Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence (AAAI-17)

# Soft Video Parsing by Label Distribution Learning

**Xin Geng,* Miaogen Ling**
MOE Key Laboratory of Computer Network and Information Integration,
School of Computer Science and Engineering,
Southeast University, Nanjing 210096, China
{xgeng,mgling}@seu.edu.cn

## Abstract

In this paper, we tackle the problem of segmenting out a sequence of actions from videos. The videos contain background and actions which are usually composed of ordered sub-actions. We refer the sub-actions and the background as semantic units. Considering the possible overlap between two adjacent semantic units, we utilize label distributions to annotate the various segments in the video. The label distribution covers a certain number of semantic unit labels, representing the degree to which each label describes the video segment. The mapping from a video segment to its label distribution is then learned by a *Label Distribution Learning* (LDL) algorithm. Based on the LDL model, a soft video parsing method with segmental regular grammars is proposed to construct a tree structure for the video. Each leaf of the tree stands for a video clip of background or sub-action. The proposed method shows promising results on the THUMOS'14 and MSR-II datasets and its computational complexity is much less than the state-of-the-art method.

## Introduction

Action detection or localization of real-world videos has been an active research topic due to its extensive application in video surveillance, human computer interaction, video retrieval, *etc*. Even under the circumstance that the action category of a video is known, it is still rather difficult to extract the actions since they could differ greatly in position and duration. In real-world videos, the actions are often composed of several continuous sub-actions (Pirsiavash and Ramanan 2014). Analogous to the grammar structure of a sentence, the sub-action, action and background (non-action segment) could all be regarded as grammar components of a video. In such case, it is reasonable to facilitate the grammar structure of the action for better action localization. This process is called video parsing (Pirsiavash and Ramanan 2014). For convenience of presentation, both the background and the sub-actions are uniformly called semantic units in the rest of this paper.

The video parsing algorithms can be roughly grouped into two families, i.e., the sliding window based method and the inference based method. Most of the sliding window based methods train a classification model to seperate the action and the background. For example, an SVM model (Oneata, Verbeek, and Schmid 2014; Wang et al. 2015) is trained to score all the video segments in the sliding widows of multiple length in the test videos and Non-Maximum Suppression (NMS) (Neubeck and Van Gool 2006; Hoai, Lan, and De la Torre 2011) is adopted to obtain the optimum global parsing of the videos. However, most methods of this kind ignore the graduality of the actions and restrict the detected action length within only a few choices of the window length. Moreover, the inference based methods often utilize the inner structure of actions to construct the state transition model for video parsing. A semi-Markov model (SMM) (Shi et al. 2011) is proposed to model the state transition between actions and the viterbi-like dynamic programming algorithm is adopted to solve the inference problem. But no sub-actions are considered. An HMM method with latent variables (Tang, Li, and Koller 2012) is proposed to construct a state transition model among the sub-actions and the maximum posteriori inference is adopted to obtain the best parsing of the test video. However, it is not clear how to use it to detect multiple actions in one video. Segmental Regular Grammars (Pirsiavash and Ramanan 2014) proposes a tree structure with segmental regular grammars to present a video. Each leaf in the tree represents a clip of sub-action or background, but the procedure of finding the worst-offending parsing in the training process of the structured SVM is quite time-consuming. Moreover, it ignores the ambiguity among the semantic units and uses a single label to describe a video segment.

It is worth noting that the semantic units are often ordered and interlaced with each other in an action. For example, Fig. 1 shows 10 key frames in a video about *high jump*. There are generally two sub-actions in the high jump action, i.e., *approaching* and *taking off*. The graduality among the three semantic units (background, approaching and taking off) often makes it hard to locate the exact boundaries among them. See, for example in Fig. 1, the key frames (b) and (c) between background and approaching, (e) and (f) between approaching and taking off, or (h) and (i) between taking off and background.

Different from previous work assuming *hard* boundaries among the semantic units, this paper proposes a video parsing method which assumes flexible *soft* boundaries among

*Corresponding author.

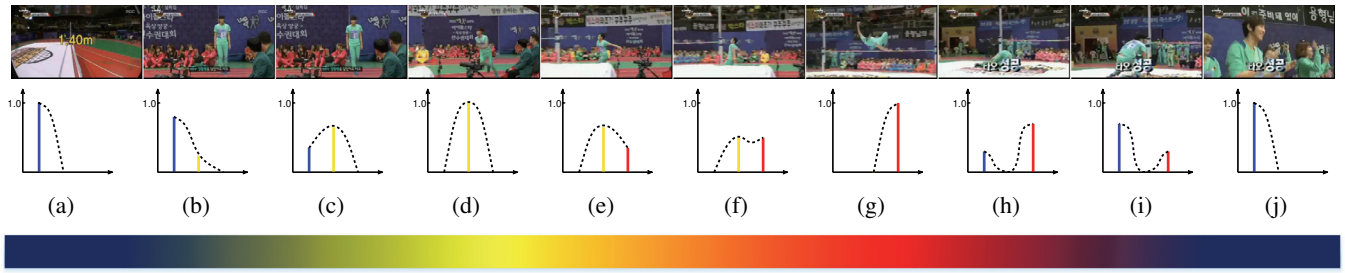



Figure 1: Example of the ambiguity among the semantic units of the *high jump* action. The first row shows the 10 key frames of the video. The second row shows the label distribution corresponding to each key frame, where blue, yellow and red represent the *background*, the *approaching* sub-action and the *taking off* sub-action, respectively. In the third row the same set of colors are used to represent the three semantic unit labels and the areas with mixture color represent the ambiguity between two adjacent semantic units.

the semantic units. In order to do this, we adopt a recently proposed machine learning paradigm called *Label Distribution Learning* (LDL) (Geng 2016). The label distribution covers a certain number of labels, representing the degree to which each label describes the instance. The description degrees of all the labels sum up to 1. LDL has been successfully applied to many real applications, such as age estimation (Geng, Zhou, and Smith-Miles 2007; Geng, Yin, and Zhou 2013), head pose estimation (Geng and Xia 2014), multi-label ranking for natural scene images (Geng and Luo 2014), facial expression recognition (Zhou, Xue, and Geng 2015), prediction of crowd opinion on movies (Geng and Hou 2015) and emotion analysis from texts (Zhou et al. 2016). In this paper, a label distribution is assigned to each video segment. Different description degrees in the label distribution correspond to different semantic units. Higher description degree indicates that the corresponding semantic unit label is more relevant to the segment. Some examples of the label distribution can be found in the second row of Fig. 1, where each label distribution corresponds to one key frame (standing for a segment) of the video. By label distribution, the proposed video parsing method allows a single segment to be associated with multiple semantic units with different importance, which can explicitly represent the ambiguity among the semantic units. The main contributions of this paper include:

1. LDL is adopted to model the ambiguity among the semantic units, which is ignored by most previous work. A bi-directional sliding process is proposed to generate the label distributions for the video segments in the sliding windows.
2. The time-consuming grammar inference process is avoided in the training process. Instead, an efficient optimization algorithm is adopted to build the mapping from a segment to its label distribution.
3. Based on the prediction of the label distribution model, a tree structure with explicit semantic meaning is constructed for the test videos.

The rest of this paper is organized as follows. First a soft video parsing method based on LDL is proposed. Then the experimental results are reported. Finally, conclusions and discussions are given.

## Soft Video Parsing by LDL

### LDL with Bi-directional Sliding Window

As mentioned before, in real-world videos, the boundary between two adjacent semantic units is often ambiguous. Consequently, a single semantic label might be insufficient to describe a particular video segment. Instead, we propose to use a label distribution (Geng 2016) to do that. The label distribution indicates the relative importance of the corresponding semantic units in each video segment. As we slide a window through the whole video, the label distribution of the acquired video segment varies at the same time. In this way, the graduality among the semantic units can be directly modeled. However, since the ground truth of most datasets usually just provides the positions of the start and end frames of each action, the transition points among the sub-actions are unknown as well as the label distributions for the video segments in the sliding windows. In order to deal with these problems, we propose in this section an iterative method to build an LDL model via a bi-directional sliding window process.

Firstly, the transition points among the sub-actions are initialized by dividing each action into equal-sized sub-actions. For example, as shown in Fig. 2, two sub-actions $a_1$ and $a_2$ of the action $A$ are equal-sized and $b$ stands for the background. Considering the general case, we suppose two adjacent segments $\boldsymbol{x}$ and $\boldsymbol{x}'$ are labeled by the semantic units $y_i$ and $y_i'$, respectively. For the forward sliding case, $\boldsymbol{x}'$ is to the right of $\boldsymbol{x}$. For the backward sliding case, $\boldsymbol{x}'$ is to the left of $\boldsymbol{x}$. The sliding window is initialized as $\boldsymbol{x}_0 = \boldsymbol{x}$ and keeps the length unchanged while sliding. The window is moved $p-1$ times ($p$ is a constant) equidistantly toward $\boldsymbol{x}'$ before reaching the union boundary of $\boldsymbol{x}$ and $\boldsymbol{x}'$ (the $p$-th shift may reach the union boundary). The label distribution for the $t$-th window $\boldsymbol{x}_t$ ($t = 0 \dots p-1$) is generated by

$$d_{\boldsymbol{x}_t}^{y_i'} = \frac{t \times \lambda \times \min(l(\boldsymbol{x}_t), l(\boldsymbol{x}'))}{p \times l(\boldsymbol{x}_t)}, \quad (1)$$

$$d_{\boldsymbol{x}_t}^{y_i} = 1 - d_{\boldsymbol{x}_t}^{y_i'}, \quad (2)$$

$$d_{\boldsymbol{x}_t}^{y} = 0, \; y \notin \{y_i, y_i'\}, \quad (3)$$

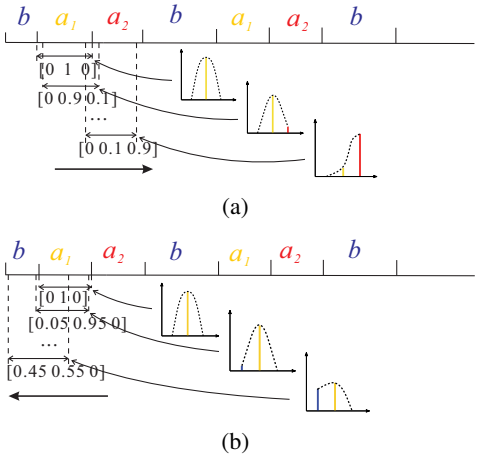



Figure 2: Examples for (a) forward sliding window and (b) backward sliding window.

where $l(\boldsymbol{x})$ is the length of $\boldsymbol{x}$ and $\lambda \in [0, 1]$ is a parameter that controls the ambiguity level, which is set to 1.0 without specific statement. Each time, the window is moved $\lambda \times \min(l(\boldsymbol{x}_t), l(\boldsymbol{x}'))/p$ frames. So, the $t$-th window will move $t \times \lambda \times \min(l(\boldsymbol{x}_t), l(\boldsymbol{x}'))/p$ frames into $\boldsymbol{x}'$. Thus $d^{y_i'}_{\boldsymbol{x}_t}$ actually calculates the ratio of the number of overlapping frames between $\boldsymbol{x}_t$ and $\boldsymbol{x}'$ to the total number of frames in $\boldsymbol{x}_t$.

For example in Fig. 2, the first $a_1$ segment slides forward in (a) and slides backward in (b), where $p$ is set as 10 and the first $a_1$ segment is set twice the length of the first $b$ segment. Each time, a label distribution $[d^{b}_{\boldsymbol{x}_t}\ d^{a_1}_{\boldsymbol{x}_t}\ d^{a_2}_{\boldsymbol{x}_t}]$ is generated by Eq. (1)-(3) for the $t$-th window ($t = 0 \ldots 9$). In this way, 10 segments with corresponding label distributions are generated for forward sliding and backward sliding, respectively.

Through the bi-directional sliding window process, each semantic unit segment will generate $2p - 1$ different segments with their corresponding label distributions (one is overlapping), all of which are then used as the training set to learn a mapping from a segment to its label distribution by LDL (Geng 2016). Let $\mathcal{X} = \mathcal{R}^q$ denote the input feature space, $\mathcal{Y} = \{y_1, y_2, \cdots, y_c\}$ denote the $c$ class labels. Given a training set $S = \{(\boldsymbol{x}_1, \boldsymbol{d}_1), (\boldsymbol{x}_2, \boldsymbol{d}_2), \cdots, (\boldsymbol{x}_n, \boldsymbol{d}_n)\}$, where $\boldsymbol{x}_i \in \mathcal{X}$, $\boldsymbol{d}_i = [d^{y_1}_{\boldsymbol{x}_i}, d^{y_2}_{\boldsymbol{x}_i}, \cdots, d^{y_c}_{\boldsymbol{x}_i}]$, the goal of LDL is to find the parameter vector $\boldsymbol{\theta}$ in a conditional mass function $p(y|\boldsymbol{x}_i; \boldsymbol{\theta})$ that can generate a label distribution similar to $\boldsymbol{d}_i$. Kullback-Leibler (KL) divergence is often used as the distance measure between the ground truth and the predicted distributions. One reasonable choice of $p(y|\boldsymbol{x}; \boldsymbol{\theta})$ is the maximum entropy model (Berger, Pietra, and Pietra 1996). Then the limited-memory quasi-Newton method L-BFGS (Liu and Nocedal 1989) is facilitated to find the optimal $\boldsymbol{\theta}$, which is denoted as $\boldsymbol{\theta}^*$.

After the initial LDL model is trained, it is reversely used to update the initial transition points among the sub-actions. Suppose $\boldsymbol{\tau}$ represents the vector composed by the position of the transition points within an action, $\boldsymbol{x}^i_{\boldsymbol{\tau}}$ represents the segment corresponding to the $i$-th sub-action determined by $\boldsymbol{\tau}$, $\boldsymbol{d}(\boldsymbol{x}^i_{\boldsymbol{\tau}}; \boldsymbol{\theta}^*)$ represents the label distribution predicted by the LDL model for $\boldsymbol{x}^i_{\boldsymbol{\tau}}$, then the optimum sub-action transition points for this action is determined by

$$\boldsymbol{\tau}^* = \underset{\boldsymbol{\tau}}{\operatorname{argmin}} \left( \sum_i D_{KL}(\boldsymbol{T}_i || \boldsymbol{d}(\boldsymbol{x}^i_{\boldsymbol{\tau}}; \boldsymbol{\theta}^*)) \right), \tag{4}$$

where $D_{KL}$ represents the KL divergence and $\boldsymbol{T}_i$ is the template label distribution for the $i$-th sub-action, where the description degree of the $i$-th sub-action is 1 and those for all the other semantic units are 0.

When the best sub-action transition points $\boldsymbol{\tau}^*$ for all the actions in the training videos are calculated by Eq. (4) in an exhaustive searching method, the bi-directional sliding window process is again applied to the updated segments to generate a new training set with label distributions, based on which a new LDL model is trained. This process repeats until the average difference of the transition points between two adjacent iterations is smaller than a predefined threshold. When the training process is finally converged, we obtain an LDL model that can be later used to predict the label distribution for any given segment. The whole iterative training process is summarized in the *Training* part of Algorithm 1.

## Grammar Model

As traditional regular grammars (Chomsky 1956) allows at most one terminal to be on the right side of the production rule, the Segmental Regular Grammars (SRG) (Pirsiavash and Ramanan 2014) is proposed for video parsing, where the production rules may generate any length of terminals: $X \to Yw_{1:z}$, $X \to w_{1:z}$, where $z$ stands for the length of the terminal $w$. For the sake of simplicity, we omit the subscript in this paper and use a terminal to stand for a certain number of frames. Thus a video can be parsed by recursively applying the production rules of SRG. For example, for the *high jump* action shown in Fig. 1, the production rules of the context-free grammars may be: $S \to SAb$, $S \to b$, $A \to a_1 a_2$, where $b$ stands for the *background*, $a_1$ and $a_2$ stand for the *approaching* and *taking off* sub-actions, respectively. By adding dummy nonterminals, the production rules could be converted to SRG which contains at most one nonterminal on the right side of the rule followed by a terminal: $S \to b$, $S \to Ab$, $A \to Ca_2$, $C \to Sa_1$. Thus each video can be parsed by SRG starting from S and a binary tree structure would be constructed with each leaf as one of the semantic units $b$, $a_1$ or $a_2$.

Each of the SRG rule $r \in \{X \to Yw\}$ has an associated score $s(\boldsymbol{v}, r, \hat{j}, j)$, which measures the degree the $k$-long ($k = j - \hat{j} + 1$) segment of the test video $\boldsymbol{v}$ matches the semantic unit $w$, where $\hat{j}$ and $j$ are the start and end frame of the $k$-long segment, respectively. The score is calculated by

$$s(\boldsymbol{v}, r, \hat{j}, j) = -D_{KL}(\boldsymbol{T}_r || \boldsymbol{d}(\boldsymbol{x}_{\mu_k}; \boldsymbol{\theta}^*)), \tag{5}$$

which is the negative KL divergence between the template distribution $\boldsymbol{T}_r$ for the rule $r$ and the predicted distribution. $\boldsymbol{T}_r$ is defined similarly as in Eq. (4), where the description degree for $w$ is 1 and those for all the other labels are 0. Assume $\mu_k$ represents the range from $\hat{j}$ to $j$, then $\boldsymbol{x}_{\mu_k}$ stands

**Algorithm 1** Soft Video Parsing

*Training:*

**Inputs:**

$\mathbf{\Phi}$: the training set $\{(\boldsymbol{v}_i, \boldsymbol{\gamma}_i) \mid 1 \leq i \leq n\}$, where $\boldsymbol{v}_i$ is the $i$-th training video, and $\boldsymbol{\gamma}_i$ is its ground truth annotations of the actions;

$\varepsilon$: the convergence threshold.

**Outputs:**

$\boldsymbol{\theta}^*$: the parameter vector of the LDL model.

Initialize all the sub-action transition points by equally divide the actions of each video in $\mathbf{\Phi}$ and $\delta$ as $+\infty$;
**while** $\delta > \varepsilon$ **do**
  Apply the bi-directional sliding window process to each video in $\mathbf{\Phi}$ to generate an LDL training set;
  Calculate $\boldsymbol{\theta}^*$ by L-BFGS;
  Update the sub-action transition points by Eq. (4);
  $\delta$ = the average difference of the current transition points from those of the last iteration;
**end while**
**return** $\boldsymbol{\theta}^*$;

*Testing:*

**Inputs:**

$\boldsymbol{v}$: the test video;

$\boldsymbol{\theta}^*$: the parameter vector of the LDL model;

$\rho_l$, $\rho_u$: the minimum and maximum semantic unit length.

**Outputs:**

$\Gamma$: the predicted parsing for $\boldsymbol{v}$.

$\pi[X, j] = -\infty, \forall\, \mathrm{j} \in \{\rho_l,\ \rho_l + 1,\ \ldots,\ l(\boldsymbol{v})\}$;
$j = \rho_l$;
**while** $j \leq l(\boldsymbol{v})$ **do**
  **for** $k = \rho_l : min(\rho_u, j)$ **do**
    $\hat{j} = j - k + 1$;
    **for** $r \in \{X \rightarrow Yw\}$ **do**
      Calculate $s(\boldsymbol{v}, r, \hat{j}, j)$ by Eq. (5);
    **end for**
  **end for**
  Calculate $\pi[X, j]$ by Eq. (6);
**end while**
**return** the optimal parsing tree $\Gamma$;

for the corresponding video segment. The predicted distribution of the video segment $\boldsymbol{d}(\boldsymbol{x}_{\mu_k}; \boldsymbol{\theta}^*)$ is calculated by the maximum entropy model and $\boldsymbol{\theta}^*$ is obtained from the *Training* part of Algorithm 1. Moreover, we can see from the SRG rule $r \in \{X \rightarrow Yw\}$ that all the nonterminals start from the first frame, so the score of the best parsing from frame 1 to $j$, $\pi[X, 1, j]$, can be simplified as $\pi[X, j]$. Assume that the semantic unit $w$ is at least $\rho_l$ and at most $\rho_u$ frames long, then for each frame $j$, we may search $k$ (the length of $w$) from $\rho_l$ to $min(\rho_u, j)$ and each possible rule for transition at $j - k$ to find the best transition point and rule for frame 1 to $j$. Thus, $\pi[X, j]$ can be recursively calculated by

$$\pi[X, j] = \max_{\substack{k=\rho_l \ldots min(\rho_u, j) \\ r \in \{X \rightarrow Yw\}}} \frac{m_{j-k} \cdot \pi[Y, j-k] + s(\boldsymbol{v}, r, \hat{j}, j)}{m_{j-k} + 1}, \tag{6}$$

where $m_{j-k}$ is the number of semantic unit segments in $Y$ from frame 1 to $j - k$. Thus the fraction item in Eq. (6) stands for the average score of all the $m_{j-k} + 1$ segments from frame 1 to $j$. Assuming $j$ to be the end frame of a test video, we would obtain the whole parsing of the video recursively. To improve efficiency, we calculate the $\pi[X, j]$ with $j$ from 1 to the end frame of the video by the dynamic programming process similar to the standard SRG method (denoted by SSRG) (Pirsiavash and Ramanan 2014), where Eq. (5) and Eq. (6) are both different, and store the best transition point $j - k$ for each rule $r$. The detailed process is summarized in the *Testing* part of Algorithm 1.

After that, the best parsing tree of the video could be inferred as follows. Assuming $j$ as the end frame of a test video, we find the rule of the highest score $\pi[X, j]$ from all possible rules $r \in \{X \rightarrow Yw\}$ and use the rule's corresponding transition point to replace the $j$. This process repeats until reaching the start frame of the video. Each time in the process we obtain a semantic unit segment of the video. Finally we get the best parsing tree encoded as $\Gamma = \{(r_t, j_t) :\ t = 1 \ldots N\}$, where $N$ is the total number of the semantic units in the parsing tree of the video, $j_t$ is the end frame of $t$-th segment and $r_t$ is the rule whose terminal instances the segment.

It should be noted that the dynamic programming inference is also involved in training of the SSRG (Pirsiavash and Ramanan 2014), while in our method, the training is via the LDL with bi-directional sliding window process. The complexity of the training process of these two algorithms are compared as follows. Both our method and SSRG are trained by iterating between learning the model parameters and updating the latent transition points among the sub-actions. The maximum iteration number is set as the same for both algorithms. The complexity of updating the latent transition points is also same for both algorithms. Thus, we only need to compare the Structured SVM used in SSRG and the L-BFGS in our method. Assume there are $n$ training videos and each video has at most $n_s$ semantic units, then at most $n((2p-1)(n_s-1)+1)$ segments can be generated by the bi-directional sliding process. Thus, the complexity of L-BFGS is $O(n\alpha p n_s)$, where $\alpha$ is the maximum number of correction matrices stored for the calculation of the Hessian matrix, which is usually smaller than 20. As for Structured SVM, the computational complexity is $O(n^2 n_r \rho_u n_{\boldsymbol{v}})$, where $n_r$ is the number of possible rules, $\rho_u$ is the maximum length of a semantic unit, and $n_{\boldsymbol{v}}$ is the maximum length of a video. Since $n_s \ll n_{\boldsymbol{v}}$, and $\alpha p \ll n n_r \rho_u$ (usually $n_r \geq 3$ and $\rho_u \geq 150$), the training of our method is much faster than that of SSRG.

# Experiments

## Methodology

**Dataset** We conduct our experiments on two benchmark datasets: THUMOS 2014 Detection Challenge dataset (Jiang et al. 2014) and MSR-II Action dataset (Yuan, Liu, and Wu 2011). We discard the videos which only contain actions in the THUMOS'14 dataset and use the union of the validation set and the test set as our dataset since we aim

at separating the actions and the background. The dataset contains 20 realistic sport actions and the total video length is more than 12 hours. Only the start and end frame labels of each action in the videos are annotated in the ground truth. The duration of action varies considerably from 0.2 seconds to 98.2 seconds. The number of actions in one video is within the range from 1 to 218. The MSR-II Action dataset contains 54 untrimmed video sequences recorded in a crowded environment, with an average length of 51 seconds. The videos include three categories of actions: hand waving, hand clapping and boxing.

**Feature representation** A bag-of-words (BoW) representation of the space-time interest points (STIP) feature (Laptev et al. 2008) is calculated for each video segment for its efficiency and effectiveness. The method of additive kernels (Vedaldi and Zisserman 2012) is applied to approximate a $\mathcal{X}^2$ kernel for the BoW feature to maintain efficiency while increasing the discriminative power. The priori knowledge of the length of each semantic unit is appended to the BoW feature. For the $i$-th video segment $\boldsymbol{x}_i$ in the video, we define a vector $\boldsymbol{\psi}(\boldsymbol{x}_i) = \frac{[1, l(\boldsymbol{x}_i), l(\boldsymbol{x}_i)^2]}{\hat{Z}}$, where $\hat{Z}$ is a normalization constant set as the square of mean length of the training videos. If the BoW feature is denoted by $\boldsymbol{\phi}(\boldsymbol{x}_i)$, then the final feature for $\boldsymbol{x}_i$ is $[\boldsymbol{\phi}(\boldsymbol{x}_i), \boldsymbol{\psi}(\boldsymbol{x}_i)]$.

**Baselines and evaluation** We compare against the state-of-the-art method based on Standard SRG (SSRG) (Pirsiavash and Ramanan 2014) and the re-scoring non-maximum suppression (RNMS) method (Wang et al. 2015), which is specialized for action localization. In SSRG, a latent structured SVM method is adopted to train the model and a dynamic programming method is used to infer a tree structure with a single label for each video segment. The RNMS trains an RBF-$\chi^2$ kernel SVM to seperate the action and background and adopts non-maximum suppression with zero overlap to find the higher scoring segments in all the candidate sliding windows.

We conduct 5-fold cross validation on the two datasets. For SSRG and our method (denoted by SP for Soft Parsing) we use the training set of the first fold to select the optimal number of sub-action via 3-fold cross validation. The optimal number is chosen from 1, 2 or 3 for each action category. After that, the optimal sub-action number is used for the rest folds. As for the parameters in SSRG and SP, the minimum semantic unit duration $\rho_l$ is set as 10 frames to eliminate the small fluctuation. For different numbers of sub-actions in each action, 1, 2 and 3, the maximum semantic unit duration $\rho_u$ is set as 300, 200 and 150, respectively and for the background, $\rho_u$ is set as 300. As for RNMS, the parameter $C$ of SVM is set by 3-fold cross validation on the training set of the first fold. It is chosen from the range $C \in \{3^{-2}, 3^{-1}, \cdots, 3^{7}\}$ (Wang et al. 2015). Moreover, the length of candidate sliding window is chosen from 10, 20, $\cdots$, 300 frames and the sliding step is set as 10 frames. Other parameters are set the same as in SSRG and SP.

The mean AP index of PASCAL 2012 (Everingham et al. 2011) and the mean frame labeling accuracy of 5-fold cross validation are adopted to evaluate the algorithms. An action detection is considered a true positive if its ratio of the intersection over the union with the ground truth is over 0.4, as it is consistent with visual inspection (Pirsiavash and Ramanan 2014). The confidence score is used to rank the video segments for the AP index. In our method (SP), it is calculated as follows. For each sub-action in the action, we calculate the KL divergence between the template distribution and its predicted distribution obtained by the maximum entropy model. Then the mean negative value of all the KL divergences is regarded as the confidence score for the whole action.

## Results

The comparative results of RNMS, SSRG and our method (SP) in the THUMOS'14 dataset on two measures, AP and frame labeling accuracy, are tabulated in Table. 1. The percentages of improvement over RNMS and SSRG are shown in the parentheses right after the performances of our method. The pair-wise t-tests with the significance level 0.1 are performed for each action. The •/▲ after the performance of SP means that SP performs significantly better than RNMS/SSRG, while the ○/△ means that SP performs significantly worse than RNMS/SSRG. Absence of symbols means that the performances of SP and RNMS/SSRG are not significantly different. As can be seen in Table. 1, on the AP measure, SP performs significantly better than RNMS on 10 actions and significantly better than SSRG on 6 actions. The improvement over RNMS/SSRG could be as high as $1464\%/157\%$ (on *bask-dunk*/*clif-dive*). In average, the AP of SP in 20 kinds of actions is $135\%/69\%$ higher than that of RNMS/SSRG. On the Frame Labeling Accuracy, SP performs significantly better than RNMS/SSRG on 19/15 actions and significantly worse than RNMS/SSRG on 0/1 action. The improvement over RNMS/SSRG could be as high as $526\%/140\%$ (on *base-pitch*), and the average Frame Labeling Accuracy of SP is $132\%/25\%$ higher than that of RNMS/SSRG.

Both SSRG and SP perform better than RNMS on 15 out of 20 actions on the two measures as they divide the action into sub-actions, which is consistent with the fact that these sports generally involves two or three steps. For example, the *basketball dunk* action are generally composed of *taking off* and *falling down* sub-actions and the *hammer throw* action often include the *swinging*, *turning* and *throwing* sub-actions. Moreover, considering the ambiguity among the semantic units, SP performs better than SSRG on 17 out of 20 actions on the two measures. For example, in the *javelin throw* action, the three semantic units (*background*, *running* and *throwing*) are interlaced with each other and it is hard to differentiate the exact boundary among them.

There are a few cases where our method does not perform the best. For example, our method SP performs worse than RNMS on the 'long-jump' action in AP. Considering the running sub-action occupies most time of the long-jump action, regarding long-jump as a whole without specially dealing with the sub-actions may be more advantageous. Moreover, SP performs worse than SSRG on the 'billiards' action in AP. The background, striking and rolling sub-actions seem to have less overlap with each other, which accords

Table 1: Average Precision (AP) (%) and Frame Labeling Accuracy (%) for the 20 kinds of sports in the THUMOS'14 dataset.

| Names | AP | | | | Frame Labeling Accuracy | | | |
|---|---|---|---|---|---|---|---|---|
| | RNMS | SSRG | SP | SP w. OAL | RNMS | SSRG | SP | SP w. OAL |
| base-pitch | 2.0 | 6.6 | 10.7 (435%,62%) | 20.6(93%) | 12.9 | 33.6 | 80.8 (526%,140%)●▲ | 81.4(1%) |
| bask-dunk | 1.1 | 12.9 | 17.2 (1464%,33%)● | 20.1 (17%) | 27.0 | 55.7 | 71.6 (165%,29%)●▲ | 71.8 (0%) |
| billiards | 2.4 | 3.7 | 3.1 (29%,-16%) | 5.8 (87%) | 17.4 | 54.9 | 74.6(329%,36%)●▲ | 75.4(1%) |
| clean-jerk | 13.4 | 8.7 | 20.1 (50%,131%)▲ | 26.7 (33%) | 40.2 | 43.4 | 58.5(46%,35%)●▲ | 62.0 (6%) |
| clif-dive | 3.6 | 14.6 | 37.5 (942%,157%)●▲ | 37.6 (0%) | 25.8 | 59.4 | 74.3(188%,25%)●▲ | 74.3(0%) |
| cric-bowl | 1.0 | 3.2 | 6(500%,87%) ● | 12.3(105%) | 16.2 | 50.4 | 77.2(377%,53%)●▲ | 78.6(2%) |
| cric-shot | 0.3 | 2.8 | 4.2(1300%,50%)● | 7.6(81%) | 14.8 | 56.4 | 78.7(432%,40%)●▲ | 80.1(2%) |
| diving | 3.4 | 8.9 | 22.5 (562%,153%)●▲ | 29.1 (29%) | 31.0 | 57.0 | 69.6 (125%,22%)●▲ | 70.9 (2%) |
| fris-catch | 7.9 | 6.3 | 9.9 (25%,57%)▲ | 12.6 (27%) | 27.9 | 57.0 | 61.2 (119%,7%)● | 66.1(8%) |
| golf-swing | 11.5 | 10.6 | 12.2(6%,15%) | 18.2(49%) | 51.7 | 59.2 | 54.5 (5%,-8%) | 61.8 (13%) |
| hamm-throw | 9.2 | 17.4 | 41.1 (347%,136%)●▲ | 42.1 (2%) | 32.0 | 48.7 | 63.8 (99%,31%)●▲ | 63.8 (0%) |
| high-jump | 8.2 | 12.2 | 24.1(194%,98%)● | 26.5(10%) | 28.1 | 52.2 | 63.8 (127%,22%)●▲ | 63.8 (0%) |
| jave-throw | 8.3 | 21.1 | 22.4 (170%,6%)● | 26.6 (19%) | 23.7 | 46.2 | 64.6 (173%,40%)●▲ | 64.6(0%) |
| long-jump | 35.3 | 18.1 | 26.7(-24%,48%) | 37.2(39%) | 33.6 | 47.5 | 66.1 (97%,39%)●▲ | 67.7 (2%) |
| pole-vault | 16.3 | 15.4 | 32(96%,108%) | 32.9(3%) | 37.9 | 52.5 | 61.9 (63%,18%)●▲ | 62.5(1%) |
| shotput | 12.9 | 13.9 | 15.7 (22%,13%) | 18.1(15%) | 37.9 | 51.5 | 60.9 (61%,18%) ●▲ | 61.7 (1%) |
| socc-pena | 5.7 | 16.5 | 17.8 (212%,8%) | 20.5(15%) | 21.2 | 53.2 | 65.8 (210%,24%)●▲ | 68.4 (4%) |
| tenn-swing | 2.0 | 6.2 | 6.9 (245%,11%) ● | 6.9(0%) | 24.5 | 66.9 | 58.4 (138%,-13%)●△ | 64.4 (10%) |
| disc-throw | 5.9 | 11.5 | 25.5 (332%,122%)●▲ | 27.1 (6%) | 35.4 | 56.3 | 56.9 (61%,1%)● | 62.3 (9%) |
| voll-spik | 5.1 | 6.0 | 9.8 (92%,63%) | 14.4 (47%) | 30.5 | 54.5 | 56.7 (84%,4%)● | 59.8 (5%) |
| Average | 7.8 | 10.8 | 18.3 (135%,69%) | 22.1 (21%) | 28.5 | 52.8 | 66.0 (132%,25%) | 68.1 (3%) |

better with the hard boundary assumption in SSRG.

As for the MSR-II action dataset, on the two measures SP performs significantly better than RNMS and SSRG on all three actions. On the AP measure, the average performance is improved by $456\%$ and $166\%$ respectively and on the Frame Labeling Accuracy measure, the average performance is improved by $408\%$ and $16\%$ respectively, as shown in Table. 2. The three actions in the dataset are all composed of repetitive movements, which can be regarded as sub-actions. Considering the sub-actions in the actions, SP and SSRG perform better than RNMS. By using the label distribution to model the graduality among the semantic units, SP further improve the performance of SSRG on all three actions.

Moreover, we experiment with different ambiguity level $\lambda$ for each action from 0 to 1 with the increment $1/8$ to reveal different ambiguity of different actions in the THUMOS'14 dataset. Note that $\lambda = 0$ stands for the situation that we do not slide the window backward or forward and only use the initial label distribution of the window. Experimental results show that different actions perform best on different ambiguity levels. For example, as shown in Fig. 3, the *tennis swing* action performs best when $\lambda = 1$. The transition between $swing$ and the $hit$ sub-actions of the *tennis swing* action are quite smooth and the boundary between them is rather difficult to differentiate. The *diving* action performs best when $\lambda = 0$. As can be seen from the videos that most of the *diving* actions happen suddenly since most of the *diving* videos are a collection of highlights of diving. Thus the action has little ambiguity with the *background*. The *javelin throw* and the *billiards* actions perform best when $\lambda = 0.75$ and $\lambda = 0.25$, respectively. Their transitions among the semantic units are relatively smooth but still differentiable. In Table. 1, we compare the performance of SP with Optimal Ambiguity Level (OAL) with the performance of SP (where $\lambda = 1$). As can be seen, by using the optimal ambiguity level, the average AP of SP can be further improved by $21\%$, and the average Frame Labeling Accuracy can be further improved by $3\%$.

## Conclusion

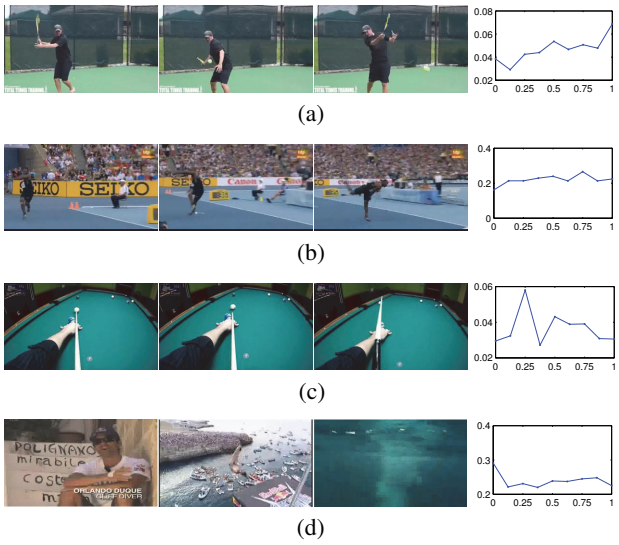
In this paper, we propose a soft video parsing method based on label distribution learning. A bi-directional sliding window process is used to generate the label distributions for different video segments, and the efficient L-BFGS algorithm is used to learn the mapping from a video segment to its label distribution. Then, a soft grammar parsing method is proposed to parse the given test videos in linear-time, which is practical for long stream videos. We experiment on 2 benchmark video datasets with various actions and duration and prove the effectiveness of our soft video parsing method. We show that the ambiguity among the semantic units can be well modeled by the label distribution and different actions might have different optimal ambiguous levels among the semantic units.

## Acknowledgments

This research was supported by National Science Foundation of China (61622203, 61273300, 61232007), Jiangsu Natural Science Funds for Distinguished Young Scholar (BK20140022), Collaborative Innovation Center of Novel Software Technology and Industrialization, Collaborative Innovation Center of Wireless Communications Technology, Fundamental Research Funds for the Central Universities, and Research Innovation Program for College Graduates of Jiangsu Province (KYLX15_0119).

Table 2: Average Precision (AP) (%) and Frame Labeling Accuracy (%) for the 3 kinds of actions in the MSR-II action dataset.

| Names | AP | | | Frame Labeling Accuracy | | |
|---|---|---|---|---|---|---|
| | RNMS | SSRG | SP | RNMS | SSRG | SP |
| hand clapping | 4.9 | 5.5 | 25.7(424%,367%)●▲ | 13.3 | 65.7 | 75.1 (465%,13%) ●▲ |
| hand waving | 5.2 | 15.0 | 26.7(413%,78%)●▲ | 18.4 | 62.9 | 76.0 (313%,21%) ●▲ |
| boxing | 2.8 | 6.5 | 19.3 (589%,197%)●▲ | 14.3 | 71.6 | 82.7 (478%,16%)●▲ |
| Average | 4.3 | 9.0 | 23.9(456%,166%) | 15.3 | 66.7 | 77.9 (408%,16%) |



(a) (b) (c) (d)

Figure 3: The key frames of four typical actions, *tennis swing*(a), *javelin throw*(b), *billiards*(c), and *diving*(d), with a descending order of optimal ambiguity level. The horizontal axis corresponds to different values of the ambiguity level $\lambda$, and the vertical axis corresponds to the AP measure.

## References


Berger, A. L.; Pietra, V. J. D.; and Pietra, S. A. D. 1996. A maximum entropy approach to natural language processing. *Computational linguistics* 22(1):39–71.

Chomsky, N. 1956. Three models for the description of language. *IEEE Trans. Information Theory* 2(3):113–124.

Everingham, M.; Van Gool, L.; Williams, C.; Winn, J.; and Zisserman, A. 2011. The pascal visual object classes challenge 2012.

Geng, X., and Hou, P. 2015. Pre-release prediction of crowd opinion on movies by label distribution learning. In *Proc. the 24th International Joint Conference on Artificial Intelligence*, 3511–3517.

Geng, X., and Luo, L. 2014. Multilabel ranking with inconsistent rankers. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 3742–3747.

Geng, X., and Xia, Y. 2014. Head pose estimation based on multivariate label distribution. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 1837–1842.

Geng, X.; Yin, C.; and Zhou, Z.-H. 2013. Facial age estimation by learning from label distributions. *IEEE Trans. Pattern Anal. Machine Intell.* 35(10):2401–2412.

Geng, X.; Zhou, Z.-H.; and Smith-Miles, K. 2007. Automatic age estimation based on facial aging patterns. *IEEE Trans. Pattern Anal. Machine Intell.* 29(12):2234–2240.

Geng, X. 2016. Label distribution learning. *IEEE Trans. on Knowledge and Data Engineering* 28(7):1734–1748.

Hoai, M.; Lan, Z.-Z.; and De la Torre, F. 2011. Joint segmentation and classification of human actions in video. In *IEEE Conf. Computer Vision and Pattern Recognition*, 3265–3272.

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

Laptev, I.; Marszałek, M.; Schmid, C.; and Rozenfeld, B. 2008. Learning realistic human actions from movies. In *IEEE Conf. Computer Vision and Pattern Recognition*, 1–8.

Liu, D. C., and Nocedal, J. 1989. On the limited memory bfgs method for large scale optimization. *Mathematical programming* 45(1-3):503–528.

Neubeck, A., and Van Gool, L. 2006. Efficient non-maximum suppression. In *IEEE Conf. 18th International Conference on Pattern Recognition*, volume 3, 850–855.

Oneata, D.; Verbeek, J.; and Schmid, C. 2014. The lear submission at thumos 2014.

Pirsiavash, H., and Ramanan, D. 2014. Parsing videos of actions with segmental grammars. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 612–619.

Shi, Q.; Cheng, L.; Wang, L.; and Smola, A. 2011. Human action segmentation and recognition using discriminative semi-markov models. *International Journal of Computer Vision* 93(1):22–32.

Tang, K.; Li, F.-F.; and Koller, D. 2012. Learning latent temporal structure for complex event detection. In *IEEE Conf. Computer Vision and Pattern Recognition*, 1250–1257.

Vedaldi, A., and Zisserman, A. 2012. Efficient additive kernels via explicit feature maps. *IEEE Trans. Pattern Anal. Machine Intell.* 34(3):480–492.

Wang, H.; Oneata, D.; Verbeek, J.; and Schmid, C. 2015. A robust and efficient video representation for action recognition. *International Journal of Computer Vision* 1–20.

Yuan, J.; Liu, Z.; and Wu, Y. 2011. Discriminative video pattern search for efficient action detection. *IEEE Trans. Pattern Anal. Machine Intell.* 33(9):1728–1743.

Zhou, D.; Zhou, Y.; Zhang, X.; Zhao, Q.; and Geng, X. 2016. Emotion distribution learning from texts. In *Proc. Conf. Empirical Methods in Natural Language Processing*.

Zhou, Y.; Xue, H.; and Geng, X. 2015. Emotion distribution recognition from facial expressions. In *Proc. the 23rd Annual ACM Conference on Multimedia Conference*, 1247–1250.