognition with an MLP compared to GRUs over five iterations. The MLP quickly starts to overfit, whereas the GRU oscillates at a constant level. Last row: A GRU without HMM, showing that short term dependencies are well captured by the fine-grained recurrent network.

same system, just with different numbers of subactions, increases by almost 20%. Qualitative results on two example videos are shown in Figure 6.

### 4.3. Analysis of the Fine-Grained Model

In order to analyze the capability of capturing temporal context with the recurrent network, we compare it to a system where a Gaussian mixture model (GMM) and a multi-layer perceptron (MLP) are used instead. Both only operate on frame level and do not capture fine-grained information between the frames. In order to provide a fair comparison to the recurrent model, we equip the MLP with a single hidden layer of rectified units such that it has the same number of parameters as the recurrent network. We use a simplified version of the system without subaction reestimation to achieve comparable results after each iteration.

Results for the first five iterations on the Breakfast dataset are shown in Table 2. We find that GRUs clearly outperform both, GMMs and MLPs, starting with 25.5% for the initial recognition, and reaching up to 29.3% after the fourth iteration. Particularly, the MLP baseline stays continuously below this performance. Thus, it can be assumed that the additional information gained by recurrent connections in this context supports classification. One can further see that the MLP reaches its best performance after the second iteration and then continuously decreases, whereas the GRU begins to oscillate around 29%, hinting that the MLP also starts to overfit at an earlier stage compared to the GRU. In the last row of Table 2, the coarse model, *i.e.* the HMM, is removed from the system. Thus, there is no modeling of subactions anymore and the GRU directly learns the original action classes. The performance is clearly worse than with the coarse model, but still the system is remarkably good, being on par with the MLP that uses the coarse model (second row of Table 2).

### 4.4. Comparison to State-of-the-Art

**Temporal Action Segmentation.** We compare our system to three different approaches published for the task: The first is the Ordered Constrained Discriminative Clustering (OCDC) proposed by Bojanowski *et al.* [3], which has been introduced on the Hollywood extended dataset. Sec-

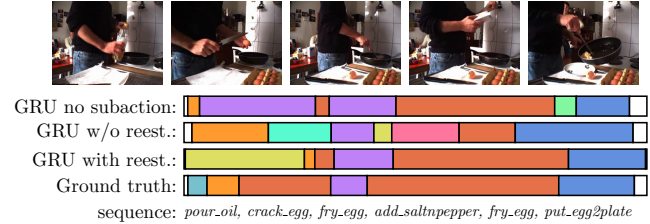
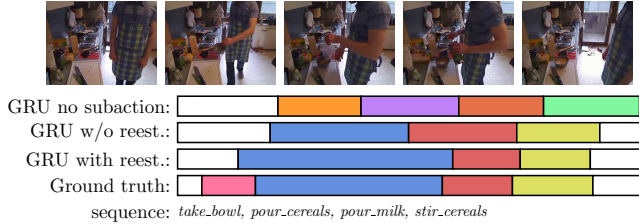


Figure 6. Example segmentation results for two samples from the Breakfast dataset showing the segmentation result for “preparing cereals” and “preparing friedegg”. Although the actions are not always correctly detected, there is still reasonable alignment of detected actions and ground truth boundaries.

	Breakfast	Hollywood Ext.
Model	Accuracy (Mof)	Jacc. (IoU)
OCDC [3]*	8.9	-
HTK [16]	25.9	8.6
ECTC [9]	27.7	-
GRU w/o reestimation	28.8	11.2
GRU + reestimation	<b>33.3</b>	<b>11.9</b>

Table 3. Comparison of temporal action segmentation performance for GRU based weak learning with other approaches. For the Breakfast dataset, we report performance as mean over frames (Mof), for Hollywood extended, we measure the Jaccard index as intersection over union for this task (\*from [9]).

ond, we compare against the HTK system used by Kuehne *et al.* [16], and third against the Extended Connectionist Temporal Classification (ECTC) by Huang *et al.* [9]. For the Breakfast dataset, we follow the evaluation script of [14, 9] and report results as mean accuracy over frames over four splits. For the Hollywood Extended dataset, we follow the evaluation script of [16] and report the Jaccard index (Jacc.) as intersection over union (IoU) over 10 splits. Results are shown in Table 3.

One can see that the proposed subaction based GRU systems show a good performance, and that both subaction based systems outperform current approaches on the two evaluated datasets. It also shows that the GRU based system without reestimation shows comparable performance to other RNN based systems, such as ECTC by [9], which uses an LSTM model with comparable size. The significant increase in accuracy of our method can be attributed to the reestimation. We also observe that the performance boost of the system with reestimation is more prominent on the Breakfast than on the Hollywood extended dataset. We attribute this to the fact that in case of Hollywood extended, all action classes usually have a consistent mean frame length, whereas in case of Breakfast, the mean length of action classes significantly varies. Thus, the benefit of adapting to action class lengths increases with the heterogeneity of the target action classes.

**Action Alignment.** We also address the task of action alignment. Here, we assume that given a video and a sequence of temporally ordered actions, the task is to infer the

	Breakfast	Hollywood Ext.
Model	Jacc. (IoD)	Jacc. (IoD)
OCDC [3]	23.4	43.9
HTK [16]**	40.6	42.4
ECTC [9]**	-	41.0
GRU w/o reestimation	41.5	45.6
GRU + reestimation	<b>47.3</b>	<b>46.3</b>

Table 4. Results for action alignment on the test set of the Breakfast and the Hollywood extended dataset reported as Jaccard index of intersection over detection (IoD) (\*\*results obtained from the authors).

respective boundaries for the given action order. We report results for the test set of Breakfast as well as for the Hollywood extended dataset based on the Jaccard index (Jacc.) computed as intersection over detection (IoD) as proposed by [3]. The results are shown in Table 4.<sup>2</sup> Here, the GRU system without reestimation performs on par with other systems for the alignment task on the Breakfast dataset, but the GRU system with reestimation again shows a clear improvement over current systems.

## 5. Conclusion

We presented an approach for weakly supervised learning of human actions based on a combination of a discriminative representation of subactions, modeled by a recurrent neural network, and a coarse probabilistic model to allow for a temporal alignment and inference over long sequences. Although the system itself already shows good results, the performance is significantly improved by approximating the number of subactions for the different action classes. Accordingly, we propose to adapt the number of subaction classes by iterating realignment and reestimation during training. The resulting models outperform state-of-the-art on various weak learning tasks such as temporal action segmentation and action alignment.

**Acknowledgments.** The work has been financially supported by the DFG projects KU 3396/2-1 (Hierarchical

<sup>2</sup>Errata: The results for Hollywood Extended have been corrected. 45.6 and 46.3 are the correct numbers, the results 50.1 and 51.1 from the original CVPR paper are incorrect.



Models for Action Recognition and Analysis in Video Data) and GA 1927/4-1 (DFG Research Unit FOR 2535 Anticipating Human Behavior) and the ERC Starting Grant ARCA (677650).

## References

- [1] J.-B. Alayrac, P. Bojanowski, N. Agrawal, I. Laptev, J. Sivic, and S. Lacoste-Julien. Unsupervised learning from narrated instruction videos. In *IEEE Conf. on Computer Vision and Pattern Recognition*, 2016. 1, 2
- [2] N. Ballas, L. Yao, P. Chris, and A. Courville. Delving deeper into convolutional networks for learning video representations. In *Int. Conf. on Learning Representations*, 2016. 4
- [3] P. Bojanowski, R. Lajugie, F. Bach, I. Laptev, J. Ponce, C. Schmid, and J. Sivic. Weakly supervised action labeling in videos under ordering constraints. In *European Conf. on Computer Vision*, 2014. 1, 2, 5, 7, 8
- [4] K. Cho, B. van Merriënboer, D. Bahdanau, and Y. Bengio. On the properties of neural machine translation: Encoder-decoder approaches. In *Proceedings of SSST@EMNLP, Eighth Workshop on Syntax, Semantics, and Structure in Statistical Translation*, pages 103–111, 2014. 4
- [5] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014. 4
- [6] J. Donahue, L. A. Hendricks, S. Guadarrama, M. Rohrbach, S. Venugopalan, K. Saenko, and T. Darrell. Long-term recurrent convolutional networks for visual recognition and description. In *IEEE Conf. on Computer Vision and Pattern Recognition*, 2015. 1
- [7] O. Duchenne, I. Laptev, J. Sivic, F. Bach, and J. Ponce. Automatic annotation of human actions in video. In *Int. Conf. on Computer Vision*, 2009. 1, 2
- [8] C. Gan, C. Sun, L. Duan, and B. Gong. Webly-supervised video recognition by mutually voting for relevant web images and web video frames. In *European Conf. on Computer Vision*, 2016. 2
- [9] D.-A. Huang, L. Fei-Fei, and J. C. Niebles. Connectionist temporal modeling for weakly supervised action labeling. In *European Conf. on Computer Vision*, 2016. 1, 2, 8
- [10] D. Jurafsky, C. Wooters, J. Segal, A. Stolcke, E. Fosler, G. Tajchaman, and N. Morgan. Using a stochastic context-free grammar as a language model for speech recognition. In *IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, volume 1, pages 189–192, 1995. 5
- [11] R. Jzefowicz, W. Zaremba, and I. Sutskever. An empirical exploration of recurrent network architectures. In *Int. Conf. on Machine Learning*, 2015. 4
- [12] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei. Large-scale video classification with convolutional neural networks. In *IEEE Conf. on Computer Vision and Pattern Recognition*, pages 1725–1732, 2014. 1
- [13] O. Koller, H. Ney, and R. Bowden. Deep hand: How to train a cnn on 1 million hand images when your data is continuous and weakly labelled. In *IEEE Conf. on Computer Vision and Pattern Recognition*, 2016. 2
- [14] H. Kuehne, A. B. Arslan, and 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. 2, 5, 6, 8
- [15] H. Kuehne, J. Gall, and T. Serre. An end-to-end generative framework for video segmentation and recognition. In *IEEE Winter Conf. on Applications of Computer Vision*, 2016. 5
- [16] H. Kuehne, A. Richard, and J. Gall. Weakly supervised learning of actions from transcripts. *arXiv preprint arXiv:1610.02237*, 2016. 2, 4, 5, 8
- [17] I. Laptev, M. Marszalek, C. Schmid, and B. Rozenfeld. Learning realistic human actions from movies. In *IEEE Conf. on Computer Vision and Pattern Recognition*, 2008. 1, 2
- [18] C. Lea, A. Reiter, R. Vidal, and G. D. Hager. Segmental spatiotemporal cnns for fine-grained action segmentation. In *European Conf. on Computer Vision*, pages 36–52, 2016. 2
- [19] J. Malmaud, J. Huang, V. Rathod, N. Johnston, A. Rabinovich, and K. Murphy. Whats cooking? interpreting cooking videos using text, speech and vision. In *Conf. of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2015. 3
- [20] M. Marszalek, I. Laptev, and C. Schmid. Actions in context. In *IEEE Conf. on Computer Vision and Pattern Recognition*, 2009. 1, 2
- [21] I. Misra, C. L. Zitnick, and M. Hebert. Shuffle and learn: Unsupervised learning using temporal order verification. In *European Conf. on Computer Vision*, 2016. 3
- [22] F. Perronnin, J. Sánchez, and T. Mensink. Improving the fisher kernel for large-scale image classification. In *European Conf. on Computer Vision*, 2010. 5
- [23] A. Richard and J. Gall. Temporal action detection using a statistical language model. In *IEEE Conf. on Computer Vision and Pattern Recognition*, 2016. 1
- [24] M. Rohrbach, S. Amin, M. Andriluka, and B. Schiele. A database for fine grained activity detection of cooking activities. In *IEEE Conf. on Computer Vision and Pattern Recognition*, pages 1194–1201, 2012. 1
- [25] J. Sanchez, F. Perronnin, T. Mensink, and J. Verbeek. Image Classification with the Fisher Vector: Theory and Practice. *International Journal on Computer Vision*, 105(3):222–245, Dec. 2013. 5
- [26] K. Simonyan and A. Zisserman. Two-stream convolutional networks for action recognition in videos. In *Advances in Neural Information Processing Systems*, pages 568–576, 2014. 1
- [27] B. Singh, T. K. Marks, M. Jones, O. Tuzel, and M. Shao. A multi-stream bi-directional recurrent neural network for fine-grained action detection. In *IEEE Conf. on Computer Vision and Pattern Recognition*, 2016. 1, 4
- [28] C. Sun, S. Shetty, R. Sukthankar, and R. Nevatia. Temporal localization of fine-grained actions in videos by domain transfer from web images. In *ACM Conf. on Multimedia*, 2015. 2
- [29] K. Tang, L. Fei-Fei, and D. Koller. Learning latent temporal structure for complex event detection. In *IEEE Conf. on Computer Vision and Pattern Recognition*, pages 1250–1257, 2012. 2

- [30] H. Wang and C. Schmid. Action recognition with improved trajectories. In *Int. Conf. on Computer Vision*, pages 3551–3558, 2013. [1](#)
- [31] P. J. Werbos. Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78:1550–1560, 1990. [4](#)
- [32] C. Wu, J. Zhang, S. Savarese, and A. Saxena. Watch-n-patch: Unsupervised understanding of actions and relations. In *IEEE Conf. on Computer Vision and Pattern Recognition*, 2015. [3](#)
- [33] Z. Wu, X. Wang, Y.-G. Jiang, H. Ye, and X. Xue. Modeling spatial-temporal clues in a hybrid deep learning framework for video classification. In *ACM Conf. on Multimedia*, 2015. [1](#)
- [34] S. Yeung, O. Russakovsky, G. Mori, and L. Fei-Fei. End-to-end learning of action detection from frame glimpses in videos. In *IEEE Conf. on Computer Vision and Pattern Recognition*, 2016. [1](#), [2](#)
- [35] J. Yue-Hei Ng, M. Hausknecht, S. Vijayanarasimhan, O. Vinyals, R. Monga, and G. Toderici. Beyond short snippets: Deep networks for video classification. In *IEEE Conf. on Computer Vision and Pattern Recognition*, 2015. [1](#)