Temporal Action Detection using a Statistical Language Model

Alexander Richard, Juergen Gall
University of Bonn, Germany
{richard,gall}@iai.uni-bonn.de

Abstract

While current approaches to action recognition on pre-segmented video clips already achieve high accuracies, temporal action detection is still far from comparably good results. Automatically locating and classifying the relevant action segments in videos of varying lengths proves to be a challenging task. We propose a novel method for temporal action detection including statistical length and language modeling to represent temporal and contextual structure. Our approach aims at globally optimizing the joint probability of three components, a length and language model and a discriminative action model, without making intermediate decisions. The problem of finding the most likely action sequence and the corresponding segment boundaries in an exponentially large search space is addressed by dynamic programming. We provide an extensive evaluation of each model component on Thumos 14, a large action detection dataset, and report state-of-the-art results on three datasets.

1. Introduction

Action recognition is a major research topic in computer vision since decades due to its applications in various fields like video understanding, human-computer-interaction, or surveillance. In recent years, there has been a major progress on classifying pre-segmented video clips. On challenging large-scale datasets like UCF-101 [23], current approaches for video-clip classification already achieve accuracies of eighty percent and more [28, 10, 22]. However, when the videos are not pre-segmented and the task requires the temporal segmentation of the activities in a video, as it is required for the temporal action detection task of Thumos 14 [8], current approaches struggle to achieve good results.

One reason is that an action class can be arbitrarily long, *e.g.*, a background class, but it can also take only a few frames in a video. Without a pre-segmentation of the video, the duration of the action classes needs to be well modeled. Another difference between video clip classification and temporal action detection is the relevance of the context. While video clips can already be well recognized by using only spatial [22] or context information [10], the differences between the frames where an activity occurs and the rest of the video are more subtle. For this task, the context of the temporal change is more important than the spatial context.

The most successful methods for temporal action detection on a dataset like Thumos 14 [8] follow a two step approach. They first extract segments from the video using a sliding temporal window and classify each segment in a second step. The final segmentation is then achieved by greedily selecting the segments with the highest scores [17, 19].

In this paper, we present an approach to temporal action detection that avoids a greedy approximation and aims to find the globally most likely action sequence in a single step by solving the segmentation and classification task jointly. Our model incorporates information about the duration of an action class by a length model and the temporal context by a language model. The length and language model are combined with a discriminative model for recognizing actions. We further show how inference can be efficiently performed using dynamic programming.

In our experiments, we provide an extensive analysis of our approach on three datasets where we evaluate the impact of the length and language model in detail. Our model achieves state-of-the-art accuracy for temporal action detection on the challenging Thumos 14 benchmark [8].

2. Related Work

Recent action recognition systems, which combine Fisher vectors of improved dense trajectories [28] or CNN features [10, 22, 7] with a classifier like a support vector machine (SVM), achieve very good performances for video clip classification on various real world datasets.

For temporal action detection, most approaches incorporate these classifiers into the detector. For instance, a sliding temporal window approach with a greedy non-maximum suppression can be applied to locate the action segments [19, 17, 29, 9]. In the context of spatio-temporal action detection, the number of windows can be reduced by finding good action proposals [6, 26] or optimal action tubes using

branch-and-bound [32]. These methods, however, focus on the spatio-temporal localization of a single action in a short video clip and cannot be applied to long videos containing a large number of different actions. Other approaches model the sequence with structured temporal models. Early works use hidden Markov models for action segmentation, see [30] for a survey. Shi *et al*. [21] use a semi-Markov model in combination with three different feature types for segment boundaries, segment content, and interaction between neighboring segments, respectively. Similar to [5, 4], in [21] action detection is formulated as a max-margin problem, which is solved by an SVM, and specific features that correlate with transitions between classes are proposed. In our work, on the contrary, none of these features are used, but an explicit length and language model are proposed. Also note that the dynamic programming used in [5, 4, 21] aims at finding a segmentation that maximizes the individual SVM scores, whereas we use dynamic programming to obtain the solution that maximizes our probabilistic model consisting of action, length, and language model.

With the publication of datasets providing action annotations on multiple granularity levels like Breakfast [11] or 50 Salads [24], hierarchical models using context free stochastic grammars gained attention [18, 11, 27]. Vo and Bobick [27] use a Bayes network and model the temporal structure of high level activities with a grammar. AND- and OR-rules define possible compositions of actions and the optimal hierarchical activity composition is computed via a message passing algorithm. The approach of Kuehne *et al*. [11, 12] similarly requires the definition of a context free grammar to model temporal structures. In this work, activities are treated as compositions of smaller sub-actions, each of which is modeled with a hidden Markov model.

The use of multiple types of granularity has also been explored without grammars. Ni *et al*. [16] track hands and object parts to infer hand-object interactions and compute dense trajectories around the tracked positions. They outperform the sliding window approach from [19] on the MPII-Cooking dataset. In [13], mid-level action elements are generated by concatenating low-level actions of different granularity levels. This approach allows for a multi-resolution reasoning even if only low level actions are annotated. Sharir and Tuytelaars [20] propose to divide the video into a spatio-temporal grid and compute the action chain with highest likelihood over the cells. Although the approach is inferior to [19] or [16] for temporal localization, it also predicts the spatial location of the actions.

In order to model long-term relations in complex event detection tasks, the authors of [3] propose the "sequence memoizer", which is a hierarchical Bayesian non-parametric model, for joint detection and classification of events. In [2], events are recognized by modeling temporal dynamics of mid-level concept detectors. Mettes *et al*. [14] apply a bag-of-fragments approach to event detection and obtain a precise temporal event localization.

Finally, Sun *et al*. [25] train long short-term memory network (LSTM) based fine-grained action detectors on both weakly labeled videos, where only video-level annotations without segmentation are available, and noisily tagged web images.

3. Temporal Action Detection

We propose a probabilistic model for temporal action detection that jointly models the segmentation and classification task. We first describe the model in Section 3.1 and present in Section 3.2 an approach for exact inference using dynamic programming.

3.1. Model

Given a video with T frames, let $\mathbf{x}_1^T$ be a sequence of feature vectors $\mathbf{x}_t \in \mathbb{R}^D$ from a D-dimensional input space representing the video. Let further $\mathcal{C} = \{1, \dots, C\}$ be the set of C possible action classes. Our goal is to segment the given input sequence into an unknown number of N segments and assign an action class to each of the segments. More specifically, we aim to find the sequence of action end positions t_1^N and corresponding action classes c_1^N that are most likely for the given video, *i.e*.

$$\max_{N, t_1^N, c_1^N} \left\{ p(c_1^N, t_1^N | \mathbf{x}_1^T) \right\} \tag{1}$$

$$= \max_{N, t_1^N, c_1^N} \left\{ p(c_1^N) p(t_1^N | c_1^N) p(\mathbf{x}_1^T | c_1^N, t_1^N) \right\}. \tag{2}$$

Note that the division by $p(\mathbf{x}_1^T)$ has been dropped in Equation (2) as it does not affect the maximizing arguments. The formulation induces a model consisting of three components. The first one, $p(c_1^N)$, is a *context* or *language model*, providing probabilities for the sequence of action labels assigned to each video segment. We stick to the term *language model* as this type of model has been developed in the context of natural language processing in order to determine the likelihood of word sequences.

The second component, $p(t_1^N | c_1^N)$, determines the ending points of the segments. Note that our model does not allow gaps, *i.e*. a sequence ending at t_n starts at $t_{n-1} + 1$, exactly one frame after the previous segment ends. This is no restriction if background is also modeled as an action class. Hence, t_1^N specifies the length of each segment and we call this component the *length model*.

The third component is the *action model*, providing the actual probability of a feature sequence $\mathbf{x}_1^T$ being generated by the given segmentation t_1^N and class labeling c_1^N.

3.1.1 Language Model

As language model, we use an m-gram,

$$p(c_1^N) = \prod_{n=1}^{N} p(c_n|c_1^{n-1}) = \prod_{n=1}^{N} p(c_n|c_{n-m}^{n-1}), \quad (3)$$

where the action class c_n is assumed to depend only on the m preceding action classes. At the beginning of a sequence, the preceding classes are assumed to be virtual sequence start classes, *i.e.* $c_k = c_\$$ for $k \leq 0$. Maximum likelihood estimation leads to the concrete model

$$p(c|h) = \frac{N(h,c)}{N(h,\cdot)}, \quad (4)$$

where h is a sequence of m preceding classes, *e.g.* c_{n-m}^{n-1} for $c = c_n$, and $N(h,c)$ is the count of occurrences of class c with the history h in the training data. Note that particularly for larger histories, $N(h,c)$ may be zero and such a sequence of action classes could never be detected. In order to deal with these unseen events, we use linear discounting with backing-off [15], *i.e.*

$$p(c|h) = \begin{cases} (1-\lambda) \cdot \frac{N(h,c)}{N(h,\cdot)}, & \text{if } N(h,c) > 0, \\ \lambda \cdot \frac{p(c|h')}{\sum_{c':N(h,c')=0} p(c'|h')}, & \text{otherwise.} \end{cases} \quad (5)$$

The parameter λ assigns a certain amount of probability mass to unseen events and is obtained using maximum likelihood estimation in combination with leaving-one-out, see [15] for details. For unseen events, we back-off to $p(c|h')$, an m-gram of lower order, *e.g.* a bigram ($m = 1$) if $p(c|h)$ is a trigram ($m = 2$). The renormalization is required for a proper probability distribution.

3.1.2 Length Model

For the length model, we assume a first-order dependence on the ending times, *i.e.*

$$p(t_1^N|c_1^N) = \prod_{n=1}^{N} p(t_n|c_1^N, t_1^{n-1}) = \prod_{n=1}^{N} p(t_n|c_n, t_{n-1}). \quad (6)$$

Note that we also simplified the distribution to be dependent on the class c_n of segment n only rather than on all classes c_1^N. Further, we assume that the ending time t_n does not depend on the actual position but only on the distance to t_{n-1}. Thus,

$$p(t_1^N|c_1^N) = \prod_{n=1}^{N} p(l_n|c_n) \quad (7)$$

where $l_n = t_n - t_{n-1}$ is the length of the segment. The distribution $p(l|c)$ can be modeled with any discrete probability distribution defined on the natural numbers. We investigate a class-dependent and class-independent Poisson distribution as well as a class-independent length model based on the average length μ of all actions,

$$p(l|c) \propto \begin{cases} 1, & \text{if } l \leq \mu, \\ \alpha^{l-\mu}, & \text{otherwise,} \end{cases} \quad (8)$$

where α is a decay factor. In the following, this model is referred to as *mean length model*. While the Poisson model prefers segments with lengths that are more likely according to the training data, the mean length model only ensures that no unreasonably long action segments are hypothesized. Without any restriction of the length, the system would tend to hypothesize a small number of long segments in order to avoid the penalty induced by the language model each time a new segment is hypothesized.

3.1.3 Action Model

In action classification, discriminative models such as support vector machines or convolutional neural networks achieve good performance [28, 22, 10]. These kinds of models can be viewed as a class posterior distribution $p(c|\mathbf{x}_1^T)$, or $p(c|\mathbf{x}_{t_{n-1}+1}^{t_n})$ for action segments in the domain of temporal action detection, respectively. Particularly when using Fisher vectors of improved dense trajectories, a linear classifier such as a support vector machine performs well [28]. We stick to this finding but replace the support vector machine with a log-linear model of the form

$$p(c|\mathbf{x}_{t_{n-1}+1}^{t_n}) = \text{softmax}(\mathbf{W}^T f(\mathbf{x}_{t_{n-1}+1}^{t_n}) + \mathbf{b}), \quad (9)$$

where $\mathbf{W}$ is a weight matrix, $\mathbf{b}$ the bias, and $f(\mathbf{x}_{t_{n-1}+1}^{t_n})$ the Fisher vector computed on the video segment $[t_{n-1}+1, t_n]$. The log-linear model is also a linear classifier but in contrast to support vector machines, it directly models a class posterior distribution. The parameters $\mathbf{W}$ and $\mathbf{b}$ can be estimated from the pre-segmented training data using gradient based optimization.

In the following, we show how to incorporate such a segment-based posterior distribution into our action model. We start with a simple factorization of $p(\mathbf{x}_1^T|c_1^N, t_1^N)$. Assuming independence of the video frames, we can rewrite the action model as a product of action segments,

$$p(\mathbf{x}_1^T|c_1^N, t_1^N) = \prod_{n=1}^{N} p(\mathbf{x}_{t_{n-1}+1}^{t_n}|c_1^N, t_1^N) \quad (10)$$

$$= \prod_{n=1}^{N} \prod_{t=t_{n-1}+1}^{t_n} p(\mathbf{x}_t|c_t, t_{n-1}^n). \quad (11)$$

Using Bayes' Theorem, Equation (11) can be transformed to contain a class posterior distribution,

$$p(\mathbf{x}_1^T|c_1^N,t_1^N) = \prod_{n=1}^{N}\prod_{t=t_{n-1}+1}^{t_n} p(c_t|\mathbf{x}_t,t_{n-1}^n)\frac{p(\mathbf{x}_t|t_{n-1}^n)}{p(c_t|t_{n-1}^n)}. \tag{12}$$

Due to the dependence on the segment start and end position, we assume that the class posterior has the same probability for each frame within the segment, *i.e.*

$$\prod_{t=t_{n-1}+1}^{t_n} p(c_t|\mathbf{x}_t,t_{n-1}^n) = p(c_n|\mathbf{x}_{t_{n-1}+1}^{t_n})^{l_n}, \tag{13}$$

where $l_n = t_n - t_{n-1}$ is again the length of segment n. Coming back to Equation (12), we further assume that neither the frame prior nor the class prior depend on the segment start and end positions. Together with Equation (13), this leads to

$$p(\mathbf{x}_1^T|c_1^N,t_1^N) = \prod_{n=1}^{N}\left(\frac{p(c_n|\mathbf{x}_{t_{n-1}+1}^{t_n})}{p(c_n)}\right)^{l_n}\prod_{t=1}^{T}p(\mathbf{x}_t). \tag{14}$$

In practice, we found that a uniform class prior $p(c)$ works well, so that factor can be omitted. Moreover, the product over the frame priors $p(\mathbf{x}_t)$ is independent of the arguments we maximize over and can thus also be omitted in the maximization.

Inserting all these results into Equation (2) leads to the final model

$$\max_{N,c_1^N,t_1^N}\left\{\prod_{n=1}^{N} p(c_n|c_{n-m}^{n-1})\cdot p(l_n|c_n)\cdot p(c_n|\mathbf{x}_{t_{n-1}+1}^{t_n})^{l_n}\right\}. \tag{15}$$

3.2. Inference

We use dynamic programming to efficiently solve the maximization problem from Equation (15). For terms of simplicity, we derive the recursion equations for a bigram language model ($m = 1$) only. The modifications that are required for higher order language models are straightforward.

In order to enable dynamic programming over the time frames $1,\dots,T$, we transform the product from Equation (15) to run over the time rather than over the number of segments. To simplify notation, let $s(t)$ be a function that maps frame t onto its corresponding segment number, *i.e.*

$$s(t) = n \quad \Leftrightarrow \quad t_{n-1} < t \leq t_n. \tag{16}$$

Then, Equation (15) can be rewritten as

$$\max_{N,c_1^N,t_1^N}\left\{\prod_{t=1}^{T}\left[p(c_{s(t)}|c_{s(t)-1})\cdot p(l_{s(t)}|c_{s(t)})\cdot p(c_{s(t)}|\mathbf{x}_{t_{s(t)-1}+1}^{t_{s(t)}})^{l_{s(t)}}\right]^{\delta(t,t_{s(t)})}\right\}, \tag{17}$$

where $\delta(t,t_{s(t)})$ is the Kronecker delta function and is one if and only if t is the ending time of segment $s(t)$. This way, the N factors from Equation (15) are sustained and the factors for the times t that are not a segment end time are one.

We now define an auxiliary function $Q(\tau,c)$ that specifies the best segmentation of the video up to time τ with class c ending at τ. Enforcing $c_n = c$ and $t_n = \tau$, we obtain

$$Q(\tau,c) = \max_{n,c_1^{n-1},t_1^{n-1}}\left\{\prod_{t=1}^{\tau}\left[p(c_{s(t)}|c_{s(t)-1})\cdot p(l_{s(t)}|c_{s(t)})\cdot p(c_{s(t)}|\mathbf{x}_{t_{s(t)-1}+1}^{t_{s(t)}})^{l_{s(t)}}\right]^{\delta(t,t_{s(t)})}\right\}. \tag{18}$$

Isolating the factors of the last segment and renaming $c_{n-1} = \tilde{c}$ and $l_n = l$ leads to the recursive equation

$$\begin{aligned} Q(\tau,c) &= \max_{n,c_1^{n-1},t_1^{n-1}}\left\{\prod_{t=1}^{\tau-l}\left[p(c_{s(t)}|c_{s(t)-1})\cdot p(l_{s(t)}|c_{s(t)})\cdot p(c_{s(t)}|\mathbf{x}_{t_{s(t)-1}+1}^{t_{s(t)}})^{l_{s(t)}}\right]^{\delta(t,t_{s(t)})}\right. \\ &\qquad \left.\cdot\, p(c|\tilde{c})\cdot p(l|c)\cdot p(c|\mathbf{x}_{\tau-l+1}^{\tau})^{l}\right\} \\ &= \max_{l,\tilde{c}}\left\{Q(\tau-l,\tilde{c})\cdot p(c|\tilde{c})\cdot p(l|c)\cdot p(c|\mathbf{x}_{\tau-l+1}^{\tau})^{l}\right\}. \end{aligned} \tag{19}$$

The score of the best segmentation in the sense of Equation (15) is now given by $\max_c Q(T,c)$. In order to reconstruct the best action segmentation, two additional traceback arrays need to be stored:

$$A(\tau,c) = \arg\max_{l}\left\{\max_{\tilde{c}}\ Q(\tau-l,\tilde{c})\cdot p(c|\tilde{c})\cdot p(l|c)\cdot p(c|\mathbf{x}_{\tau-l+1}^{\tau})^{l}\right\} \tag{20}$$

is the best-scoring length of the segment with class c ending at time τ and

$$B(\tau,c) = \arg\max_{\tilde{c}}\left\{\max_{l}\ Q(\tau-l,\tilde{c})\cdot p(c|\tilde{c})\cdot p(l|c)\cdot p(c|\mathbf{x}_{\tau-l+1}^{\tau})^{l}\right\} \tag{21}$$

is the best predecessor class of the segment ranging from $[\tau - A(\tau,c)+1,\tau]$ with class c. The optimal segmentation can then be reconstructed using Algorithm 1. Starting at the last frame T and the best ending class c at this frame, the best hypothesized segment start frame can be obtained as $T-A(T,c)+1$. The ending frame of the preceding segment is then $T-A(T,c)$ and the best hypothesized class is stored in $B(T,c)$. The optimal segmentation is reconstructed by iterating this scheme until the first frame is reached.

Algorithm 1 Reconstruction of the best segmentation

```
segments ← []
(τ, c) ← (T, max_c̃ Q(T, c̃))
while τ > 0 do
    t_s ← τ − A(τ, c) + 1
    t_e ← τ
    segments ← [(t_s, t_e, c), segments]
    (τ, c) ← (τ − A(τ, c), B(τ, c))
end while
return segments
```

3.3. Runtime

Since for each frame each possible action length needs to be evaluated, and for each class each predecessor class has to be considered, inference is quadratic in the number of frames classes, *i.e*. $\mathcal{O}(C^2T^2)$. However, limiting the maximal action length to a constant L, it can easily be reduced to $\mathcal{O}(C^2LT)$, allowing to process long videos. For higher order language models, predecessor classes also need to be stored. For a trigram language model, for instance, a function $Q(\tau, c, \tilde{c})$ needs to be computed, which increases the runtime to $\mathcal{O}(C^3LT)$. In practice, however, the runtime is dominated by the computation of the action model, which is not affected by an increased history in the language model. Thus, the difference of using a bigram or a trigram language model does hardly affect the runtime at all.

4. Experiments

Datasets. We evaluate our method on three datasets. A detailed analysis is provided on **Thumos 14** [8], a large dataset for action recognition and temporal action detection. The dataset offers a vast amount of training data, *i.e*. the videos from UCF101 [23], a set of $2,500$ background videos, and a validation set with $1,010$ temporally untrimmed videos of which 200 are temporally annotated with the 20 classes relevant for the detection task. For training, we only use these 200 videos from the validation set and the videos from UCF101 corresponding to the relevant 20 classes. The test set comprises 212 temporally annotated videos.

MPII-Cooking [19] is a large dataset for fine grained action detection of cooking activities. It contains more than 8 hours of video with recordings of 12 different persons performing 65 different cooking related actions, including a class for background activity. We follow the protocol of [19] and use leave-one-person-out cross-validation, resulting in seven splits.

Finally, we conduct experiments on **50 Salads** [24], a dataset originally designed for hierarchical activity recognition. There are three high level activities and a set of 17 low level activities of finer granularity. Since our method is not designed for hierarchical activity detection, we report detection results on the low level only.

Setup. The action model is trained by segmenting the training data according to the ground truth and computing a Fisher vector of improved dense trajectories [28] for each segment. The ground truth annotation of the training data is also used to estimate the length- and language model.

For detection, we extract unnormalized Fisher vectors of improved dense trajectories for each video frame and store the result as an integral image. This way, the Fisher vector for an arbitrary hypothesized segment can be computed efficiently by looking up the segment start and end time in the integral image and applying the normalization.

We compare our method to a sliding window baseline similar to the one used in [19]. Starting with a window size of 30 frames and a step size of 10 frames, both values are increased by a factor of $\sqrt{2}$ until the window size exceeds $1,000$ frames. Non-maximum suppression is then applied to remove all overlapping windows. As classifier, we use the log-linear model from Equation (9) that is also used in our method.

For our approach, the maximal action length is limited to $1,000$ frames and we increment t by 10 instead of 1 in Equation (17), *i.e*. we only evaluate every 10^{th} frame to reduce runtime. If not mentioned otherwise, we use the mean length model from Equation (8). With these settings, our algorithm needs 7.5h on a CPU with eight 1.2GHz cores for inference on Thumos 14. The code is available online.[1] [2]

Evaluation Protocol. For the evaluation on Thumos 14, we use the official evaluation script provided by the authors of [8]. The script computes mean average precision (mAP) over the detections. A detection is marked as correct if its intersection over union ratio is larger than some overlap threshold. Since we find this evaluation method very useful as it also gives insight in how a method performs for various overlap ratios, we also apply it to MPII-Cooking and 50 Salads. For MPII-Cooking, we additionally report precision and recall as well as single class mAP based on the midpoint hit criterion as proposed in [19] to be able to compare to other methods using this dataset.

Language Model. In this section, we evaluate the effect of the language model on the performance of the system. To this end, we trained different kinds of language models on the temporally annotated validation set and compared their strength and their effect on the detection.

The strength of a language model can be measured using the perplexity [1, 15]. For a single sequence with N action classes, it is defined as

$$\mathrm{PP} = \Big(\prod_{n=1}^{N} p(c_n | c_{n-m}^{n-1}) \Big)^{-\frac{1}{N}}. \tag{22}$$

[1] https://github.com/alexanderrichard/squirrel

[2] We appreciate the permission to reuse the matrix classes from [31].

m-gram		Perplexity validation	Perplexity test	Overlap 0.1	0.2	0.3	0.4	0.5
no LM		-	-	0.285	0.266	0.222	0.170	0.106
unigram	($m = 0$)	7.999	8.401	0.195	0.178	0.150	0.118	0.086
bigram	($m = 1$)	3.484	4.204	0.366	0.334	0.277	0.219	0.152
trigram	($m = 2$)	1.176	1.203	0.397	0.357	0.300	0.232	0.152

Table 1. Effect of the language model order on Thumos 14. Language model perplexities are reported on the validation and test set for unigram, bigram, and trigram language models. Results are reported as mAP for different overlap ratios as proposed in [8].

The perplexity for a dataset consisting of multiple sequences is the product of the language model probabilities for each sequence where N is replaced by the total number of segments in the dataset. Intuitively, the perplexity can be seen as the number of possible choices per position. A small perplexity corresponds to a strong language model.

Table 1 shows the results on Thumos. Note that the perplexities on the training data (*i.e*. the validation set) and on the test data are very similar, indicating that the learned language model works well for the action context on the test set. Further, the perplexity decreases with increasing m-gram order, making the language model stronger.

The system with the unigram language model performs clearly worse than the system without a language model. Since 50% of the classes in Thumos are background, the model has a strong bias towards background. Moreover, background usually gets a high classification score for any segment, so the model tends to predict multiple consecutive background segments. This can be prevented by taking context into account. A bigram already leads to a huge gain in performance. Using more context, *e.g*. with a trigram, can still boost the performance, although the gain is not as intense as for the bigram. For the remainder of the paper, we use a trigram language model as it produces the best results.

Model Components. In Table 2, the impact of each component is analyzed. In addition to the results at each of the five overlap ratios, we also report the average length of the detected segments in frames. The detection result of a video from the Thumos 14 test set in Figure 1 serves as an example for the cases discussed in the table.

We start with an analysis of the action model. In Section 3.1.3, we argue that a uniform prior $p(c)$ in Equation (14) works well in practice. If we use a non-uniform class prior (Table 2 (b)), the performance is far below the performance of the uniform prior (Table 2 (a)) and the average segment length is shorter. This is due to the fact that the division by the prior emphasizes infrequent classes. Hence, longer actions are more likely to be split into multiple short segments of rare classes which are then sometimes falsely classified, see Figure 1 (b).

Due to the interplay between the action, length, and language model, the impact of the power factor is a little bit more complex. Without the length and language model, the power factor l_n penalizes long segments, *cf*. (f) and (g) in Table 2. However, when length and language model are included, l_n has another effect: It enhances the action model compared to the language- and length model. So, omitting l_n increases the impact of the length model. Thus, sequences that are longer than μ (see Equation (8)), typically background, are more likely to be split. The language model strongly penalizes consecutive background segments, what explains the short action artifacts in Figure 1(c).

Without length- and language model, long segments which include multiple short actions are classified as background since most of the frames are actually from the background. Thus, performance drops and the average segment length increases, *cf*. (f) in Table 2 and Figure 1.

When using a language model without length model, the performance is still not satisfying and the average segment length is quite large, see (e) in Table 2 and Figure 1. The reason is that each time a new segment is hypothesized, a language model probability is multiplied to the probability score of the system. Thus, there is a clear tendency towards few, long segments in order to avoid language model penalties. Adding a length model compensates for this effect since unreasonably long segments are penalized. Note the interdependence of both models. While the complete system which includes both performs well, the effect of the length model is too strong if the language model is omitted, *cf*. (d) in Table 2. The hypothesized segments are rather short in this case and the performance also drops again. Moreover, false detections occur due to the loss of context information, see Figure 1 (d).

Length Model. We also evaluate our method on MPII-Cooking and 50 Salads in addition to Thumos 14, starting with an investigation of three different kinds of length models. The choice of the length model depends on the dataset and the characteristics of the action classes. A strong length model, such as the class-dependent Poisson model, is superior on 50 Salads and MPII-Cooking but performs worse on Thumos 14, *cf*. Table 4. To analyze this effect, we compare the distribution of the ground truth lengths of each class with the distribution generated by the Poisson model.

We discretize both distributions as a histogram with 20

		avg. length	Overlap				
		ground truth: 212.5	0.1	0.2	0.3	0.4	0.5
(a)	complete system (Eq. (15))	182.1	0.397	0.357	0.300	0.232	0.152
Action Model							
(b)	action model with class prior	108.0	0.310	0.295	0.187	0.113	0.066
(c)	action model w/o length factor	117.3	0.334	0.298	0.237	0.166	0.108
Length- and Language Model							
(d)	w/o LM	147.4	0.285	0.266	0.222	0.170	0.106
(e)	w/o length model	422.8	0.209	0.166	0.128	0.086	0.049
(f)	w/o LM and length model	543.8	0.135	0.111	0.086	0.065	0.041
(g)	w/o LM, length, and length factor	643.1	0.104	0.089	0.068	0.053	0.035

Table 2. Effect of the model components. In the second column, the average length of the detected action segments is given. Evaluation follows the protocol from [8].

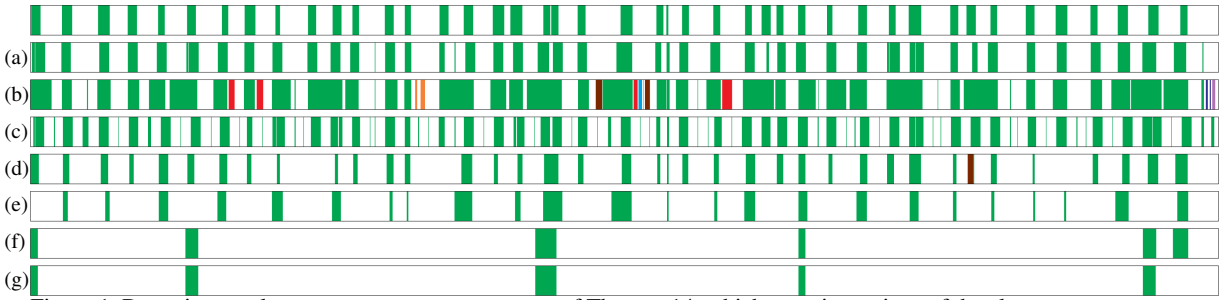

Figure 1. Detection result on `video_test_0001058` of Thumos 14, which contains actions of the class `Hammer Throw`. The first row contains the ground truth, the other rows show the detection results for the systems (a)-(g) from Table 2. Different classes have different colors.

Thumos 14	MPII-Cooking	50 Salads
2.77	1.20	1.31

Table 3. χ^2-distance between the ground truth length distribution and the Poisson model averaged over all classes.

bins with a width of 50 frames each. Then, we compute the χ^2-distance between both distributions and report the mean distance over all classes in Table 3. While the χ^2-distances for MPII-Cooking and 50 Salads are comparably small, the value for Thumos 14 is twice as large, indicating that the Poisson model is a worse representation of the true length distribution on Thumos 14 than on the other datasets.

The mean length model, which only compensates for the bias of the language model towards long segments, performs best on Thumos where the Poisson distribution is a poor model of the underlying distribution as shown in Table 3. Even on 50 Salads and MPII-Cooking where explicit length modeling is superior, the mean length model outperforms the sliding window baseline. We also investigated the decay factor α from Equation (8). For values between 0.5 and 0.9, the results do not change substantially. Only if α is very close to one, the effect of the length model vanishes since long segments are not anymore penalized. In this case, the performance decreases towards the system without length model, *cf*. Table 2 (e).

The class-independent Poisson model is a model in between, not as strong as the class-dependent model, but more explicit than the mean length model. Only on Thumos 14, where the true length distribution is hard to fit, it performs better than the class-dependent model. On the other datasets, the class-dependent model is still superior.

An example detection for each of the three length models on a video from 50 Salads is illustrated in Figure 2. In contrast to the class-dependent model (Figure 2 a), the class-independent Poisson model (Figure 2 b) tends to avoid short segments, particularly for the background class. The mean length model (Figure 2 c) prefers short segments, which results in an over-segmentation of long actions.

A lower bound on the segment lengths is defined by the subsampling of frames. On 50 Salads, the 0.1 overlap mAP slightly decreases from 0.391 to 0.379 and 0.376 for 5, 10, and 20 frame subsampling. For efficiency reasons, we stick to the 10 frame subsampling for all experiments.

Comparison to State-of-the-art. Our method outperforms the sliding window baseline consistently on all datasets, *cf*. Table 4. On Thumos 14, our system achieves 3% higher mAP for overlap 0.1 to 0.4 and is still 1% better for overlap 0.5 compared to the winning submission from INRIA [17]. Their approach is based on a sliding window and a model combination of their system from the classifica-

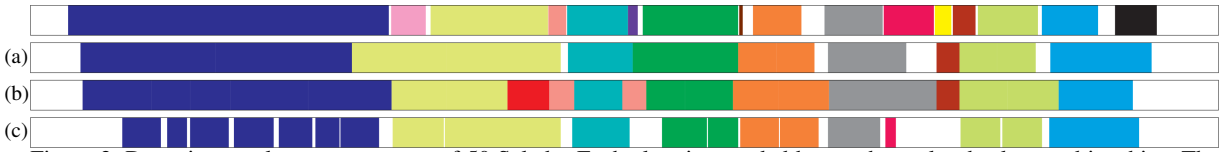

Figure 2. Detection result on `rgb-03-1` of 50 Salads. Each class is encoded by another color, background is white. The first row contains the ground truth, the other lines show the detection results of our system with (a) a class-dependent Poisson model, (b) a class-independent Poisson model, and (c) the mean length model.

	Overlap				
Method	0.1	0.2	0.3	0.4	0.5
Thumos 14					
sliding window	0.325	0.279	0.206	0.150	0.086
Univ. of Florence [9]	0.046	0.034	0.024	0.014	0.009
CHUK & SIAT [29]	0.182	0.170	0.140	0.117	0.083
INRIA (challenge winner) [17]	0.367	0.334	0.270	0.208	0.144
ours w/ mean length model	**0.397**	**0.357**	**0.300**	**0.232**	**0.152**
ours w/ class-independent Poisson model	0.337	0.307	0.255	0.191	0.127
ours w/ class-dependent Poisson model	0.251	0.234	0.201	0.144	0.088
MPII-Cooking					
sliding window	0.222	0.197	0.158	0.126	0.079
ours w/ mean length model	0.220	0.209	0.180	0.135	0.104
ours w/ class-independent Poisson model	0.219	0.200	0.163	0.129	0.098
ours w/ class-dependent Poisson model	**0.248**	**0.239**	**0.220**	**0.192**	**0.140**
50 Salads (low level)					
sliding window	0.201	0.158	0.126	0.100	0.080
ours w/ mean length model	0.305	0.295	0.260	0.208	0.153
ours w/ class-independent Poisson model	0.375	0.357	0.306	0.237	0.149
ours w/ class-dependent Poisson model	**0.379**	**0.368**	**0.352**	**0.312**	**0.229**

Table 4. Comparison of our method to recently published results and the sliding window baseline on the three datasets Thumos 14, MPII-Cooking, and 50 Salads. We use the evaluation protocol proposed for Thumos [8] and report the results as mAP for different overlap ratios.

	Multi-class		per class
Method	prec	recall	mAP
sl. window, holistic [19]	17.7	40.3	44.2
+ pose features [19]	19.8	40.2	45.0
multiple granularity [16]	28.6	48.2	54.3
ours	45.0	25.9	58.3

Table 5. Multi-class precision and recall and single class mAP on MPII-Cooking. We used the evaluation protocol from [19].

tion challenge and a trajectory based classifier trained on the data for the detection task. This has been the best published result on the dataset so far. Our system also outperforms the other challenge submissions [9, 29] which both use a sliding window. Wang *et al*. [29] additionally include CNN features in their classifier. Table 5 shows multi-class precision/recall and single class mAP on MPII-Cooking. The authors of [19] use dense trajectories as features (holistic) and additional pose features. Ni *et al*. [16] use dense trajectory features and detect hand-object interactions. While the existing approaches achieve a high recall at the cost of a comparably low precision, our approach achieves a higher precision at the cost of lower recall. In terms of single class mAP, we are 14% better than [19] and 4% better than [16].

5. Conclusion

In this paper, we proposed a new method for temporal action detection that jointly models the segmentation and recognition of actions. Our approach includes a length and language model in addition to an action classifier. Using dynamic programming, we can efficiently infer the globally optimal action segmentation and classification. We have evaluated our method on three recent datasets and outperformed the state-of-the-art on Thumos 14 and MPII-Cooking by 3% and 4%. Moreover, an analysis of the impact of each model component revealed that the combination of length and language model is crucial for good performance. An investigation of three different length models on each of the three datasets revealed that a strong length model, *e.g*. a class-dependent Poisson model, is beneficial if it represents the true distribution of the action lengths.

Acknowledgements: Authors acknowledge financial support from the DFG Emmy Noether program (GA 1927/1-1) and the donation of a Titan X by NVidia.

References

[1] L. R. Bahl, F. Jelinek, and R. L. Mercer. A maximum likelihood approach to continuous speech recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 5:179–190, 1983. 5

[2] S. Bhattacharya, M. M. Kalayeh, R. Sukthankar, and M. Shah. Recognition of complex events: Exploiting temporal dynamics between underlying concepts. In *IEEE Int. Conf. on Computer Vision and Pattern Recognition*, pages 2243–2250, 2014. 2

[3] Y. Cheng, Q. Fan, S. Pankanti, and A. Choudhary. Temporal sequence modeling for video event detection. In *IEEE Int. Conf. on Computer Vision and Pattern Recognition*, pages 2235–2242, 2014. 2

[4] E. Eyjolfsdottir, S. Branson, X. P. Burgos-Artizzu, E. D. Hoopfer, J. Schor, D. J. Anderson, and P. Perona. Detecting social actions of fruit flies. In *European Conf. on Computer Vision*, pages 772–787, 2014. 2

[5] M. Hoai, Z.-Z. Lan, and F. De la Torre. Joint segmentation and classification of human actions in video. In *IEEE Int. Conf. on Computer Vision and Pattern Recognition*, pages 3265–3272, 2011. 2

[6] M. Jain, J. Van Gemert, H. Jégou, P. Bouthemy, and C. G. Snoek. Action localization with tubelets from motion. In *IEEE Int. Conf. on Computer Vision and Pattern Recognition*, pages 740–747, 2014. 1

[7] M. Jain, J. C. van Gemert, and C. G. Snoek. What do 15,000 object categories tell us about classifying and localizing actions? In *IEEE Int. Conf. on Computer Vision and Pattern Recognition*, pages 46–55, 2015. 1

[8] 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. 1, 5, 6, 7, 8

[9] S. Karaman, L. Seidenari, and A. Del Bimbo. Fast saliency based pooling of Fisher encoded dense trajectories. Technical report, University of Florence, 2014. 1, 8

[10] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei. Large-scale video classification with convolutional neural networks. In *IEEE Int. Conf. on Computer Vision and Pattern Recognition*, pages 1725–1732, 2014. 1, 3

[11] H. Kuehne, A. Arslan, and T. Serre. The language of actions: Recovering the syntax and semantics of goal-directed human activities. In *IEEE Int. Conf. on Computer Vision and Pattern Recognition*, pages 780–787, 2014. 2

[12] H. Kuehne, J. Gall, and T. Serre. An end-to-end generative framework for video segmentation and recognition. In *IEEE Winter Conf. on Applications of Computer Vision*, 2016. 2

[13] T. Lan, Y. Zhu, A. R. Zamir, and S. Savarese. Action recognition by hierarchical mid-level action elements. In *Int. Conf. on Computer Vision*, 2015. 2

[14] P. Mettes, J. C. van Gemert, S. Cappallo, T. Mensink, and C. G. Snoek. Bag-of-fragments: Selecting and encoding video fragments for event detection and recounting. In *ACM Int. Conf. on Multimedia Retrieval*, 2015. 2

[15] H. Ney, U. Essen, and R. Kneser. On structuring probabilistic dependences in stochastic language modelling. *Computer Speech & Language*, 8:1–38, 1994. 3, 5

[16] B. Ni, V. R. Paramathayalan, and P. Moulin. Multiple granularity analysis for fine-grained action detection. In *IEEE Int. Conf. on Computer Vision and Pattern Recognition*, pages 756–763, 2014. 2, 8

[17] D. Oneata, J. Verbeek, and C. Schmid. The LEAR submission at Thumos 2014. Technical report, Inria, 2014. 1, 7, 8

[18] H. Pirsiavash and D. Ramanan. Parsing videos of actions with segmental grammars. In *IEEE Int. Conf. on Computer Vision and Pattern Recognition*, pages 612–619, 2014. 2

[19] M. Rohrbach, S. Amin, M. Andriluka, and B. Schiele. A database for fine grained activity detection of cooking activities. In *IEEE Int. Conf. on Computer Vision and Pattern Recognition*, pages 1194–1201, 2012. 1, 2, 5, 8

[20] G. Sharir and T. Tuytelaars. Action in chains: A chains model for action localization and classification. In *IEEE Winter Conf. on Applications of Computer Vision*, pages 610–617, 2014. 2

[21] Q. Shi, L. Wang, L. Cheng, and A. Smola. Discriminative human action segmentation and recognition using semi-Markov model. In *IEEE Int. Conf. on Computer Vision and Pattern Recognition*, 2008. 2

[22] K. Simonyan and A. Zisserman. Two-stream convolutional networks for action recognition in videos. In *Advances in Neural Information Processing Systems*, pages 568–576, 2014. 1, 3

[23] K. Soomro, A. R. Zamir, and M. Shah. Ucf101: A dataset of 101 human actions classes from videos in the wild. *arXiv preprint arXiv:1212.0402*, 2012. 1, 5

[24] S. Stein and S. J. McKenna. Combining embedded accelerometers with computer vision for recognizing food preparation activities. In *ACM Int. Joint Conf. on Pervasive and Ubiquitous Computing*, pages 729–738, 2013. 2, 5

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

[26] J. C. van Gemert, M. Jain, E. Gati, and C. G. Snoek. APT: Action localization proposals from dense trajectories. In *British Machine Vision Conference*, 2015. 1

[27] N. N. Vo and A. F. Bobick. From stochastic grammar to bayes network: Probabilistic parsing of complex activity. In *IEEE Int. Conf. on Computer Vision and Pattern Recognition*, pages 2641–2648, 2014. 2

[28] H. Wang and C. Schmid. Action recognition with improved trajectories. In *Int. Conf. on Computer Vision*, pages 3551–3558, 2013. 1, 3, 5

[29] L. Wang, Y. Qiao, and X. Tang. Action recognition and detection by combining motion and appearance features. Technical report, The Chinese University of Hong Kong and Shenzhen Institutes of Advanced Technology, 2014. 1, 8

[30] D. Weinland, R. Ronfard, and E. Boyer. A survey of vision-based methods for action representation, segmentation and recognition. *Computer Vision and Image Understanding*, 115:224–241, 2011. 2

[31] S. Wiesler, A. Richard, P. Golik, R. Schluter, and H. Ney. RASR/NN: The RWTH neural network toolkit for speech recognition. In *IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, pages 3281–3285, 2014. 5

[32] J. Yuan, Z. Liu, and Y. Wu. Discriminative video pattern search for efficient action detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33:1728–1743, 2011. 2