Real-Time Temporal Action Localization in Untrimmed Videos by Sub-Action Discovery

Rui Hou[1]
rhou@crcv.ucf.edu
Rahul Sukthankar[2]
sukthankar@google.com
Mubarak Shah[1]
shah@crcv.ucf.edu

[1] Center for Research in Computer Vision
University of Central Florida
Orlando, USA

[2] Google Research
Mountain View, USA

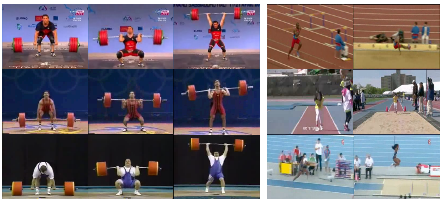
Abstract

This paper presents a computationally efficient approach for temporal action detection in untrimmed videos that outperforms state-of-the-art methods by a large margin. We exploit the temporal structure of actions by modeling an action as a sequence of sub-actions. A novel and fully automatic sub-action discovery algorithm is proposed, where the number of sub-actions for each action as well as their types are automatically determined from the training videos. We find that the discovered sub-actions are semantically meaningful. To localize an action, an objective function combining appearance, duration and temporal structure of sub-actions is optimized as a shortest path problem in a network flow formulation. A significant benefit of the proposed approach is that it enables real-time action localization (40 fps) in untrimmed videos. We demonstrate state-of-the-art results on THUMOS'14 and MEXaction2 datasets.

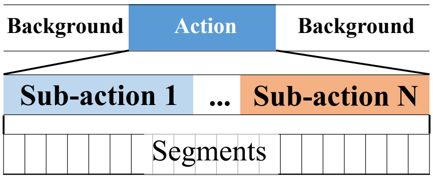
1 Introduction

Video action recognition continues to be a popular topic in computer vision. Although previous attempts have shown promising results, most are designed for clip-level classification in manually trimmed datasets. However, in the real world, the majority of videos are untrimmed, where an action of interest may occur only in a small part of a long video. The challenge is to temporally localize an action of interest, while ignoring other irrelevant actions and the background. In this context, most of the methods designed for trimmed videos will fail. In this paper, we address the problem of real-time temporal action localization, which predicts the beginning and ending frames of an action in an untrimmed video, in a computationally efficient manner.

In the proposed approach (Fig. 1), an action is modeled as a sequence of automatically discovered, semantically meaningful sub-actions, whose number and duration can vary from action to action. For instance, the "clean and jerk" action (left) is decomposed into three sub-actions, corresponding to "lift", "clean" and "jerk", while the "long jump" (right) has two



(a)



(b)

Figure 1: (a) Automatically discovered sub-actions in diverse instances of two actions. The number of sub-actions is automatically determined and they are found to be semantically meaningful. Each row shows one example of "clean and jerk" and "long jump" actions from diverse videos. (b) A typical untrimmed video consists of many background segments with one or more actions. We group short segments from untrimmed video into sub-actions whose temporal structure is exploited for temporal action localization.

sub-actions, that correspond to "approaching run" and "jump". Note that our discovered sub-actions are consistent across different instances that vary significantly in terms of viewpoint and visual appearance.

Decomposing actions into characteristic sub-actions is a challenging task, due to significant intra-class variability, such as variations in viewpoints, human poses and the execution speed of the action. Therefore, most former approaches either rely on manual annotation of sub-actions or fix the number of sub-actions per action. Manually defining and annotating sub-actions is a subjective task and the quality of annotations depends on the judgment and preference of the annotators. Also, manual annotation of sub-actions requires heavy labor. In addition, actions contain different complexity levels; typically a few sub-actions are enough to represent simple actions, while complex actions would need to be represented with more sub-actions. Obviously, limiting the number of sub-actions to a fixed number for all actions would limit their representation power. In our approach, the number of sub-actions as well as the sub-actions themselves are learned from the training data automatically. Moreover, from the experiments, we observe that the automatically discovered sub-actions are semantically meaningful and they are consistent across different instances of an action.

Given the rate at which videos are generated, applications such as surveillance and video retrieval demand computationally efficient approaches for action detection in untrimmed video. Our proposed approach is designed with this in mind and can process 40 frames per second (on commodity hardware) including all steps, i.e., feature extraction, sub-action detection and action detection.

This paper makes the following contributions. 1) We propose a novel, fully automated algorithm that discovers a discriminative sequence of sub-actions directly from data. The discovered sub-actions are semantically meaningful. 2) We exploit the temporal structure of actions by modeling an action as a sequence of sub-actions, which takes sub-action durations and time between sub-actions into consideration. 3) The proposed approach is computationally efficient and processes video in real-time. 4) We evaluate the proposed sub-action based sequential model on large-scale temporal action localization task and show that the proposed method achieves state-of-the-art results on THUMOS'14 and MEXaction2 datasets.

2 Related Work

The goal of spatio-temporal action localization is to localize actions in space and time simultaneously. Some researchers use segmentation based approaches to solve this problem [5, 17]. Another class of popular approaches is based on detection and tracking [23], which use object detectors [6] or human detectors [2, 11, 26] to obtain candidate action locations.

Different from the above approaches, we aim to solve temporal action localization in untrimmed and unconstrained videos. Almost all approaches for spatio-temporal action localization employ trimmed videos. However, untrimmed videos for temporal action localization are much longer and unconstrained, posing significant challenges. Therefore, solving this problem requires us not only to discriminate between actions but also to distinguish actions from the background (which is taken from the same scene).

When only video-level labels are available, weakly supervised learning based approach can be used. Lai *et al.* [9] use multiple instance learning to select the key concepts in untrimmed videos and localize actions temporally. Sun *et al.* [18] use web images as a prior to improve detection performance. However, as large scale action datasets with temporal annotation, such as THUMOS'14 [7], are introduced, researchers have explored approaches based on learning directly from temporally annotated datasets. Wang *et al.* [22] combine the improved dense trajectories (iDTF) [20, 27] as motion features and frame-level CNN as appearance features together. Karaman *et al.* [8] utilize Fisher encoded iDTF with saliency based pooling. Oneata *et al.* [13, 14] fuse motion, visual and audio features to train classifiers and use the classification scores as a contextual feature to help localization. Shou *et al.* [16] propose a loss function considering temporal overlap and learn segment-based 3D ConvNets for localization. Yeung *et al.* [24] train a recurrent neural network to predict the temporal bounds of actions. Heilbron *et al.* [3] propose temporal proposal to locate the actions temporally. However, none of the above mentioned approaches model the temporal structure of an action by a sequence of sub-actions. Gaidon *et al.* [1] propose to model action as a sequence of atomic action units (actoms). However, their approach relies on manually annotated actoms. In contrast, the sub-actions in our approach are automatically discovered. According to [4, 25], data-driven concepts or sub-actions or actoms mainly have two advantages over manually defined concepts. First, a large number of data-driven concepts can be determined as far as they are different in the corresponding feature space, since they do not necessarily have to be semantically different. Second, manually defined concepts need extra knowledge from sophisticated human raters and they must be carefully designed to maximize their usage.

Several works have aimed to automatically discover mid-level representations. Tang *et al.* [19] propose to treat states of temporal segments as latent variables and model durations of states as well as transitions between states using variable-duration HMM. Lan *et al.* [10] automatically discover mid-level action elements by clustering spatio-temporal segments and represent videos by a hierarchy of mid-level action elements. However, our approach differs from them in several key respects. First, the sub-actions discovered by our approach are consistent across different instances of an action and they are semantically meaningful. By contrast, videos of an action may consist of different sets of sub-actions and the sub-actions may not have clear semantic meanings in [10, 19]. Second, the number of sub-actions in our method is automatically discovered, while [19] manually defines the number of states. Third, the distances between sub-actions are taken into consideration in our approach. Thus our model allows temporal overlap or gap between sub-actions, which is more flexible and robust in realistic videos. In contrast, no overlap or gap between states is allowed in [19]. Last, we address temporal action localization problem while [10] attacks the easier whole-clip action recognition problem.

3 Proposed Approach

Temporal action localization can be formulated as a classification problem where the goal is to assign action labels to each frame, and determine the beginning and ending frames of an action in untrimmed videos. Traditionally, an action has been represented by a single model, which is simple and efficient, nevertheless it is not robust to intra-class variation (in particular in context of untrimmed videos), leading to unsatisfactory performance.

An important fact about actions is that they are usually composed of multiple semantic sub-actions (Fig. 1(b)). While the sub-actions may vary in appearance and duration (e.g., the length of the "approach" run in the "long jump" action), a given action nearly always consists of the same set of sub-actions in a consistent order. Thus, we choose to model an action as a series of sequential sub-actions and train a separate classifier for each sub-action.

An important issue, in context of modeling an action using sub-actions, is how to determine the number of sub-actions for each action. One obvious solution is to manually identify a set of sub-actions for each action and generate training sets by annotating each sub-action in every video; that would be a daunting task. Instead, we propose an automatic method to discover sub-actions for each action. Our approach for discovering sub-actions consists of three main steps. First, temporal segments of all training videos of an action are clustered into different parts. Second, similar parts are merged to obtain candidate sub-actions. Finally, boundaries between candidate sub-actions are adjusted to obtain final sub-actions. Sub-actions discovered in this way are consistent and semantically meaningful (Fig. 1(a)).

Assume a video v is composed of temporal segments of fixed length. Each temporal segment $i \in v$, is represented by a Fisher vector $x_i \in \mathbb{R}^d$ computed over the features, which is used as an input to our model. Let n denotes the number of actions and m_l the number of sub-actions for action l $(l \in \{1,\dots,n\})$. And let a n dimensional one-hot vector $g_i \in \mathbb{R}^n$ denotes the action labels for segment i. If segment i does not belong to action l then $g_i(l)=$ 0. Otherwise, segment i contains sub-action $g_i(l)$ of action l.

4 Discovering Sub-actions

Our key assumption is that all the video clips of an action l share the same sequence of m_l sub-actions. The goal is to design an approach that can automatically find the appropriate number of sub-actions for each action in an unsupervised manner. Sub-actions should correspond to different semantic parts and be consistent in videos clips of the same action. Moreover, the sub-actions in an action should occur in a specific order.

4.1 Clustering Segments Into Parts

Since the number of sub-actions in an action is unknown, we first cluster segments in each video of an action l into k_l parts (we use k_l equal to half of the number of segments in the shortest training video of action l) to serve as candidate sub-actions. These candidate sub-

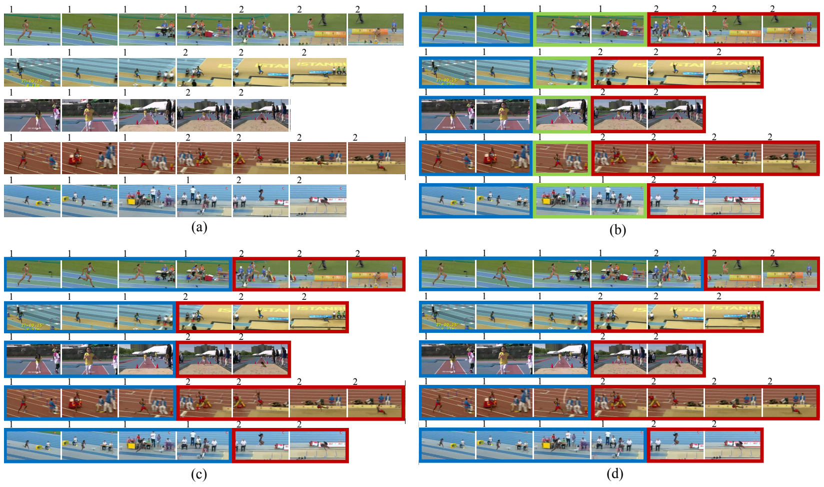


Figure 2: An example of segmenting "high jump" action into several sub-actions. **(a)** Rows represent different length videos of the same action. Temporal segments within a video are represented by key frames. The number on the top of a frame represents the ground truth index of sub-action in the action. In this action there are two sub-actions. **(b)** In the first step, all segments of each video of an action are clustered into k_l (in this case 3) sequential parts, which are shown by borders of different colors (blue, green and red). However, as can be seen that the first sub-action is broken into two parts. **(c)** In the second step, we use hierarchical agglomerative clustering to merge similar parts. Then the first two parts in **(b)** are merged. However, in the first clip, one segment is incorrectly merged with the first part. **(d)** Shows the partitioning results after adjustment. The partitions are updated iteratively.

actions are updated and adjusted later to select the most discriminative sub-actions. Below, we describe the procedure of clustering segments into parts.

For each video, we want to find k_l tight clusters of segments such that the distance of each segment in a cluster from the cluster center is minimized. Let $c_p = \frac{1}{|p|}\sum_{i=0}^{|p|-1} x_{b_p+i}$ be the center of cluster p, where x_{b_p+i} is a feature vector of segment b_p+i, $b_p+i (i \in \{0,1,2,...,|p|-1\})$ represents all the segments in cluster p and b_p is the first segment which belongs to cluster p. Let $\gamma_{b_p+i,p}$ be a binary variable, which is 1 if segment b_p+i is assigned to cluster p, otherwise it is 0. Since we cluster *temporal* segments, which have sequential order, we cannot arbitrarily cluster segments using a standard approach like K-means. Therefore, we define an objective function that imposes a sequential order on the cluster as follows:

$$L = \sum_p \sum_{i=0}^{|p|-1} \gamma_{b_p+i,p} ||x_{b_p+i} - c_p||^2, \quad \text{where } b_p + i \in p \text{ and } b_p + |p| = b_{p+1}. \tag{1}$$

The objective function represents the sum of the squares of the Euclidean distances of each segment to the center of its assigned cluster. Our goal is to find values for $\gamma_{b_p+i,p}$ so as to minimize L. We also need to determine cluster centers c_p. This problem is solved using an expectation maximization (EM) like algorithm. In the expectation step, we keep c_p fixed, and only update $\gamma_{b_p+i,p}$ for segments which are at the part boundaries. In the maximization step, we minimize L with respect to c_p, keeping $\gamma_{b_p+i,p}$ fixed. This two-stage optimization is then repeated until convergence. Note that exploiting the sequential ordering constraint enables the proposed method to be more computationally efficient than naive EM clustering.

4.2 Updating Sub-actions

After clustering, segments in an video are divided into sequential parts, which are candidate sub-actions. However, these candidates may not be sufficiently good to be directly used in our model. There are two main issues. First, sub-actions may be split into multiple parts (Fig. 2a), requiring them to be merged. Second, the partitioning results may be inconsistent over different videos of an action (Fig. 2b). To remove inconsistencies, sub-action labels for some of the segments should be adjusted.

Both of the above issues can be effectively solved by a set of linear SVM classifiers. An SVM classifier is trained between each two adjacent candidate sub-actions, where segments belonging to one candidate sub-action are taken as positive samples and segments belonging to the adjacent candidate sub-action are treated as negative samples. The SVMs try to distinguish adjacent candidate sub-actions. The discriminant function of an SVM classifier is $y = \text{sign}(w \cdot x + b)$, where w, b are learned parameters, y is predicted sub-action label and x is the feature vector of a segment.

Merging similar candidate sub-actions together. We use hierarchical agglomerative clustering to merge similar candidate sub-actions. The distance metric used in the clustering is the SVM margin obtained from the above learned SVM. The SVM margin is determined by its primal form $\frac{1}{\|w\|}$. This process is repeated iteratively by merging candidate sub-actions with the smallest distance until no distance is lower than the threshold (0.9). After merging, action l has m_l sub-actions.

Optimizing sub-actions. In order to ensure that sub-actions are consistent among all videos of an action, we use an iterative procedure to adjust sub-action partitions by alternating between the following two steps. 1) Fix sub-action labels y, train an SVM (w, b) for every pair of adjacent sub-actions. 2) Given the learned SVMs (w, b), update the sub-action labels y of the two segments at the boundary of each adjacent sub-action pairs.

The procedure converges when no sub-action labels are updated, which usually takes 3–4 iterations. In each iteration, only segments at the boundary of adjacent sub-actions pair are tested with the current SVMs and their sub-action labels may be updated. The sub-action labels for all the other segments are kept the same, making the procedure efficient.

5 Sub-Action Detectors

After obtaining the final sub-action partitions for all training videos, a set of sub-action classifiers $T_z(\cdot)(1 \le z \le m_l)$ is trained separately for each action l. In order to recognize and temporally localize actions in the untrimmed testing videos, we first detect sub-action candidates and then combine these sub-action detections to localize actions.

For a sub-action z of action l, we collect all the segments $\{i | g_i(l) = z\}$ as positive samples to train a SVM model. We perform Platt Scaling for the decision values to obtain the probability that the given segment is present in the sub-action z. The model $T_z(\cdot)$ gives us probability of the sub-action in a segment. However, localizing actions only based on this prediction value can be suboptimal. Some false negative segments may break an action instance into multiple instances, leading to inaccurate localization. To reduce the number of false alarms, we take the duration of actions into consideration as well. In most cases, the majority of instances of the same action have similar durations. For instance, the "jump" action usually finishes within 1 second, while "clean and jerk" always takes several seconds. We assume the duration d of a sub-action follows a Gaussian distribution $\Phi(d) \sim \mathcal{N}(\mu, \sigma)$.

Combining the prediction and duration scores together, sub-action z's confidence value with duration d can be expressed as in Eq. 2. The confidence score is computed as the product of classifier prediction and duration score. The prediction score of a duration is the average of SVM probability output from all the segments within that duration.

$$P_z(d) = \frac{1}{|d|} \sum_{i \in d} T_z(x_i) \cdot \Phi_z(d). \tag{2}$$

Assume, we have s segments in a video for which we already have computed sub-action scores. Our aim is to determine starting and ending segment of a sub-action in this video. We compute an $s \times s$ upper triangular detection scores matrix. The column and row indices of the matrix represent the candidate starting and ending segment, respectively. For each starting and ending segment pair that represents a candidate duration, the sub-action score is computed using Eq. 2. Then, using dynamic programming we optimally determine beginning and ending segments for the sub-action.

6 Detecting Actions

Our approach represents each action by a sequence of sub-actions. We enforce that the sub-actions of an action must occur in a sequential order, but two adjacent sub-actions may have some temporal overlap or gap between them. We assume the time between adjacent sub-actions follows a Gaussian probability density function $\Psi(\cdot) \sim \mathcal{N}(\mu, \sigma)$. This function penalizes sub-action combinations with a wrong order or those that have a greater gap than desired. The parameters μ and σ of $\Psi(\cdot)$ are learned from the validation set.

Given an unclipped video of unknown action, we first segment it into equal length segments. We then apply all sub-action detectors, and corresponding to each sub-action detector we get a sequence of detection scores. Now the aim becomes selecting optimal combinations of sub-action detections to detect an action, by considering both sub-action scores and distance between sub-actions. This inference is formulated as a network flow problem. We build a flow graph as shown in Fig. 3. The start and end time of each sub-action detection d are represented as two vertical bars, which are connected by an observation edge with cost $-P(d)$ (Eq. 2). Let d and d' be two adjacent sub-action detections. They are connected by an transition edge with cost $-\Psi(d, d')$. The objective function is

$$\min_f \Big(-\sum_d P(d) f_d - \sum_{d,d'} \Psi(d,d') f_{d,d'}\Big), \quad s.t. f_d, f_{d,d'} \in \{0,1\} \text{ and } \sum_{d'} f_{d,d'} = f_d = \sum_{d'} f_{d',d}, \tag{3}$$

where $f_{dd'}$ represents the flow from detection d to d'. The first constraint enforces that flow is either 1 or 0. A sub-action detection d is selected if $f_d = 1$ and its adjacent sub-action detection d' is also selected if $f_{d,d'} = 1$. Each sub-action detection can be selected to obtain an action detection only once. The second constraint ensures that the incoming flow to a node is equal to its outgoing flow. The shortest path problem is solved efficiently using dynamic programming, similar as in [15] which finds high-quality approximate solutions to min-cost flow problem for multiple objects tracking. A selected shortest path represents a valid action detection, whose sub-action scores and distance scores between sub-actions are optimal. We

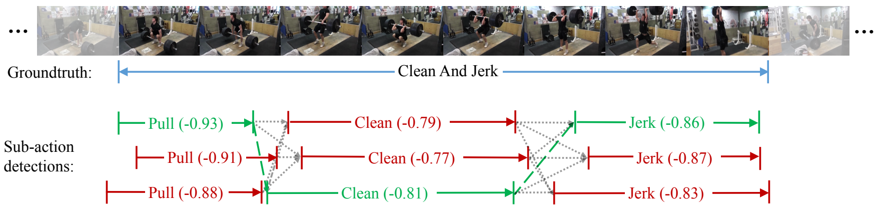


Figure 3: An example of network flow with three sub-actions. The groundtruth is shown by a blue line. The observation edges are denoted as solid lines, and the transition edges are expressed as dashed lines. The green lines represent the selected optimal path. Since the optimal path considers both sub-action scores and distance scores between sub-actions, the locally optimal detection of sub-action 'jerk' with lowest cost is correctly ignored.

keep solving the shortest path problem until the cost of the obtained path is higher than a threshold, which is learned from the validation set. Every time a shortest path is selected, its corresponding sub-action detections are removed from the graph.

7 Experimental Results

We evaluate our method on THUMOS'14 and Mexaction2 datasets. We use the Improved Dense Trajectory Features (iDTF) [20] as low-level features, which consist of four descriptors: histogram-of-gradients (HOG), histogram-of-flow (HOF), motion-based histograms (MBH) and trajectories. In the experiments, we use the first 3 descriptors with late fusion. We generate a 256-bin GMM and build a Fisher vector representation for both datasets.

THUMOS'14. The temporal action detection task of THUMOS'14 [7] consists of 20 classes of sports actions. We use both train and validation sets for training sub-action models, and fix the length of segments to 0.3s. The thresholds and parameters for Gaussian distributions are learned by a ten-fold cross-validation process. We follow the detection measurement protocol specified in the THUMOS'14 temporal action localization challenge.

We present direct comparisons against top performers on the THUMOS'14 challenge leader board [14, 21], which use the same low-level features. Moreover, we also compare our results with two recent deep learning based approaches [16, 24].

For the set of THUMOS'14 classes, the number of sub-actions discovered by our approach is as follows: two actions (Clean & Jerk and High Jump) consist of 3 sub-actions, 15 actions are divided into 2 sub-actions, and 3 actions only contain one sub-action (we call these singleton actions). The mAP (Mean Average Precision) is reported with different intersection over union (IOU) thresholds, α. The results are shown in Table 1. As is clear from these results, our approach significantly outperforms the other approaches.

We also conduct two ablative experiments. In the first experiment, we skip the optimization of part partitioning step (Section 4.2), and assign sub-actions labels for each segments based on the merging results (Section 4.1). The mAPs of this model, denoted as Ours (w/o opt), are consistently worse than those from our complete system, verifying the effectiveness of our part partition optimization step.

In the second experiment, we assume that all actions are composed of only a sub-action (singleton actions). We see that the ablative singleton model, denoted as Ours (Singleton),

	$\alpha = 0.5$	$\alpha = 0.3$	$\alpha = 0.1$
Wang et al. [22]	8.3	14.0	18.2
Oneata et al. [14]	14.4	27.0	36.6
Yeung et al. [24]	17.1	36.0	48.9
Shou et al. [16]	18.8	-	-
Yuan et al. [28]	18.8	30.8	40.9
Ours	**22.0**	**43.7**	**51.3**
Ours (w/o opt.)	20.4	36.1	49.5
Ours (Singleton)	19.3	33.3	48.2

Table 1: Temporal action localization results on THUMOS'14. mAP is reported for different intersection-over-union thresholds α.

performs worse than our proposed approach. This validates our hypothesis that sub-action discovery improves action localization.

Mexaction2. This is a two actions dataset: "BullChargeCape" and "HorseRiding". This data set has three parts. The first part is INA videos, which were collected from TV shows. The videos in this part are untrimmed and split into train, validation and test sets. The second part is from Youtube videos, and the third part is from UCF101 Horse Riding clips. Moreover, the clips in the last two parts are trimmed and are only used for training. We use the train set for training sub-action detectors. The length of segments is fixed as 0.1s. The validation set is used for learning the parameters for Gaussian models and thresholds for action detectors.

We present the comparison against baseline, which uses Bag of Visual Words DTF features, and Shou *et al.* [16]'s reported results. We use the same metric as in [16], which report the mAP with IOU threshold $\alpha = 0.4$. Since both actions are hard to be divided into sub-actions, we get one sub-action for them (they are singleton actions). Our sub-action based approach achieves huge improvement for "BullCharge-Cape" and slightly better result for "HorseRiding" (Table 2). In this experiment, we did not use the distance term.

	Baseline [12]	Shou *et al.* [16]	Ours
BullCharge	0.3	11.6	**26.2**
HorseRiding	3.1	3.1	**3.8**
mAP	1.7	7.4	**15.0**

Table 2: Average precision on MEXaction2 dataset

7.1 Analysis of Our Results

Table 1 shows results from the ablation experiments, i.e. Ours (w/o opt) and Ours (Singleton), in order to verify the contributions of different components of our method. We observe:

1. When α changes from 0.5 to 0.3 in Table 1, our approach outperforms other methods at all the overlap thresholds. The mAP increases greatly at high IOU and reaches 43.7% at $\alpha = 0.3$, demonstrating that we always have greater IOU with ground truth.
2. Our sub-action based model consistently outperforms the singleton model for temporal action detection (Tables 1, 2). However, when IOU threshold decreases to 0.1, singleton model shows less difference (48.2% vs. 51.3%), while when $\alpha = 0.3$ the difference is 10% (Table 1), showing the benefits of modeling sub-actions.

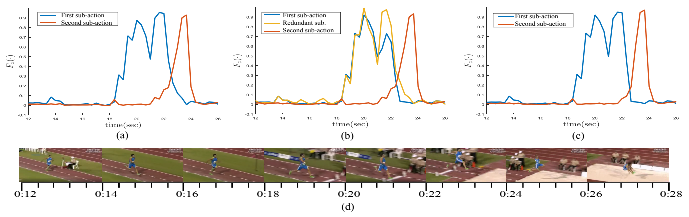


Figure 4: An example showing the sub-action classifier scores for different segments using different steps of our method. **(a)** Results after clustering segments into parts (Section 4.1). **(b)** Results obtained after merging similar parts. **(c)** Results of our full approach using all steps. **(d)** Example frames at different time stamps. Note that the two highly overlapped parts in (a) are correctly merged in (b) The more subtle but equally important change occurs from (b) to (c) where the sub-action partitions are changed so as to accentuate the difference between the blue and red peaks to reduce the area under the intersection interval at 0:22 where partition inconsistency could occur in (b).

3. Without the merging of similar parts, the first sub-action is divided into two redundant parts. As shown in Fig. 4a, the two-part model generates similar scores for the segments. Training separate models for these redundant sub-actions would be wasteful.
4. Comparing plots in Fig. 4b with plots in Fig. 4c, we appreciate the importance of part optimization (Section 4.2), which adjusts boundaries to accentuate the difference between discovered sub-actions.
5. Our approach outperforms other methods by a large margin when an action can be clearly divided into sub-actions (e.g., cliff diving, javelin throw and long jump). When the sub-actions are hard to determine, our algorithm is still competitive (e.g., billiards).
6. Finally, our approach performs less well on short duration actions (golf swing) or actions that resist decomposition into sub-actions (frisbee catch). When the duration of an action varies widely (horse riding), the duration term may hurt performance.

Runtime Analysis. The proposed approach is very efficient, particularly for long videos. Compared to sliding window based methods, for each segment we only compute the low-level features once. Compared to CNN-based methods, we train much faster. For the MEXaction2 experiments, during sub-action discovery, clustering segments takes 121s and updating sub-actions takes 87s. The total time for sub-action training is 1686s. The improved DTF feature extraction and Fisher Vector computation in our implementation is optimized by avoiding the unnecessary string-float conversions. Using dynamic programming and our optimized feature extraction and Fisher Vector computation, our temporal action detection system can process videos at 40 fps.

8 Conclusion

We present a real-time system for temporal action detection in untrimmed videos that learns discriminative and semantically meaningful sub-actions. Both the number of sub-actions for each action and the sub-actions themselves are discovered automatically. We demonstrate state-of-the-art localization performance on standard action datasets. Since the proposed method is agnostic to the type of low-level features, a natural extension would be to integrate the sub-action detection framework with deep learning.

References

[1] Adrien Gaidon, Zaid Harchaoui, and Cordelia Schmid. Actom sequence models for efficient action detection. In *CVPR*, 2011.

[2] Georgia Gkioxari and Jitendra Malik. Finding action tubes. In *CVPR*, 2015.

[3] Fabian Caba Heilbron, Juan Carlos Niebles, and Bernard Ghanem. Fast temporal activity proposals for efficient detection of human actions in untrimmed videos. In *CVPR*, 2016.

[4] Rui Hou, Amir Roshan Zamir, Rahul Sukthankar, and Mubarak Shah. DaMN–discriminative and mutually nearest: Exploiting pairwise category proximity for video action recognition. In *ECCV*, 2014.

[5] Mihir Jain, Jan van Gemert, Hervé Jégou, Patrick Bouthemy, and Cees Snoek. Action localization with tubelets from motion. In *CVPR*, 2014.

[6] Mihir Jain, Jan C. van Gemert, Thomas Mensink, and Cees Snoek. Objects2action: Classifying and localizing actions without any video example. In *ICCV*, 2015.

[7] Y.-G. Jiang, J. Liu, A. Roshan Zamir, G. Toderici, I. Laptev, M. Shah, and R. Sukthankar. THUMOS challenge: Action recognition with a large number of classes. http://crcv.ucf.edu/THUMOS14/, 2014.

[8] Svebor Karaman, Lorenzo Seidenari, and Alberto Del Bimbo. Fast saliency based pooling of Fisher encoded dense trajectories. In *THUMOS'14 Action Recognition Challenge*, 2014.

[9] Kuan-Ting Lai, Felix X Yu, Ming-Syan Chen, and Shih-Fu Chang. Video event detection by inferring temporal instance labels. In *CVPR*, 2014.

[10] Tian Lan, Yuke Zhu, Amir Roshan Zamir, and Silvio Savarese. Action recognition by hierarchical mid-level action elements. In *ICCV*, 2015.

[11] Pascal Mettes, Jan C van Gemert, and Cees GM Snoek. Spot on: Action localization from pointly-supervised proposals. In *European Conference on Computer Vision*, pages 437–453. Springer, 2016.

[12] MEXaction2. http://mexculture.cnam.fr/xwiki/bin/view/Datasets/Mex+action+dataset, 2015.

[13] Dan Oneata, Jakob Verbeek, and Cordelia Schmid. Efficient action localization with approximately normalized fisher vectors. In *CVPR*, 2014.

[14] Dan Oneata, Jakob Verbeek, and Cordelia Schmid. The LEAR submission at thumos 2014, 2014.

[15] Hamed Pirsiavash, Deva Ramanan, and Charless C Fowlkes. Globally-optimal greedy algorithms for tracking a variable number of objects. In *CVPR*, 2011.

[16] Zheng Shou, Dongang Wang, and Shih-Fu Chang. Temporal action localization in untrimmed videos via multi-stage cnns. In *CVPR*, 2016.

[17] Khurram Soomro, Haroon Idrees, and Mubarak Shah. Action localization in videos through context walk. In *ICCV*, 2015.

[18] Chen Sun, Sanketh Shetty, Rahul Sukthankar, and Ram Nevatia. Temporal localization of fine-grained actions in videos by domain transfer from web images. In *ACM Multimedia*, 2015.

[19] Kevin Tang, Li Fei-Fei, and Daphne Koller. Learning latent temporal structure for complex event detection. In *CVPR*, 2012.

[20] Heng Wang and Cordelia Schmid. Action recognition with improved trajectories. In *CVPR*, 2013.

[21] Heng Wang and Cordelia Schmid. LEAR-INRIA submission for the THUMOS workshop, 2013.

[22] Limin Wang, Yu Qiao, and Xiaoou Tang. Action recognition and detection by combining motion and appearance features. In *THUMOS'14 Action Recognition Challenge*, 2014.

[23] Philippe Weinzaepfel, Zaid Harchaoui, and Cordelia Schmid. Learning to track for spatio-temporal action localization. In *CVPR*, 2015.

[24] Serena Yeung, Olga Russakovsky, Greg Mori, and Li Fei-Fei. End-to-end learning of action detection from frame glimpses in videos. In *CVPR*, 2016.

[25] Felix X Yu, Rongrong Ji, Ming-Hen Tsai, Guangnan Ye, and Shih-Fu Chang. Weak attributes for large-scale image retrieval. In *CVPR*, 2012.

[26] Gang Yu and Junsong Yuan. Fast action proposals for human action detection and search. In *CVPR*, 2015.

[27] Jun Yuan, Yong Pei, Bingbing Ni, Pierre Moulin, and Ashraf Kassim. ADSC submission at THUMOS challenge 2015, 2015.

[28] Jun Yuan, Bingbing Ni, Xiaokang Yang, and Ashraf A. Kassim. Temporal action localization with pyramid of score distribution features. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.