Action Shuffle Alternating Learning for Unsupervised Action Segmentation

Jun Li
Oregon State University
lijju2@oregonstate.edu

Sinisa Todorovic
Oregon State University
sinisa@oregonstate.edu

Abstract

This paper addresses unsupervised action segmentation. Prior work captures the frame-level temporal structure of videos by a feature embedding that encodes time locations of frames in the video. We advance prior work with a new self-supervised learning (SSL) of a feature embedding that accounts for both frame- and action-level structure of videos. Our SSL trains an RNN to recognize positive and negative action sequences, and the RNN's hidden layer is taken as our new action-level feature embedding. The positive and negative sequences consist of action segments sampled from videos, where in the former the sampled action segments respect their time ordering in the video, and in the latter they are shuffled. As supervision of actions is not available and our SSL requires access to action segments, we specify an HMM that explicitly models action lengths, and infer a MAP action segmentation with the Viterbi algorithm. The resulting action segmentation is used as pseudo-ground truth for estimating our action-level feature embedding and updating the HMM. We alternate the above steps within the Generalized EM framework, which ensures convergence. Our evaluation on the Breakfast, YouTube Instructions, and 50Salads datasets gives superior results to those of the state of the art.

1. Introduction

This paper is about unsupervised action segmentation, where the goal is to localize salient latent actions in untrimmed videos. The actions are salient as differences in their features allow for segmentation, and the actions are latent as they may not have a distinct semantic meaning, since no supervision about the actions is available. The ground truth can be used only for evaluation. This is a long-standing vision problem with a wide range of applications. It can be used for mapping a long video to a significantly shorter sequence of action segments, and thus for facilitating and reducing complexity of subsequent video interpretation. It can also be used in applications where manual video annotation is prohibitively expensive or not reliable.

In this paper, we focus on a particular setting studied in recent work [18], where all videos show the same activity (e.g., a cooking activity) which can be decomposed into a temporal sequence of simpler actions (e.g., cooking includes cutting, mixing, peeling). While the activity exhibits variations across videos, they are mostly manifested in varying lengths and features of each action of the activity, whereas variations in the total number and temporal ordering of actions are limited by the very nature of the activity (e.g., cooking usually requires a certain order of actions).

For such a setting, related work [18] makes the following restrictive assumptions that every action appears only once, and all actions always occur and follow the same temporal ordering in all videos, referred to as *fixed transcript*. Based on these assumptions, they learn a temporal feature embedding by training a regression model to predict a temporal position of every frame in the video, where the frame positions are normalized to the video length. Then, they use the K-means for clustering these embedded features of all video frames, and interpret the resulting clusters as representing the latent actions. After computing a likelihood for every cluster, they run the standard Viterbi algorithm on every video for localizing the latent actions.

We make *three contributions*, as illustrated in Fig. 1. First, we relax the above assumption about the fixed temporal ordering of actions in videos. We specify a Hidden Markov Model (HMM), and thus infer a MAP ordering of actions, instead of the fixed transcript. Unlike [18], our HMM explicitly models the varying lengths of latent actions, and thus constrains implausible solutions in the domain (e.g., chopping cannot take a few frames). Also, our HMM uses a multilayer perceptron (MLP) for estimating the likelihood of the frame labeling with the latent actions.

Second, we specify a new self-supervised learning (SSL) for *action-level temporal feature embedding*. As other SSL approaches [18, 22, 13, 19, 5, 14, 8, 3, 33], ours exploits the temporal structure of videos, where the structure we mean that videos are sequences of actions with small variations in action ordering. However, the cited references typically focus on capturing the temporal structure at the frame level (e.g., by shuffling or permuting frames [22, 19], encoding

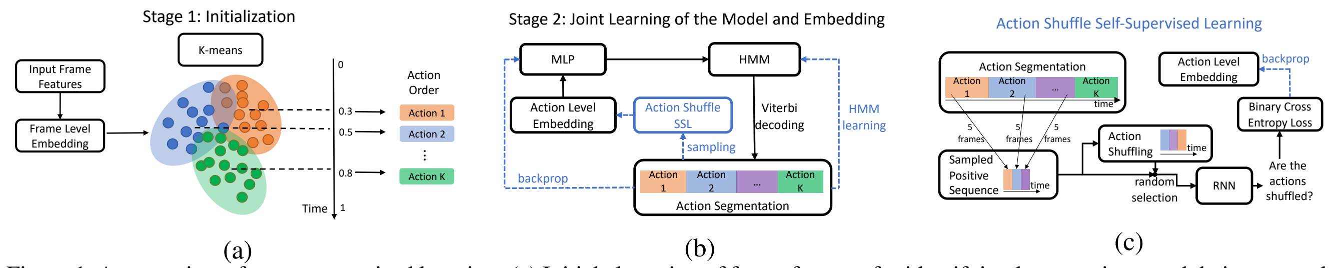

Figure 1. An overview of our unsupervised learning. (a) Initial clustering of frame features for identifying latent actions, and their temporal ordering; the frame-level feature embedding is from [18]. (b) The iterative joint training of HMM, MLP, and Action Shuffle SSL for inferring action segmentation within the generalized EM framework. (c) SSL for learning our action-level temporal feature embedding. The data manipulation samples positive and negative sequences, where the former respect temporal ordering of actions in the predicted action segmentation and the latter shuffle the ordering of actions.

frames' positions in video [18]) or the entire video level (e.g., by manipulating the playback speed [8, 3, 33]). In contrast, we seek to learn a feature embedding that would account for the correct ordering of actions along the video.

As illustrated in Fig. 1 and Fig. 2, our SSL is based on recognizing positive and negative sequences of actions, both generated as sequences of action segments randomly sampled from a video. The key difference between the positive and negative sequences is that in the former the actions are laid out in the same temporal ordering as in the video, and in the latter the actions are shuffled resulting in an incorrect ordering. As shown in Fig. 1, our SSL trains an RNN on these positive and negative sequences, and minimization of the incurred binary cross-entropy loss produces our action-level temporal embedding of frame features.

As our third contribution, we formulate a joint training of HMM, MLP, and Action Shuffle SSL within the Generalized EM framework [28]. This extends prior work (e.g. [18, 32]) where different components of their approaches are usually learned independently. Note that our Action Shuffle SSL requires access to action segments for generating the positive and negative examples. Action segmentation, in turn, requires the HMM inference. After the HMM inference, we can update the action-level feature embedding through the Action Shuffle SSL, as well as update the HMM parameters. As these updates will change both features and HMM, it seems reasonable to run the HMM inference again. We integrate all these steps within the generalized EM framework. A convergence guarantee of our joint training follows from the generalized EM algorithm [28].

Fig. 1 shows an overview of our unsupervised learning that consists of two stages. The initial stage follows [18]. Given frame features from all videos, we first compute the frame-level feature embedding of [18], and then run the K-means for identifying labels of the latent actions. As shown in Fig. 1a, for every cluster, we compute the mean of frames' normalized positions in videos, then, sort the clusters by their means in the ascending order, and finally take the sorted cluster indices as labels of the corresponding latent actions. This ascending order of action labels is taken as a likely transcript, and used in the HMM inference for constraining the Viterbi algorithm to respect the transcript. In the second stage, we perform the generalized EM algorithm for iteratively updating the HMM, MLP, and action-level embedding.

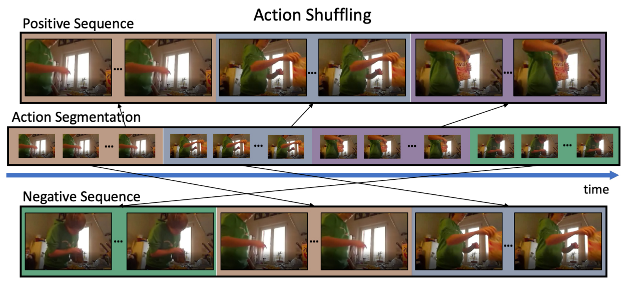

Figure 2. Action shuffling: Both positive and negative examples are generated as sequences of 3 action segments randomly sampled from a video, where the former respects the action ordering in the video, and the latter shuffles the ordering.

Evaluation on the challenging Breakfast [15], YouTube Instructional [2], and 50Salads [27] datasets demonstrate our superior performance over the state of the art.

The rest of the paper is organized as follows: Sec. 2 reviews related work, Sec. 3 presents our SSL, Sec. 4 specifies our HMM and its inference, Sec. 5 formalizes our joint training, Sec. 6 extends our approach to a more general setting, and Sec. 7 presents our experiments.

2. Related Work

This section reviews closely related work on action segmentation under a reduced level of supervision.

Transcript supervised action segmentation. In this problem, we have access to the true action ordering in training videos, but their exact action boundaries are unknown. While this problem is different from ours, our approach draws motivation from recent advances. For example, Extended Connectionist Temporal Classification (ECTC) regularizes action alignment with consistency of frame similarity [11]. Some approaches alternatively align actions with frames and update their action models [17, 23]. Other methods first estimate a video segmentation, and then use this estimation to train a classifier for frame labeling [25, 20].

Set supervised action segmentation. In this problem, we have access to the ground-truth set of actions occurring in the training video, but we do not know their ordering and temporal extents. This problem was first addressed with multi-instance learning [24]. In [21], Set-Constrained Viterbi (SCV) algorithm is used to produce pseudo-ground-truth labels of frames for the subsequent fully supervised action segmentation. In [9], Set Constrained Temporal Transformation (SCT) is used to predict action labels of oversegmented temporal regions.

Unsupervised action segmentation and detection focuses on exploiting the temporal structure of videos [32, 29, 10, 26, 12]. For example, [10] detects all pairs of matching video segments, [32] learns co-occurrence and temporal relations between actions, [26] proposes a Generalized Mallows Model to jointly learn action appearance and temporal structure, [12] groups consecutive frames to form communities similar to social networks. Sec. 1 summarizes [18] as the most closely related approach to ours, and points out our differences and extensions.

Temporal SSL has been recently used in various video interpretation problems, including action segmentation, for learning a temporal feature embedding. Existing temporal SSL methods are typically aimed at capturing the temporal structure of video either at the frame level or at the entire video level. Examples of frame-level temporal SSL include the following: [22] determines whether a sequence of frames from a video is shuffled or in the correct temporal order; [19] predicts a permutation of a sequence of frames; and [5, 14] estimate both spatial and temporal ordering of frame patches. Also, examples of video-level temporal SSL include the following: [13] learns a feature embedding that captures changes in the video's motion dynamics such as speed and warping; [31] identifies the arrow of time in videos; and [8, 3, 33] predict the playback speed.

All of these approaches perform the data manipulation for their temporal SSL without taking into account the ordering of action segments in videos. For example, when they shuffle frames in a given video [22] or when they manipulate the video's speed [3, 13], the correct frame ordering or speed is readily available and deterministic, given by the very input video. Instead, our video manipulation shuffles latent actions whose "correct" ordering is only inferred and not necessarily well-aligned with ground truth (since the ground truth is not available in unsupervised learning). Also, our SSL accounts for higher-level temporal structures in the video beyond a sequence of frames.

3. Action Shuffle SSL

Our temporal feature embedding rests on the assumption that actions tend to occur in similar relative locations across videos showing the same activity. We capture this temporal consistency at the level of frames and the level of actions. As shown in Fig. 3, we first learn the frame-level temporal embedding as in [18], in the initial stage of our approach (see Fig 1a), and then iteratively learn our action-level embedding through the Generalized EM algorithm (see Fig 1b). In each iteration of the Generalized EM, our action-level embedding is updated as described below.

Fisher features of video frames of positive and negative action sequences are input to an RNN for predicting whether the input actions are shuffled. Both positive and negative sequences consist of 3 action segments randomly sampled from a video, where every action segment has 5 consecutive frames randomly selected from that action's time interval in the video. The sampled actions are shuffled in negative sequences, and respect their time ordering in positive sequences. Since the frame-level embedding is learned as the output of an MLP, as in [18], our SSL *shares the same* MLP to produce the input to our RNN. We train the RNN on the binary cross entropy loss, and take its hidden layer as our action-level feature embedding.

4. HMM and Its Inference

Videos are represented as sequences of frame features $\boldsymbol{x}_{1:T_m} = [x_1, ..., x_t, ..., x_{T_m}]$, where T_m is the length of mth video, $m = 1, \ldots, M$. For HMM modeling and inference, x_t represents the action-level feature embedding. Following [18, 2, 26], we assume that there are at most N latent actions, $\mathbb{C} = \{c : c = 1, \ldots, N\}$, and that each c may occur only once in a video. Thus, for a given video $\boldsymbol{x}_{1:T}$, our goal is to find an optimal action segmentation, $(\hat{\boldsymbol{c}}_{1:K}, \hat{\boldsymbol{l}}_{1:K})$, where $\hat{\boldsymbol{c}}_{1:K} = [\hat{c}_1, \ldots, \hat{c}_K]$ is the predicted action sequence, $K \leq N$, $\hat{c}_k \in \mathbb{C}$, k is the index of a video

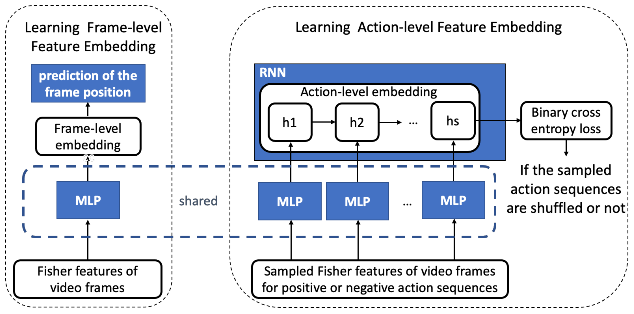
Learning Frame-level Feature Embedding
prediction of the frame position
Frame-level embedding
MLP
Fisher features of video frames
shared
Learning Action-level Feature Embedding
RNN
Action-level embedding
h1
h2
...
hs
Binary cross entropy loss
If the sampled action sequences are shuffled or not
MLP
MLP
...
MLP
Sampled Fisher features of video frames for positive or negative action sequences

Figure 3. (Left) Learning the frame-level embedding of [18]. (Right) Learning our action-level embedding. The sampled positive and negative action sequences are input to an RNN for predicting if they are shuffled or not. The MLP of both frame-level embedding and our action-level embedding is shared.

segment $k = 1, \ldots, K$, and $\hat{\boldsymbol{l}}_{1:K} = [\hat{l}_1, \ldots, \hat{l}_K]$ are their corresponding lengths such that $\sum_{k=1}^{K} \hat{l}_k = T$.

The HMM. We specify the following HMM for identifying a MAP action segmentation:

$$\begin{aligned}
&p(\boldsymbol{c}_{1:K}, \boldsymbol{l}_{1:K}|\boldsymbol{x}_{1:T}) \\
&\quad \propto p(\boldsymbol{x}_{1:T}|\boldsymbol{c}_{1:K}, \boldsymbol{l}_{1:K}) p(\boldsymbol{l}_{1:K}|\boldsymbol{c}_{1:K}) p(\boldsymbol{c}_{1:K}), \\
&\quad = \Big[\prod_{t=1}^{T} p(x_t|c_{k(t)})\Big]\Big[\prod_{k=1}^{K} p(l_k|c_k)\Big]\Big[\prod_{k=1}^{K-1} p(c_{k+1}|c_k)\Big].
\end{aligned} \tag{1}$$

In (1), the likelihood $p(x_t|c)$ is estimated as

$$p(x_t|c) \propto \frac{p(c|x_t)}{p(c)}, \tag{2}$$

where $p(c|x_t)$ is computed by passing x_t to the MLP and taking the resulting softmax score for action c. The prior $p(c)$ is defined as a uniform distribution for all $c \in \mathbb{C}$. The length of each action is modeled as a Poisson distribution:

$$p(l|c) = \frac{\lambda_c^l}{l!} e^{-\lambda_c}, \tag{3}$$

where λ_c is the mean length for class c.

The transition probability $p(c_{k+1}|c_k)$ in (1) is defined with respect to the initialized ordering of the latent actions, $\boldsymbol{c}_{1:N}^{\text{init}} = [1, 2, 3, \ldots, N]$, computed by the K-means in the initial stage of our unsupervised learning, as mentioned in Sec. 1. Specifically, we define:

$$p(c_{k+1}|c_k) \propto \begin{cases} \dfrac{\lambda_{c_k} + \lambda_{c_{k+1}}}{\sum_{j=c_k}^{c_{k+1}} \lambda_{c_j}} & , \quad c_{k+1} > c_k \\ 0 & , \quad \text{otherwise} \end{cases}, \tag{4}$$

where "$\propto$" denotes the appropriate normalization such that $\sum_{c=c_k+1}^{N} p(c|c_k) = 1$. From (4), $p(c_{k+1}|c_k)$ prevents transitions that would result in the opposite action ordering from the initial $\boldsymbol{c}_{1:N}^{\text{init}}$. Also, $p(c_{k+1}|c_k)$ favors transitions between actions with consecutive labels in $\boldsymbol{c}_{1:N}^{\text{init}}$ the most. But we also allow transitions to actions with larger labels, $(c_{k+1} - c_k) > 1$, especially if the expected time interval between c_k and c_{k+1} is short, along the initial action ordering in $\boldsymbol{c}_{1:N}^{\text{init}}$. This allows for skipping some of the latent actions in the HMM inference. Importantly, $p(c_{k+1}|c_k)$ penalizes "long skips" since the denominator in (4) strictly increases with the number of actions skipped $(c_{k+1} - c_k - 1)$.

Note that our $p(c_{k+1}|c_k)$ extends previous work [18], where $p(c_{k+1}|c_k) = 0$ for all $(c_{k+1} - c_k) \neq 1$, i.e., in [18], the predicted action sequence is made equal to the fixed initial transcript, $\hat{\boldsymbol{c}}_{1:N} = \boldsymbol{c}_{1:N}^{\text{init}}$. Another extension of [18] is our length model $p(l|c)$ in (3).

The HMM Inference. The MAP $(\hat{\boldsymbol{c}}_{1:K}, \hat{\boldsymbol{l}}_{1:K})$ is computed by the Viterbi inference. Given $\boldsymbol{x}_{1:T}$ and the initial transcript $\boldsymbol{c}_{1:N}^{\text{init}}$, the Viterbi algorithm recursively maximizes the posterior in (1) such that the first k actions of the transcript $\boldsymbol{c}_{1:k} = [c_1, ..., c_k]$ at time t respect the action ordering in $\boldsymbol{c}_{1:N}^{\text{init}}$, $\boldsymbol{c}_{1:k} \preceq \boldsymbol{c}_{1:N}^{\text{init}}$

$$\begin{aligned}
p(\hat{\boldsymbol{c}}_{1:k}, \hat{\boldsymbol{l}}_{1:k}|\boldsymbol{x}_{1:t}) = \max_{\substack{t',\, t'<t \\ c_k \in \boldsymbol{c}_{k-1:N}^{\text{init}}}} \Bigg\{ & p(\hat{\boldsymbol{c}}_{1:k-1}, \hat{\boldsymbol{l}}_{1:k-1}|\boldsymbol{x}_{1:t'}) \\
& \cdot \Big(\prod_{s=t'}^{t} p(x_s|c_{k(s)})\Big) \cdot p(l_k|c_k) \cdot p(c_k|c_{k-1}) \Bigg\},
\end{aligned} \tag{5}$$

where $l_k = t - t'$ and $p(c_k|c_{k-1})$ penalizes transitions for which $c_k - c_{k-1} > 1$. We set $p(\cdot|\boldsymbol{x}_{1:0}) = 1$, and $p(c_1|c_0) = \kappa$, where $\kappa > 0$ is a constant. The final prediction is given by the final recursion of $p(\hat{\boldsymbol{c}}_{1:K}, \hat{\boldsymbol{l}}_{1:K}|\boldsymbol{x}_{1:T})$.

The Likelihood MLP. Our HMM uses an MLP to estimate frame likelihoods, as specified in (2). Note that this likelihood MLP is different from the MLP used in our SSL and shown in Fig. 3. The likelihood MLP is a framewise classifier, initially trained on annotations produced by the K-means in the initial stage of our approach (see Fig. 1a). In the subsequent training iterations, the likelihood MLP is supervised by the frame labelings, $\{\hat{\boldsymbol{c}}_{1:K}\}_{m=1}^{M}$, predicted in the previous iteration for all videos.

5. Alternating Learning

We jointly learn the HMM, MLP, and action-level feature embedding by alternating the Expectation and Maximization steps of the Generalized EM algorithm. In the E-step, we estimate the following $Q(\theta, \theta^{\text{old}})$ function of the θ parameters, where $\theta = \{W, \Lambda\}$, W are the parameters of the likelihood MLP and RNN for the action-level embedding, and $\Lambda = \{\lambda_c : c \in \mathbb{C}\}$ is the set of mean action lengths:

$$\begin{aligned}
&Q(\theta, \theta^{\text{old}}) = \\
&\tfrac{1}{T_m} \textstyle\sum_{(\boldsymbol{c}^m, \boldsymbol{l}^m)} \sum_{m=1}^{M} p(\boldsymbol{c}^m, \boldsymbol{l}^m|\boldsymbol{x}^m; \theta^{\text{old}}) \log p(\boldsymbol{c}^m, \boldsymbol{l}^m, \boldsymbol{x}^m; \theta),
\end{aligned} \tag{6}$$

where from (1) the joint log-likelihood in (6) is

$$\begin{aligned}
&\log p(\boldsymbol{c}^m, \boldsymbol{l}^m, \boldsymbol{x}^m; \theta) = \\
&\quad \textstyle\sum_{t=1}^{T_m} \log p(x_t^m|c_{k(t)}^m; W) + \sum_{k=1}^{K_m} \log p(l_k^m|c_k^m; \Lambda) \\
&\quad + \textstyle\sum_{k=1}^{K_m} \log p(c_{k+1}^m|c_k^m; \Lambda)
\end{aligned} \tag{7}$$

For our videos, the posteriors are products of framewise likelihoods over typically more than 1000 frames. Hence, the posteriors in (6) are very close to zero for all latent variables $(\boldsymbol{c}^m, \boldsymbol{l}^m)$ that are different from the MAP ones, $(\boldsymbol{c}^m, \boldsymbol{l}^m) \neq (\hat{\boldsymbol{c}}^m, \hat{\boldsymbol{l}}^m)$, specified in (5). Therefore, for our problem setting, it is appropriate to approximate (6) as $Q(\theta, \theta^{\text{old}}) \approx \frac{1}{T_m} \log p(\hat{\boldsymbol{c}}^m, \hat{\boldsymbol{l}}^m, \boldsymbol{x}^m; \theta)$.

In the M-step, we maximize $Q(\theta, \theta^{\text{old}})$ with respect to W and Λ by performing fixed-step gradient descent which updates the parameters of the likelihood MLP and RNN as:

$$W^{\text{new}} = W^{\text{old}} + \alpha \frac{1}{M} \sum_{m=1}^{M} \sum_{t=1}^{T_m} \nabla \log p(x_t^m | \hat{c}_{k(t)}^m; W), \quad (8)$$

where α is the learning rate. Also, maximizing $Q(\theta, \theta^{\text{old}})$ with respecto Λ gives the following update rule for the mean length of every action $\lambda_c \in \Lambda$:

$$\lambda_c^{\text{new}} = \lambda_c^{\text{old}} + \frac{1}{M} \sum_{m=1}^{M} \left(\frac{\sum_{k=1}^{K_m} \hat{l}_k^m \cdot 1(c = \hat{c}_k^m)}{\sum_{k=1}^{K} 1(c = \hat{c}_k^m)} - \lambda_c^{\text{old}} \right) \quad (9)$$

where M is the number of videos.

In [28], the interested reader can find the proof that the weak convergence is guaranteed, $p(\boldsymbol{c}_{1:k}, \boldsymbol{l}_{1:k} | \boldsymbol{x}_{1:t}; \theta^{\text{new}}) > p(\boldsymbol{c}_{1:k}, \boldsymbol{l}_{1:k} | \boldsymbol{x}_{1:t}; \theta^{\text{old}})$, as long as the above updating process maintains that $Q(\theta^{\text{new}}, \theta^{\text{old}}) > Q(\theta^{\text{old}}, \theta^{\text{old}})$. In our experiments, we empirically observe convergence when a difference between the new and old log-posteriors is less than $\epsilon = 10^{-3}$.

6. Learning with All Activities

As in [18], we also address action segmentation across all activities. In this case, we have no access to the activity class, so the assumption that actions tend to occur in similar relative temporal orderings across all videos may not be justified. To address this issue, we take the following steps.

Similar to the previously discussed one-activity setting, we first learn the frame-level embedding for all videos across all activities. The frames are then clustered in the frame-level embedding space to construct a bag-of-words representation for each video with a soft assignment. This allows for clustering videos based on their bag-of-words representation, and each resulting cluster is interpreted as a set of videos showing the same activity. Finally, we perform the second stage of our approach, as illustrated in Fig. 1b, for each video set separately.

7. Experiments

Datasets. For evaluation, we use three benchmark datasets: Breakfast [15], YouTube Instructional [2], and 50Salads [27].

Breakfast is a large-scale dataset, consisting of 10 different complex activities of people making breakfast, with approximately 8 actions per activity class. Every video has on average 6.9 actions, and the video lengths vary from a few seconds to several minutes. For evaluation on Breakfast, as unsupervised input video features, we use the reduced Fisher Vector features [16], as in [26, 18].

YouTube Instructions shows five activities: *making coffee, cpr, jumping car, changing car tire, potting a plant* in 150 videos with an average length of about two minutes. As in [26, 18], for evaluation on YouTube Instructions, we use the unsupervised features proposed by [2].

50Salads has 4.5 hours of video footage of one complex activity, that of making a salad. Its video length is much longer than that of Breakfast and YouTube Instructions. As in [18], two different action-granularity levels are used for evaluation: mid-level with 17 action classes and eval-level with 9 action classes.

Evaluation Metrics. For establishing a correspondence between the predicted segmentation and ground-truth segments, we follow [2, 26, 18] and use the Hungarian algorithm for one-to-one matching based on the overlap between the matched segments over all videos. For evaluation on Breakfast and 50Salads, we compute the mean over frames (MoF). For YouTube Instructions, we report the F1-score, where for calculating precision and recall the positive detections must overlap more than 50% with the matched ground-truth segments.

Implementation Details. In our experiments, the dimension of our action-level feature embedding is set to be 20. The MLP used for the embedding has one 40×20 hidden layer. The hidden-to-hidden layer in the RNN is 20×20. The likelihood MLP for the HMM has one $40 \times N$ hidden layer. One iteration of the Generalized EM algorithm is referred to as epoch. In our experiments, we observe convergence after 20 epochs. It only takes several hours for training, *e.g.* an average of 1 hour training time given an activity from Breakfast. We observe as the number of epochs increases we get non-decreasing Q function given by (6). Within each epoch we generate $2 \cdot M$ positive and negative action sequences, two per video, for our SSL of the RNN. The backprop for the SSL is performed with the SGD with momentum, and our learning rate is 0.001.

Ablations. We consider the following variants of our approach for evaluating the effect of each component:

- ASAL = Our full approach with the alternating learning of the action-level embedding and HMM;
- FTE + HMM = From our full approach we removed the action-level embedding and keep the frame-level temporal embedding (FTE) of [18]; the HMM is iteratively updated, but not FTE as it belongs to the first stage of our approach;
- ActionShuffle + initHMM = From our full approach we remove the alternating learning, but use the action-level embedding and the HMM initialized on the K-means results from the first stage of our approach; this version tests the effect of our joint learning.
- ActionShuffle + Viterbi = From our full approach we remove the HMM, but use the action-level embedding and the Viterbi algorithm constrained to respect the initial fixed transcript of actions as in [18]; this version amounts to running [18] with our action-level feature

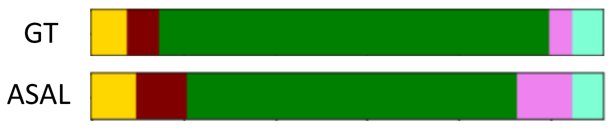

Figure 4. A representative result for our full approach ASAL on a sample video *P04_webcam02_P04_friedegg* from the Breakfast dataset. Top-down, the rows correspond to the ground truth sequence of actions (pour_oil, crack_egg, fry_egg, take_plate, put_egg2plate) and our action segmentation.

embedding instead of their FTE.

7.1. Evaluation for the Same Activity

In this section, evaluation is done for the setting where all videos belong to the same activity class. Table 1, Table 2, and Table 3 show that our approach outperforms the state of the art, on all three datasets, for this setting. On Breakfast, our approach gives a higher F-1 score by 11.5, and a higher MoF by 10.7 than the strong baseline [18]. On Youtube Instructional, our approach outperforms [18] by 3.8 in F1-score and 5.9 in MoF. On 50salads, we get better results than [18] for both eval and mid granularity level features by 3.7 and 4.2 in MoF, respectively. In addition, for a comparison to an upper-bound performance, Table 1 also includes the best results of recent approaches on Breakfast trained under two other learning settings which increase the level of supervision – specifically, fully-supervised and weakly transcript-supervised learning as reviewed in Sec. 2. As can be seen, our approach comes in performance very close to the best weakly supervised approach on Breakfast. In addition, we also show the results of LSTM+AL [1], as a representative of approaches that use a different evaluation method with a per-video local Hungarian matching.

Figure 4 illustrates a representative action segmentation that our approach ASAL produced for a sample video from Breakfast. As can be seen, ASAL is usually good at identifying salient actions, but may miss the start and end frames of the corresponding ground-truth actions.

Effect of Feature Embedding. For the setting where all videos belong to the same activity class, we compare the results of our full approach ASAL and FTE+HMM. FTE+HMM computes the frame-level temporal embedding (FTE) in the first stage, and does not update it in the second stage. Table. 4 shows that, on Breakfast, our full approach ASAL gives better results than FTE+HMM by 2.1 in F1-score and 4.0 in MoF. The large performance gain of our approach suggests that our action-level embedding successfully captures the temporal structure of videos.

Effect of Alternating Training and HMM. For the setting where all videos belong to the same activity class, we evaluate two different ways for learning the components of our approach – specifically, the joint alternating learning of our full approach ASAL, and separate learning of the action-level embedding and HMM in ActionShuffle+initHMM based on the K-means results. Specifically, ActionShuffle+initHMM trains the likelihood MLP with the framewise pseudo-ground truth from the K-means,

	Breakfast	
Fully Supervised		MoF
HTK [15]		28.8
TCFPN [7]		52.0
HTK+DTF w. PCA [16]		56.3
RNN+HMM [7]		60.6
Weakly Supervised		MoF
OCDC [4]		8.9
HTK [17]		25.9
CTC [11]		21.8
ECTC [11]		27.7
HMM+RNN [23]		33.3
TCFPN [7]		18.3
NN-Viterbi [25]		43.0
D3TW [6]		45.7
CDFL [20]		50.2
Unsupervised	F1-score	MoF
Mallow [26]	-	34.6
CTE [18]	26.4	41.8
(LSTM+AL) [1]	-	(42.9*)
VTE-UNET [30]	-	48.1
Our ASAL	**37.9**	**52.5**

Table 1. Comparison of our approach with the state of the art on Breakfast. The table also shows the latest best results on Breakfast for the fully supervised and weakly supervised learning, as an upper-bound performance to ours. The dash means “not reported”, and * means that results are evaluated with the “per video” Hungarian matching, not the Hungarian matching over all videos. In the unsupervised setting, we get the best F1-score and MoF, and come very close to the best weakly supervised performer.

	YouTube Instructions	
Unsupervised		
	F1-score	MoF
Frank-Wolfe [2]	24.4	-
Mallow [26]	27.0	27.8
CTE [18]	28.3	39.0
VTE-UNET [30]	29.9	-
Our ASAL	**32.1**	**44.9**
(LSTM+AL) [1]	(39.7*)	-

Table 2. Comparison of our approach with the state of the art under unsupervised learning on YouTube Instructions. The dash means “not reported”, and * means that results are evaluated with the “per video” Hungarian matching, not the Hungarian matching over all videos. We achieve the best F1-score and MoF.

50Salads		
Unsupervised		
	Granularity level	MoF
VTE-UNET [30]	eval	30.6
CTE [18]	eval	35.5
Our ASAL	eval	**39.2**
(LSTM+AL) [1]	eval	(60.6*)
VTE-UNET [30]	mid	24.2
CTE [18]	mid	30.2
Our ASAL	mid	**34.4**

Table 3. Comparison of our approach with the state of the art under unsupervised learning on 50salads. The dash means "not reported", and * means that results are evaluated with the "per video" Hungarian matching, not the Hungarian matching over all videos. We achieve the best results on both action-granularity levels.

Embeddings		
Breakfast		
	F1-score	MoF
FTE + HMM	35.4	47.7
ASAL	**37.9**	**52.5**

Table 4. Evaluation of different feature embeddings on Breakfast. ASAL is our full approach, and FTE+HMM does not use our action-level embedding but only the frame-level embedding of [18]. We achieve better results, which suggests that our action-level embedding successfully captures the temporal structure.

and learns the expected action length in (3) as an average of action lengths in the K-means. Then, such an HMM is inferred with the Viterbi algorithm, and the resulting action segmentation is used for our SSL of the action-level feature embedding. In ActionShuffle+initHMM, there is no alternating training, i.e., after the initial separate learning all components are not updated further. In addition, we also evaluate ActionShuffle+Viterbi that does not have our HMM but the model used in [18]. This model evaluates frame likelihoods, but does not capture action lengths and action transitions. Also, their model inference is constrained to the fixed initial transcript of actions. Table. 5 shows that our full approach gives superior performance on Breakfast in comparison with ActionShuffle+initHMM and ActionShuffle+Viterbi. ActionShuffle+initHMM gives better results than ActionShuffle+Viterbi, which suggests that accounting for action lengths and transitions in the HMM is very important.

For evaluating convergence of our alternating learning, in Fig. 5, we plot values of the Q function, given by (6), over training epoches on the activity "Juice" from Breakfast. The vertical axis is the mean of Q values over all videos in one epoch. The figure shows that our Generalized EM algorithm converges after 20th epoch on the activity "Juice".

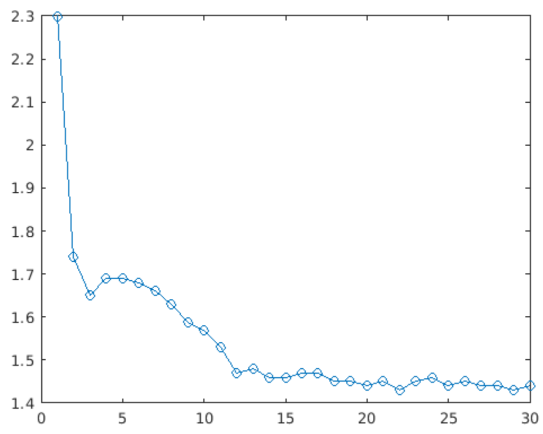

Figure 5. The mean of Q values over all videos for training epochs on videos of the activity "Juice" from Breakfast. The plot shows that our Generalized EM algorithm converges after 20th epoch.

Difference in Learning and Models		
Breakfast	F1-score	MoF
ActionShuffle+Viterbi	30.7	44.1
ActionShuffle+initHMM	32.1	46.8
ASAL	**37.9**	**52.5**

Table 5. Effect of Alternating Training and HMM. We compare our full approach ASAL with ActionShuffle+Viterbi, where no HMM is used, and ActionShuffle+initHMM, where no joint training is used. ASAL gives the best performance.

7.2. Evaluation across All Activities

In this section, we evaluate our approach in a more general unsupervised setting, where videos belong to different activity classes with different temporal structures of actions. Following [18], we also perform the Hungarian matching of all predicted actions to the ground truth action segments. For this setting, all experiments are evaluated on the Breakfast dataset. We assume $N = 5$ actions for each activity, and that there are 10 activities in Breakfast. Then the matching is performed between 50 different action clusters to 48 ground-truth actions, and the remaining unmatched clusters are set as background.

In Table 6, we compare our full approach ASAL with the state of the art on Breakfast, for this multi-activity setting, and observe that ASAL gives the best results.

Effect of Embedding For the multi-activity setting, we compare our ASAL with FTE+HMM which does not use the action-level feature embedding. We also define another variant of our approach, where the action-level embedding is learned for all activity classes without taking into account that videos belong to different activities – the approach referred to as Global emb+HMM. As shown in Table 7, the action-level embedding learned for each estimated activity

Learning across All Activities	
Breakfast	MoF
CTE [18]	16.4
Our ASAL	**20.2**

Table 6. Evaluation in the setting when videos belong to multiple different activity classes. Our full approach ASAL gives better results than [18].

in ASAL gives better performance than Global emb+HMM, since actions of different activities have very different temporal orderings which cannot be reliably captured by our the action-level embedding aimed only for a single activity.

Embedding Across All Activities	
Breakfast	MoF
Global emb + HMM	18.3
FTE + HMM	17.9
ASAL	**20.2**

Table 7. Evaluation of embeddings across all activities. The superior performance of our ASAL suggests that the action-level embedding is not suitable for capturing large variations in the temporal structure of multiple distinct activities.

Effect of Alternating Learning and HMM For the multi-activity setting, we also compare our full approach ASAL with ActionShuffle+initHMM and ActionShuffle+Viterbi. As shown in Table 8, ASAL gives the superior performance.

HMM & Training Across All Activities	
Breakfast	MoF
ActionShuffle+Viterbi	17.2
ActionShuffle+initHMM	18.7
ASAL	**20.2**

Table 8. Evaluation of HMM and two different learning strategies on Breakfast for the setting when videos belong to different activities. We compare our full approach ASAL with ActionShuffle+Viterbi, where no HMM is used, and ActionShuffle+initHMM, where learning of the HMM and action-level embedding is independent and not alternated. ASAL gives the best performance.

8. Conclusion

In this paper, we have advanced unsupervised action segmentation by making the following contributions. First, we have specified a new self-supervised learning (SSL) as a verification of the temporal ordering of actions. The proposed SSL provides the action-level feature embedding used in an HMM for inferring a MAP action segmentation. Second, our HMM explicitly models action lengths and transitions, and in this way relaxes the restrictive assumptions of prior work that all videos have a fixed time ordering of actions. Third, we have unified learning of the action-level embedding and HMM within the Generalized EM framework. Our evaluation studies two different settings – when videos show a single activity or multiple distinct activities – on the Breakfast, Youtube Instructions, and 50Salads datasets. In both unsupervised settings, and on all three datasets, our approach achieves superior results relative to the state of the art, and even comes close to the best weakly supervised performer on Breakfast for the single-activity setting. We have also presented a detailed ablation study that demonstrates advantages of the proposed: a) action-level embedding relative to the frame-level temporal embedding of prior work; b) modeling of action lengths and transitions relative to fixing inference to a predefined action transcript as in prior work; and c) joint alternating training of all components of our approach relative to their separate training.

Acknowledgement. This work was supported in part by DARPA XAI Award N66001-17-2-4029 and DARPA MCS Award N66001-19-2-4035.

References

[1] Sathyanarayanan N Aakur and Sudeep Sarkar. A perceptual prediction framework for self supervised event segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1197–1206, 2019. 6, 7

[2] Jean-Baptiste Alayrac, Piotr Bojanowski, Nishant Agrawal, Josef Sivic, Ivan Laptev, and Simon Lacoste-Julien. Unsupervised learning from narrated instruction videos. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4575–4583, 2016. 2, 3, 5, 6

[3] Sagie Benaim, Ariel Ephrat, Oran Lang, Inbar Mosseri, William T. Freeman, Michael Rubinstein, Michal Irani, and Tali Dekel. Speednet: Learning the speediness in videos. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9919–9928, 2020. 1, 2, 3

[4] Piotr Bojanowski, Rémi Lajugie, Francis Bach, Ivan Laptev, Jean Ponce, Cordelia Schmid, and Josef Sivic. Weakly supervised action labeling in videos under ordering constraints. In *European Conference on Computer Vision*, pages 628–643. Springer, 2014. 6

[5] Uta Büchler, Biagio Brattoli, and Björn Ommer. Improving spatiotemporal self-supervision by deep reinforcement learning. In *ECCV*, pages 797–814, 2018. 1, 3

[6] Chien-Yi Chang, De-An Huang, Yanan Sui, Li Fei-Fei, and Juan Carlos Niebles. D3tw: Discriminative differentiable dynamic time warping for weakly supervised action alignment and segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3546–3555, 2019. 6

[7] Li Ding and Chenliang Xu. Weakly-supervised action segmentation with iterative soft boundary assignment. In *Pro-*

ceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 6508–6516, 2018. 6

[8] Dave Epstein, Boyuan Chen, and Carl Vondrick. Oops! predicting unintentional action in video. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 916–926, 2020. 1, 2, 3

[9] Mohsen Fayyaz and Jurgen Gall. Sct: Set constrained temporal transformer for set supervised action segmentation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020. 3

[10] Basura Fernando, Sareh Shirazi, and Stephen Gould. Unsupervised human action detection by action matching. In *CVPR*, 2017. 3

[11] De-An Huang, Li Fei-Fei, and Juan Carlos Niebles. Connectionist temporal modeling for weakly supervised action labeling. In *European Conference on Computer Vision*, pages 137–153. Springer, 2016. 3, 6

[12] H. Jain and G. Harit. Unsupervised temporal segmentation of human action using community detection. In *2018 25th IEEE International Conference on Image Processing (ICIP)*, pages 1892–1896, 2018. 3

[13] Simon Jenni, Givi Meishvili, and Paolo Favaro. Video representation learning by recognizing temporal transformations. In *European Conference on Computer Vision*, 2020. 1, 3

[14] Dahun Kim, Donghyeon Cho, and In So Kweon. Self-supervised video representation learning with space-time cubic puzzles. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01):8545–8552, 2019. 1, 3

[15] Hilde Kuehne, Ali Arslan, and Thomas Serre. The language of actions: Recovering the syntax and semantics of goal-directed human activities. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 780–787, 2014. 2, 5, 6

[16] Hilde Kuehne, Juergen Gall, and Thomas Serre. An end-to-end generative framework for video segmentation and recognition. In *2016 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1–8. IEEE, 2016. 5, 6

[17] Hilde Kuehne, Alexander Richard, and Juergen Gall. Weakly supervised learning of actions from transcripts. *Computer Vision and Image Understanding*, 163:78–89, 2017. 3, 6

[18] Anna Kukleva, Hilde Kuehne, Fadime Sener, and Jurgen Gall. Unsupervised learning of action classes with continuous temporal embedding. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 12066–12074, 2019. 1, 2, 3, 4, 5, 6, 7, 8

[19] Hsin-Ying Lee, Jia-Bin Huang, Maneesh Singh, and Ming-Hsuan Yang. Unsupervised representation learning by sorting sequences. In *IEEE International Conference on Computer Vision, ICCV*, pages 667–676, 2017. 1, 3

[20] Jun Li, Peng Lei, and Sinisa Todorovic. Weakly supervised energy-based learning for action segmentation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 6243–6251, 2019. 3, 6

[21] Jun Li and Sinisa Todorovic. Set-constrained viterbi for set-supervised action segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10820–10829, 2020. 3

[22] Ishan Misra, C Lawrence Zitnick, and Martial Hebert. Shuffle and learn: unsupervised learning using temporal order verification. In *European Conference on Computer Vision*, pages 527–544. Springer, 2016. 1, 3

[23] Alexander Richard, Hilde Kuehne, and Juergen Gall. Weakly supervised action learning with rnn based fine-to-coarse modeling. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 754–763, 2017. 3, 6

[24] Alexander Richard, Hilde Kuehne, and Juergen Gall. Action sets: Weakly supervised action segmentation without ordering constraints. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5987–5996, 2018. 3

[25] Alexander Richard, Hilde Kuehne, Ahsan Iqbal, and Juergen Gall. Neuralnetwork-viterbi: A framework for weakly supervised video learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7386–7395, 2018. 3, 6

[26] Fadime Sener and Angela Yao. Unsupervised learning and segmentation of complex activities from video. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8368–8376, 2018. 3, 5, 6

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

[28] Martin A Tanner. *Tools for statistical inference: observed data and data augmentation methods*, volume 67. Springer Science & Business Media, 2012. 2, 5

[29] Rosaura G. VidalMata, Walter J. Scheirer, Anna Kukleva, David Cox, and Hilde Kuehne. Joint visual-temporal embedding for unsupervised learning of actions in untrimmed sequences, 2020. 3

[30] Rosaura G VidalMata, Walter J Scheirer, Anna Kukleva, David Cox, and Hilde Kuehne. Joint visual-temporal embedding for unsupervised learning of actions in untrimmed sequences. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 1238–1247, 2021. 6, 7

[31] Donglai Wei, Joseph J Lim, Andrew Zisserman, and William T Freeman. Learning and using the arrow of time. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8052–8060, 2018. 3

[32] Chenxia Wu, Jiemi Zhang, Silvio Savarese, and Ashutosh Saxena. Watch-n-patch: Unsupervised understanding of actions and relations. In *CVPR*, 2015. 2, 3

[33] Yuan Yao, Chang Liu, Dezhao Luo, Yu Zhou, and Qixiang Ye. Video playback rate perception for self-supervised spatio-temporal representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 1, 2, 3