Fast Temporal Activity Proposals for Efficient Detection of Human Actions in Untrimmed Videos

Fabian Caba Heilbron[1] fabian.caba@kaust.edu.sa
Juan Carlos Niebles[2,3] jniebles@cs.stanford.edu
Bernard Ghanem[1] bernard.ghanem@kaust.edu.sa

[1]King Abdullah University of Science and Technology (KAUST), Saudi Arabia
[2]Department of Computer Science, Stanford University [3]Universidad del Norte, Colombia
http://www.cabaf.net/temporalproposals

Abstract

In many large-scale video analysis scenarios, one is interested in localizing and recognizing human activities that occur in short temporal intervals within long untrimmed videos. Current approaches for activity detection still struggle to handle large-scale video collections and the task remains relatively unexplored. This is in part due to the computational complexity of current action recognition approaches and the lack of a method that proposes fewer intervals in the video, where activity processing can be focused. In this paper, we introduce a proposal method that aims to recover temporal segments containing actions in untrimmed videos. Building on techniques for learning sparse dictionaries, we introduce a learning framework to represent and retrieve activity proposals. We demonstrate the capabilities of our method in not only producing high quality proposals but also in its efficiency. Finally, we show the positive impact our method has on recognition performance when it is used for action detection, while running at 10FPS.

1. Introduction

With the growth of online and personal media archives, people are generating, storing and consuming very large collections of videos. A need that arises from such large sources of video is the ability to process them for content-based indexing and search. In particular, many such applications would benefit from automatic recognition of events, actions, and activities in continuous video streams. To achieve this, computer vision algorithms are required to temporally localize the activities of interest within long video sequences. Such visual recognition setting corresponds to the well-known task of action/activity detection.

Current methods for action temporal localization rely on applying action classifiers at every time location and at multiple temporal scales, in a temporal sliding window fashion.

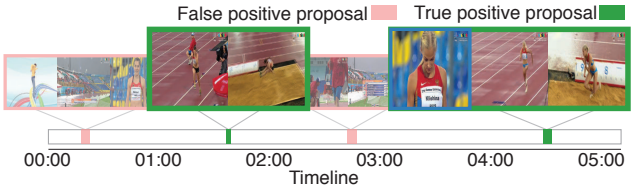

Figure 1. Visualization of temporal action proposals for a sample long video of five minutes. Our method is not only able to retrieve the temporal locations of actions with high recall but also it generates proposals quickly.

However, due to the computational complexity of such classifiers and the large number of possible temporal locations and scales, the sliding window approach is computationally infeasible for large-scale video analysis applications.

In order to avoid this exhaustive evaluation of video classifiers, a number of researchers have recently introduced the idea of spatial or spatiotemporal proposals for the task of action recognition [31, 35, 20, 13, 10]. Within this paradigm, a video is first processed to produce a set of candidate video segments or *proposals*, which are likely to contain a human action or activity as Figure 1 illustrates. These proposals are then used as a reduced candidate set, on which more sophisticated action classifiers can be applied for recognition. A good method for generating proposals should therefore have the following properties: (a) it should recover the true temporal locations of actions with *high recall* and relatively good precision, and (b) it should produce proposals *quickly*.

These two requirements make proposal generation a computationally challenging vision task. In addition, the proposal algorithm should be versatile enough to find candidates for any action or activity class, and simultaneously provide potential starting and ending times for each candidate activity. The large variation in motion, scenes, and objects involved, styles of execution, camera viewpoints, camera motion, background clutter and occlusions impose additional burden to the proposal generation process.

Although the concept of action proposals in video has

been introduced in previous work, most existing methods target spatiotemporal proposals, which are very helpful in localizing and disambiguating actions that occur in the same time. However, as we will see in the experimental section, these methods only achieve marginal recall improvement over simple baselines, such as uniform random sampling, and are too computationally expensive to scale to large datasets like THUMOS [14] or ActivityNet [4]. These two observations motivate our proposed work on localizing proposals only in time, which we expect to be a fundamental building block in the development of scalable and practical action/activity detection algorithms in the future.

Contributions. In this paper, we introduce a new method that produces temporal proposals in untrimmed videos. Our work has the following contributions. First, we propose a sparse learning framework for scoring temporal segments according to how likely they are to contain an action. Second, we experimentally show that current and state-of-the-art proposal methods are not well suited for generating temporal proposals in realistic and large-scale scenarios. Third, we show empirical evidence that our proposed sparse learning framework achieves high recall and efficiency on several benchmark datasets. Finally, we incorporate our proposal generation method into a standard activity detection framework and show that it can significantly boost the overall action detection performance.

2. Related Work

We first describe the importance of using candidate regions from the object domain perspective. Then, we relate our work with previous approaches on action proposals.

For object detection in images, the use of a sliding window (exhaustive search) strategy to localize objects is no longer popular due to the high computational cost that it entails. Instead, generic or class-specific object proposals are used to quickly find possible locations of *an object* in an image. Only these locations are in turn tested by an object classifier to recognize whether or not they contain a specific class of object. Since these proposal methods have very high recall and a low false positive rate, their use in object detection has significantly reduced the runtime of otherwise slow object classifiers [9]. Some popular object proposal methods are SelectiveSearch [30], MCG [2], Objectness [1], and EdgeBoxes [41]. We refer the reader to [12] for an extensive review of the advances in object proposals in the image domain.

Only very recently have proposal methods been extended to the video domain. For example, several works attempt to produce spatiotemporal tubes as proposals for objects in video [22, 7]. However, very limited research has targeted temporal proposals for activities in video; even though a proposal generation method would be crucial for efficient activity detection in long untrimmed videos. Currently, activity detection methods simply apply a computationally expensive activity classifier to a temporally sliding window (or randomly sampled temporal chunks). Very recent work generates spatiotemporal proposals in video, including tubelets [13], action tubes [10], the *actionness* measure [6], proposals from dense trajectories [31], the *fast proposal* method [35], and *Bag of Fragments* [20]. All these methods rely either on dense trajectories or use hierarchical grouping approaches for generating the proposals. This tends to violate efficiency constraints for creating action proposals in a large-scale scenario.

Moreover, previous action proposal methods are not developed or designed to propose temporal segments where a human activity can be confined. In fact, current approaches have not been evaluated beyond simple short videos. So, their scalability and detection performance in real-world scenarios is uncertain. To overcome the existing limitations, our proposed algorithm generates temporal activity candidates using an efficient scoring function that can be adapted to varying activity types of interest.

3. Temporal Activity Proposals

To be applicable at large-scales and in practical scenarios, a useful activity proposal method is driven by two competing goals. **(i)** The proposal method *must be* computationally efficient, in representing, encoding, and scoring a temporal segment. **(ii)** The proposal method *must be* discriminative of activities that we are interested in, so as to only retrieve temporal segments that contain visual information indicative of these activity classes. On one hand, sampling a long video uniformly without using any content-based information can generate proposals very quickly; however, more often than not, this strategy will retrieve segments that are not related to the activity classes we seek. On the other hand, executing the most successful activity recognition methods in the literature will not be feasible. Although extracting dense trajectories and encoding them using Fisher vectors have become the defacto standard in the majority of state-of-the-art methods for trimmed activity recognition [14, 24, 5, 32, 3], this strategy is too slow for extracting activity proposals, especially at large-scale. In this work, we propose a temporal proposal strategy that is a successful and, more importantly, tunable tradeoff between these two goals. In what follows, we give a detailed account of our proposed method and emphasize how each goal is mindfully considered in its design.

Fig 2 shows an overview of our approach for generating temporal activity proposals. Given a set of training videos, we extract features that captures spatial and temporal appearance. We next learn a universal dictionary that encodes discriminative information for a set of activity classes. After constructing this dictionary, we can use it to efficiently

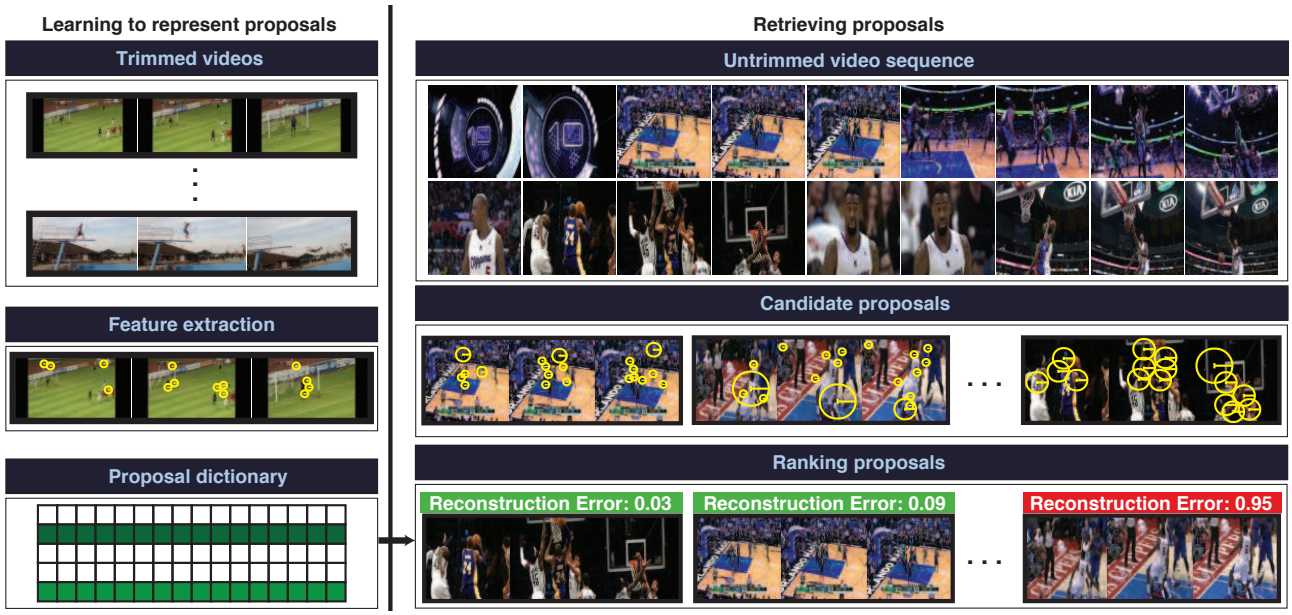

Figure 2. At training time, our approach describes trimmed action instances using STIPs and learns a dictionary that encodes their visual contents. Given a test sequence, we first generate a large set of candidate temporal segments. These candidates are described using STIPs and ranked using the learned dictionary. Finally, a subset of the candidate segments set is selected as our activity proposals based on the predicted activity proposal ranking.

encode temporal segments in unseen video sequences. To do this, we first generate a large set of candidates temporal segments. To output the final temporal activity proposals, these segments are efficiently ranked according to how well they are represented by the dictionary and only the most representative among them are retrieved.

3.1. Candidate Proposals

Our method starts with an initial set of candidates, from which we eventually select and retrieve proposals. This candidate proposal set is usually much larger in cardinality than the generated proposals. In what follows, we describe how these candidate proposals are extracted and represented.

Generating candidate proposals: Within an input video sequence, we sample temporal segments of different lengths from the entire video. This sampling is done uniformly over time, but we sample likely proposal lengths from a distribution compiled from a set of training videos containing temporally localized activity classes. In doing so, we partition the input video into a large pre-defined number of proposal candidates, which can overlap in time.

Feature extraction: As a tradeoff between computational efficiency and representation/discrimination power, we decide to use Spatio Temporal Interest Points (STIPs) [17], which are extracted in the aforementioned proposal candidates. We follow the standard practice and encode each STIP point using Histogram of Oriented Gradients (HOG) and Histogram of Optical Flow (HOF) to characterize its spatial and temporal appearance. Note that STIPs have been successfully applied in previous work to action recognition [27] and video summarization [40]. In practice, we further speedup this extraction and representation process by performing it in parallel across the entire set of candidates. As such, each proposal candidate is represented as a set of feature descriptors $\mathbf{x}_i \in \mathbb{R}^{172}$, which can be concatenated for convenience in matrix form as $\mathbf{X}_k = [\mathbf{x}_1|\cdots|\mathbf{x}_{n_k}]$, where n_k is the total number of STIPs detected in the k^{th} proposal candidate.

3.2. Learning to Represent Proposals

Inspired by seminal works in image classification [34] and action recognition [11], we assume that each STIP feature in a proposal candidate $\mathbf{X}_k$ can be represented linearly using a sparse set of dictionary/vocabulary elements, which are learned offline from a large video corpus with temporally annotated activity instances. Moreover, we expect that the representations of different STIP features within the same proposal candidate, which originate from a particular activity class, would not be independent, but instead share commonalities. In fact, the merits of representing interest points jointly instead of independently are well-studied in the image classification literature [8, 28]. To combine all these requirements together, two design choices emerge, both of which we will explore. **(a)** *class-independent proposal learning*: We seek to compute an over-complete dictionary that can jointly represent STIP features within the same proposal candidate using a sparse set of dictionary elements. Here, the focus is solely on representation and is agnostic to any supervised information that is possibly available. **(b)** *class-induced proposal learning*: We seek a

dictionary similar to the one in **(a)**, but that also leads to a proposal representation that is discriminative of the activity classes we are interested in. Both choices are viable and each has its own merits [19, 21]. However, for the purpose of retrieving meaningful activity proposals to be used in activity detection, we will experimentally show in Section 4 that **(b)** is superior to **(a)**. Previous work in image classification has also reached this conclusion [15, 16].

3.2.1 Class-Independent Proposal Learning

From the training set, we compile all temporal segments $\mathbf{X}_l$ that belong to annotated activities to form the data matrix $\mathbf{X} = [\mathbf{X}_1|\cdots|\mathbf{X}_l]$, where l corresponds to the total number of trimmed instances in the training set. Then, we cast the dictionary learning problem of $\mathbf{D}_U$ as follows.

$$(\mathbf{D}_U, \mathbf{A}^*) = \arg\min_{\mathbf{D},\mathbf{A}} \frac{1}{n}\|\mathbf{X} - \mathbf{D}\mathbf{A}\|_F^2 + \lambda\|\mathbf{A}\|_{2,1}, \quad (1)$$

where $\mathbf{X} \in \mathbb{R}^{172\times n}$, $\mathbf{D} \in \mathbb{R}^{172\times d}$, $\mathbf{A} \in \mathbb{R}^{d\times n}$ with d equal to the number of dictionary elements, and n equal to the total number of STIP points in the training subset. Note that, $\mathbf{A} = [\mathbf{A}_1|\cdots|\mathbf{A}_k]$ is a stacked matrix that contains the reconstruction coefficients for all STIPs in all the activity instances. Inspired by the results of [40] on video summarization, we use a ℓ_1/ℓ_2 matrix regularizer to encourage joint sparsity in the representations of each activity instance. In fact, this regularization scheme encourages that the STIPs in each temporal segment $\mathbf{X}_k$ share the same sparse support in representation, i.e. they use similar dictionary elements for reconstruction. We control the reconstruction quality with the tradeoff parameter λ which we set in practice to $\lambda = 0.05$. This joint representation scheme is a form of multi-task learning (MTL), where tasks that share dependencies in features or learning parameters are jointly solved in order to capitalize on their inherent relationships. In our case, coding each individual STIP is considered a single task. Note that this type of learning has been successfully applied to classical problems (e.g. image annotation [25], image classification [37], and object tracking [39, 38]) and has outperformed state-of-the-art methods that resort to independent learning.

To solve Eq (1), we follow a conventional strategy of fixed point optimization, which iteratively updates each of the variables $\mathbf{D}_U$ and $\mathbf{A}$ separately by fixing one of them at a time. So at every iteration, two update steps are required and we summarize them next. We initialize $\mathbf{D}_U$ using K-Means. The iterative method is terminated when the relative change in objective is smaller than a pre-defined threshold.

Updating A: This is referred to as the coding step. It requires the solution to Eq 2, which is a non-smooth convex program that can be optimized using the Alternating Direction Method of Multipliers (ADMM).

$$\min_{\mathbf{A}} \frac{1}{n}\|\mathbf{X} - \mathbf{D}\mathbf{A}\|_F^2 + \lambda\|\mathbf{A}\|_{2,1} \quad (2)$$

Updating $\mathbf{D}_U$: This update requires the solution to Eq (3), which is a linear least squares problem in matrix form. It can be optimized by solving a set of linear systems.

$$\min_{\mathbf{D}} \|\mathbf{X} - \mathbf{D}\mathbf{A}\|_F^2 \quad (3)$$

3.2.2 Class-Induced Proposal Learning

The second design choice for the dictionary learning is to incorporate supervised information (i.e. activity class labels) into the learning process. Here, we describe how we learn a universal dictionary in a supervised fashion using the available class labels in the training set. We formulate the problem in Eq (4).

$$\begin{aligned}(\mathbf{D}_S, \mathbf{A}^*, \mathbf{W}^*) = \arg\min_{\mathbf{D},\mathbf{A},\mathbf{W}} \frac{1}{n}\|\mathbf{X} - \mathbf{D}\mathbf{A}\|_F^2 + \lambda_1\|\mathbf{A}\|_{2,1} \\ + \lambda_2\|\mathbf{W}^T\mathbf{A} - \mathbf{Y}\|_F^2 + \lambda_3\|\mathbf{W}\|_F^2, \quad (4)\end{aligned}$$

where $\mathbf{W} \in \mathbb{R}^{d\times c}$, and $\mathbf{Y} \in \{0,1\}^{c\times n}$ with c equal to the number of classes. Here, $\mathbf{Y}$ is the label matrix for all the STIPs in the training set, whereby each STIP inherits the label of the activity instance it belongs to. The matrix $\mathbf{W}$ is a column-wise concatenation of c one-vs-all linear classifiers. The main difference between this formulation and Eq (1) is the training classification loss term, $\|\mathbf{W}^T\mathbf{A} - \mathbf{Y}\|_F^2$, which empowers the dictionary $\mathbf{D}_S$ with discriminative properties. Of course, different forms of this loss term can be used, including the hinge loss used in SVMs or the logistic loss used in logistic regression. We choose this simple ℓ_2 loss for computational reasons. To be less sensitive to overfitting and for better classifier generalization, we add an energy regularizer on $\mathbf{W}$. Similar to class-independent proposal learning, we use alternating optimization to solve Eq (4) and the same initialization for $\mathbf{D}_S$.

Updating W: This requires solving Eq (5), which is a linear least squares problem. The classifiers learned in this step will not be used for activity recognition later. They are merely intermediate variables that guide the dictionary to being more discriminative.

$$\min_{\mathbf{W}} \|\mathbf{W}^T\mathbf{A} - \mathbf{Y}\|_F^2 + \lambda_2\|\mathbf{W}\|_F^2 \quad (5)$$

Updating A: This coding step requires solving Eq (6), which can be solved using ADMM.

$$\min_{\mathbf{A}} \frac{1}{n}\|\mathbf{X} - \mathbf{D}\mathbf{A}\|_F^2 + \lambda_1\|\mathbf{A}\|_{2,1} + \lambda_2\|\mathbf{W}^T\mathbf{A} - \mathbf{Y}\|_F^2 \quad (6)$$

Updating D: This step is the same as Eq (3).

3.3. Retrieving Proposals

Once we have learned a dictionary-based proposal representation, our aim is to retrieve activity proposals from unseen input videos which contain a human activity that is similar to activity instances in the training set. As described in Section 3.1, this input video is segmented into a large number of proposal candidates. Then, we jointly encode each proposal candidate by solving Eq (7) using the learned dictionary $\mathbf{D}$ (either $\mathbf{D}_U$ or $\mathbf{D}_S$). Since this procedure is done online, we achieve further speedup by using the sparse codes $\mathbf{A}_k$ of one proposal candidate $\mathbf{X}_k$ as an initialization to the sparse codes $\mathbf{A}_j$ of another proposal candidate $\mathbf{X}_j$ that overlaps with it. This is valid, since Eq (7) is convex and it is guaranteed to converge to the global solution no matter what the initialization. Finally, the average reconstruction error $\frac{1}{n_k}\|\mathbf{X}_k - \mathbf{D}\mathbf{A}_k\|_F^2$ is used to rank the proposal candidates. A small error value indicates that the candidate $\mathbf{X}_k$ can be represented well with the learned dictionary and thus it is likely to belong to one of the activity classes belonging to the training set. Obviously, we retrieve the final activity proposals as the candidates with lowest reconstruction error.

$$\mathbf{A}_k^* = \arg\min_{\mathbf{A}_k} \frac{1}{n_k}\|\mathbf{X}_k - \mathbf{D}\mathbf{A}_k\|_F^2 + \lambda\|\mathbf{A}_k\|_{2,1}. \qquad (7)$$

4. Experimental Results

In this section, we test our proposal generation method under two different settings. We generate and rank proposals using a dictionary learned in an **class-independent** (Section 3.2.1) fashion, as well as, using a dictionary learned in a **class-induced** setting (Section 3.2.2).

We evaluate the performance of our temporal activity proposal method from three perspectives. First, we study the quality of our activity proposals by measuring the ability of our method to retrieve proposals that overlap with the occurrence of actions in long videos in Section 4.2. Second, we evaluate the processing speed of our method to assess how quickly it can generate activity proposals in Section 4.3. Finally, we apply our proposal generation method to the task of action detection in Section 4.4, and show how our method can contribute to improving the performance of existing methods in the action detection task. Throughout the experiments, we compare our method to baseline and state-of-the-art proposal methods. Our results show that our framework consistently achieves improved performance over the state-of-the-art.

4.1. Experimental Setup

Datasets: We test our proposal method on two different datasets for action temporal localization. First, we conduct experiments on the MSR-II Action dataset [36]. It contains 54 video sequences with an average length of 51 seconds each. These videos depict three action classes: Boxing, Clap, and Wave. Following the standard evaluation protocol, we use videos in the KTH [27] dataset for training. Second, we test the quality of our method on the labeled untrimmed videos from the challenging THUMOS 2014 Detection Challenge dataset [14]. This dataset compiles videos from YouTube from 20 sport actions, and is considered one of the most challenging datasets for action detection. Videos in this dataset have an average length of 3 minutes with the actions usually confined to a small fraction of the video. We use 200 untrimmed videos from the validation subset to train our proposal method. For testing, we evaluate on the remaining 213 videos that are provided with temporal activity annotations.

Baselines: We compare our methods to two baselines and three state-of-the-art techniques. (1) Uniform sampling: it ranks each candidate proposal uniformly at random. (2) Binary Proposal Classifier (BPC): from the raw video descriptors, a binary linear classifier is learned to discriminate between *action vs non-action*. We use STIPs from the trimmed video as positive instances and STIPs from background segments as negative instances. To rank proposals, we perform mean pooling over all the classifier scores within each candidate proposal. The same candidate proposal generation approach described in Section 3.1 is used to feed (1) and (2). (3) Action localization Proposals from dense Trajectories (APT) [31]; (4) Fast Action Proposals for Human action Detection and Search (FAP) [35]; and (5) Bag of Fragments (BoFrag) [20]. Note that APT and FAP are originally designed to generate spatiotemporal proposals, so we project their resulting proposals to the temporal dimension only and discard the spatial information. This process may result in highly overlapping or duplicate temporal proposals, so we remove these for a more fair comparison against our method. While these methods target spatiotemporal proposals and not directly compute temporal-only proposals, we choose to compare to them as they are the closest methods in the literature that are applicable to our problem. We obtain the pre-computed proposals for APT and FAP on MSR-II directly from the authors. On Thumos14, we run an implementation of APT available online to extract proposals, we obtain pre-computed proposals for BoFrag directly from the authors, but we were not able to obtain an implementation for FAP.

Matching criterion: In order to evaluate the quality of proposals generated by each method, we measure the overlap between each proposal and the ground truth temporal annotations. To do this, we compute the *temporal Intersection over Union (tIoU)* as the intersection over union of the two time intervals. If the tIoU of a proposal is above a predefined threshold, the proposal is considered a true positive.

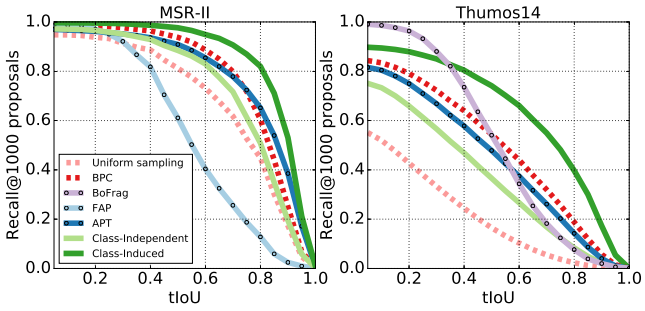

Figure 3. Recall rate at different tIoU thresholds.

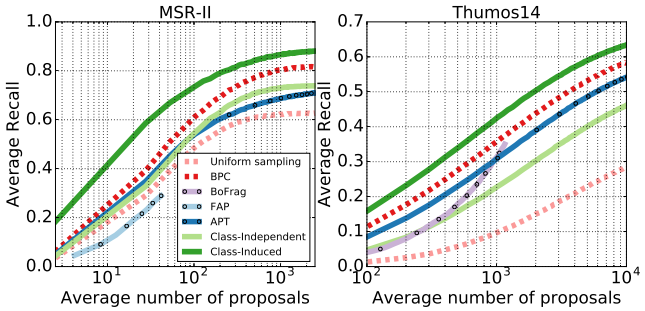

Figure 4. Average recall rate against average selected number of proposals. The recall is averaged over multiple tIoU thresholds from 0.5 to 1.

4.2. Recall Analysis

As discussed earlier, a good action proposal method should retrieve as many true activity segments in a video as possible, i.e. it should achieve a high recall rate. We analyze the quality of our temporal proposals from two perspectives. First, we study how tight our proposals are in comparison to the ground truth temporal locations. Second, we evaluate the quality of our proposal ranking by measuring recall on the top-ranked proposals generated by our method. All throughout, we report the performance of the baselines and state-of-the-art methods for comparison.

4.2.1 Proposal Localization Quality

In this experiment, we obtain a fixed number of proposals from each method and measure recall by comparing the retrieved proposals to the ground truth at various tIoU thresholds. We only allow one detection per ground truth instance.

Figure 3 plots the recall of each method against the tIoU threshold. Consider for instance the results on the MSR-II dataset in Figure 3 (left). For a tIoU threshold of 0.8, we see that our class-induced method achieves a recall of 80%, while the second-best is APT with a recall of 65%. We observe the same behaviour across the entire range of tIoU thresholds for both datasets, which indicates that our proposal generation method achieves the highest action localization quality.

As expected, the proposals scored with the class-independent approach achieve lower recall rates than the ones scored in the class-induced fashion. We attribute the resulting gap in the curves to the fact that our class-induced approach is empowered with discriminative properties. Moreover, our approach clearly outperforms the Uniform sampling baseline. For example, when tIoU is fixed to 0.8, our class-induced method achieves improvements in recall of 40% and 38% on MSR-II and Thumos14 respectively with respect to that baseline. We note that the BPC baseline obtains good recall at low tIoU thresholds; however, this behavior is not consistent at high tIoU thresholds.

From Figure 3 (left), it is clear that the Fast Proposals (FAP) approach is not well suited for proposing temporal segments at high tIoU thresholds. When the tIoU threshold is greater than 0.4 the recall rate decreases dramatically. For example, its recall rate is 0.6 at a tIoU of 0.5, which is lower than what we obtained by uniform sampling.

As compared to APT, the quality of our class-induced proposals is clearly better, since they achieve significant recall improvement at higher tIoU thresholds. As reference, our method obtains a 30% improvement in recall over APT when tIoU is fixed to 0.5 in Thumos14. Interestingly, BoFrag achieves a high recall at lower tIoU thresholds. However, our class-induced proposals outperform BoFrag by a large margin for tIoU thresholds greater than 0.4.

4.2.2 Proposal Ranking Quality

In this experiment, we study the performance of our method in terms of recall when only a limited number of proposals is retrieved. To do this, we select a set number of top-ranked proposals generated by each method and measure the average recall between tIoU 0.5 to 1. As such, the average recall measure summarizes proposal performance across tIoU thresholds and it correlates with detection performance as shown in [12]. Notice that APT, BoFrag and FAP produce a fixed not scored number of proposals; therefore, we randomly select the number of retrieved proposals for these two methods. Results of this experiment are depicted in the recall curves in Figure 4. In these plots, we gradually increase the number of retrieved proposals and record the recall rate of each method. For example, in Figure 4 (left), when retrieving only the top-50 proposals per video, our class-induced method reports a recall of 0.7 on MSR-II outperforming all other methods. This shows that our method ranks proposals better than competing methods, which results in higher recall when a small number of proposals is retrieved. Such behavior is important to guarantee an acceptable recall rate for time-sensitive detection tasks.

We also note that our class-induced proposal method performs better than the class-independent method. In this case, the class-induced proposals reach a high recall faster. Compared to the BPC baseline, we are able to obtain a

	Time [seconds]			Speedup	FPS
	Feature	Proposal	Total		
BPC	191.1	307.3	498.4	15.9	10.8
APT	2828.5	5120.3	7948.8	1.0	0.68
Ours	191.1	342.5	533.6	**14.9**	**10.2**

Table 1. Time comparison between our class-induced method and other approaches. Reported times correspond to the average processing time for a single 3-minute video from the Thumos14 dataset, which contains a total of 141 hours of video.

higher recall rate, no matter how many proposals are retrieved. As shown in Figure 4 (right), our method can achieve a recall of 0.5 with only 1000 proposals, as compared to 0.35 obtained by the BPC.

In comparison to state-of-the-art approaches, our class-induced method produces temporal proposals with higher specificity. Similar to the localization quality analysis, FAP shows low recall performance at a small number of proposals. Notice also that FAP tends to generate a relatively small number of proposals. For example, on MSR-II, it generates an average of $\approx$ 50 proposals. Additionally, we observe than BoFrag produces a modest performance when using a small number of proposals which is an indication of a poor ranking quality. On the other hand, APT shows an acceptable performance when using a small number of proposals. However, our method obtains higher recall with much smaller number of proposals.

4.3. Efficiency Analysis

In this experiment, we measure the processing speed of our method and compare it against competing approaches on the Thumos14 dataset. As a reminder, proposal generation methods are used to reduce the computational burden of applying expensive action classifiers exhaustively in a sliding window fashion. Therefore, it is important for these methods to process video segments efficiently.

Table 1 summarizes the running time of our proposed method in comparison with competing approaches. The reported time is the average time needed to generate the maximum number of proposals from a single video in the Thumos14 dataset. Note that the average length of this video is 180 seconds. Our method achieves the best recall performance while keeping an attractive computational time. In fact, when comparing our method against APT, we are able to generate the same number of proposals 15 times faster while obtaining higher recall. Our method significantly speeds up the required time for generating proposals, while leveraging visual features that can be computed faster.

4.4. Application to Action Detection

The end goal of our temporal proposals is to improve the detection of human activities in long, untrimmed video sequences. To analyze the merits of our method towards this goal, we incorporate our method into an action detection pipeline as follows. First, we train action classifiers using the trimmed action instances available in the training set of each dataset. At the testing stage, we process input video to generate temporal proposals with each method. We then apply the trained action classifiers to each temporal proposal. We evaluate the detection performance of this pipeline by measuring the mean Average Precision (mAP). We run the same pipeline with each proposal generation method to compare the final action detection performance.

For MSR-II, we first extract improved dense trajectories [32]. Then, we encode the obtained descriptors using Fisher vectors [26]. As in [31], we learn a Gaussian Mixture Model with 128 components. We also use PCA to reduce the descriptors dimensionality to the half. For training, we use a *one-vs-all* linear SVM classifier. As for the large-scale experiment in Thumos14, we use the features provided in the action detection challenge [14]. Next, we learn a χ^2 kernel SVM within a Multi-Channel approach as in [18].

In Table 2, we summarize the mAP detection results obtained by the various methods on both datasets. As stated earlier, we randomly retrieve proposals for APT and FAP from the initial subset of proposals they generate. This is because their lack of scored proposals. We demonstrate that our class-induced proposals provide a significant benefit to the action detection pipeline. In both datasets, our proposals obtain better performance compared to more computationally demanding approaches such as APT. Additionally, the results show that our method is able to obtain an acceptable mAP with only a small number of proposals.

Table 3 compares our temporal action detection results to the state-of-the-art on the Thumos14 detection challenge [23, 33, 29]. Our Class-Induced proposals achieve a 13.5% mAP score at 0.5 overlap threshold, as compared to 14.3% obtained by the top performer in Thumos14 [23]. Although the computation time of the latter method is not available for direct comparison, we estimate it to be higher than APT, as it uses a sliding window approach. Therefore, this result is encouraging considering that our method scans less temporal windows and provides a faster detection pipeline.

Method	Sun *et al.* [29]	Wang *et al.* [33]	Oneata *et al.* [23]	**Ours**
mAP	4.4%	8.3%	14.3%	**13.5%**

Table 3. Comparison to the state-of-the-art on the Thumos14 detection challenge. We report the mean Average Precision (mAP) at 0.5 threshold. Our Class-Induced proposal method achieves a competitive performance while keeping an attractive computational complexity.

4.5. Qualitative Results

In Figure 5, we present a qualitative analysis of proposals generated by our class-induced method. We show the **Top-5** highest ranked and the **Bottom-5** worst ranked proposals.

Notice the ability of our method to highly score proposals that are related with a previously seen action. For ex-

	Baselines						Previous Work						Our Method					
	Uniform Sampling			BPC			FAP [35]			APT [31]			Class-Independent			Class-Induced		
tIoU	0.125	0.5	0.8	0.125	0.5	0.8	0.125	0.5	0.8	0.125	0.5	0.8	0.125	0.5	0.8	0.125	0.5	0.8
# Proposals	MSR-II																	
10	45.3	29.7	11.5	65.8	46.3	18.9	68.9	27.1	12.1	66.5	45.1	22.1	56.7	32.3	18.3	72.9	55.4	26.3
50	50.1	31.3	13.2	73.1	49.9	21.1	71.3	28.2	12.3	72.7	49.8	28.7	58.3	33.7	18.9	76.1	57.7	29.1
100	52.2	30.9	9.7	75.1	55.2	21.3	–	–	–	74.1	54.5	33.7	58.6	34.1	19.2	**80.1**	**60.3**	**33.9**
	Thumos14																	
10	9.1	2.4	0.9	22.7	6.2	3.5	–	–	–	22.2	5.8	3.2	15.4	3.5	1.7	25.7	9.5	4.1
100	12.1	3.2	1.3	29.7	8.7	4.3	–	–	–	27.1	7.9	4.1	19.1	5.9	2.9	33.5	12.1	6.9
1000	19.1	3.1	1.9	32.1	9.9	5.1	–	–	–	30.7	9.1	4.8	21.8	6.7	3.3	**35.7**	**13.5**	**7.5**

Table 2. Detection performance (mAP) for different approaches and datasets. Given the lack of available implementation, results for FAP in Thumos14 are not reported.

Figure 5. **Left:** Top-5 best ranked proposals from entire Thumos14 testing set. **Right:** Bottom-5 worst ranked proposals from entire Thumos14 testing set.

Figure 6. Two illustrative examples where our proposal method correctly covers the ground truth, and one example where it fails.

ample, all the five best ranked proposals are related with one of the 20 classes on Thumos14. As illustrated by the figure, our proposal method is able to tightly localize the actions. Additionally, our method ranks unseen actions with low scores, as exemplified in the proposal that contains frames from a penalty kick foul. Interestingly, we find an incomplete high jump action ranked in the bottom. This is evidence that our proposal method is able to discard low quality proposals.

In general, as showed in Section 4.2, our proposal method is able not only to retrieve proposals with good localization but also to rank them with high score. In Figure 6, we show two illustrative examples where the Top-3 best ranked proposals correctly match the ground truth, and one example where it fails to retrieve the actions.

5. Conclusions

We introduced a framework that generates temporal action proposals on untrimmed videos. We demonstrated that our method is able to generate high quality proposals in term of: *localization* and *ranking*. From the efficiency point of view, our proposal method was able to generate proposals 15 times faster than previous approaches, as it runs at 10FPS. We also showed that our proposals can serve an important role in an end-to-end action detection pipeline by improving its overall performance on a large-scale benchmark. For future work, we are interested in further improving the efficiency of our proposal method by interleaving or combining the feature extraction and proposal representation stages, which are currently done independently.

Acknowledgments: Research in this publication was supported by the King Abdullah University of Science and Technology (KAUST) Office of Sponsored Research, the Stanford AI Lab-Toyota Center for Artificial Intelligence Research, and a Google Faculty Research Award (2015).

References

[1] B. Alexe, T. Deselaers, and V. Ferrari. Measuring the objectness of image windows. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(11):2189–2202, 2012.

[2] P. Arbelaez, J. Pont-Tuset, J. Barron, F. Marques, and J. Malik. Multiscale combinatorial grouping. In *CVPR*, 2014.

[3] I. Atmosukarto, B. Ghanem, and N. Ahuja. Trajectory-based fisher kernel representation for action recognition in videos. In *ICPR*, 2012.

[4] F. Caba Heilbron, V. Escorcia, B. Ghanem, and J. Carlos Niebles. Activitynet: A large-scale video benchmark for human activity understanding. In *CVPR*, 2015.

[5] F. Caba Heilbron, A. Thabet, J. C. Niebles, and B. Ghanem. Camera motion and surrounding scene appearance as context for action recognition. In *ACCV*. 2014.

[6] W. Chen, C. Xiong, R. Xu, and J. J. Corso. Actionness ranking with lattice conditional ordinal random fields. In *CVPR*, 2014.

[7] K. Fragkiadaki, P. Arbelaez, P. Felsen, and J. Malik. Learning to segment moving objects in videos. In *CVPR*, 2015.

[8] S. Gao, I. Tsang, L.-T. Chia, and P. Zhao. Local features are not lonely - laplacian sparse coding for image classification. In *CVPR*, 2010.

[9] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*, 2014.

[10] G. Gkioxari and J. Malik. Finding action tubes. In *CVPR*, 2015.

[11] T. Guha and R. Ward. Learning sparse representations for human action recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(8):1576–1588, 2012.

[12] J. Hosang, R. Benenson, P. Dollár, and B. Schiele. What makes for effective detection proposals? *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(4):814–830, 2016.

[13] M. Jain, J. v. Gemert, H. Jégou, P. Bouthemy, and C. G. Snoek. Action localization with tubelets from motion. In *CVPR*, 2014.

[14] 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.

[15] M. Jiu, C. Wolf, C. Garcia, and A. Baskurt. Supervised learning and codebook optimization for bag-of-words models. *Cognitive Computation*, 4(4):409–419, 2012.

[16] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*. 2012.

[17] I. Laptev. On space-time interest points. *International Journal of Computer Vision*, 64(2-3):107–123, 2005.

[18] I. Laptev, M. Marszalek, C. Schmid, and B. Rozenfeld. Learning realistic human actions from movies. In *CVPR*, 2008.

[19] J. Mairal, J. Ponce, G. Sapiro, A. Zisserman, and F. R. Bach. Supervised dictionary learning. In *NIPS*. 2009.

[20] 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 *Proceedings of the 5th ACM on International Conference on Multimedia Retrieval*, pages 427–434. ACM, 2015.

[21] J. Niebles, H. Wang, and L. Fei-Fei. Unsupervised learning of human action categories using spatial-temporal words. *International Journal of Computer Vision*, 79(3):299–318, 2008.

[22] D. Oneata, J. Revaud, J. Verbeek, and C. Schmid. Spatio-temporal object detection proposals. In *ECCV*. 2014.

[23] D. Oneata, J. Verbeek, and C. Schmid. The LEAR submission at thumos 2014. *THUMOS14 Action Recognition Challenge*, 2014.

[24] X. Peng, L. Wang, X. Wang, and Y. Qiao. Bag of visual words and fusion methods for action recognition: Comprehensive study and good practice. *arXiv preprint arXiv:1405.4506*, 2014.

[25] A. Quattoni, X. Carreras, M. Collins, and T. Darrell. An efficient projection for $l_{1,\infty}$ regularization. In *ICML*, 2009.

[26] J. Sánchez, F. Perronnin, T. Mensink, and J. Verbeek. Image classification with the fisher vector: Theory and practice. *International Journal of Computer Vision*, 105(3):222–245, 2013.

[27] C. Schuldt, I. Laptev, and B. Caputo. Recognizing human actions: a local svm approach. In *ICPR*, 2004.

[28] A. Shabou and H. LeBorgne. Locality-constrained and spatially regularized coding for scene categorization. In *CVPR*, 2012.

[29] C. Sun, S. Shetty, R. Sukthankar, and R. Nevatia. Temporal localization of fine-grained actions in videos by domain transfer from web images. In *Proceedings of the 23rd Annual ACM Conference on Multimedia Conference*, pages 371–380. ACM, 2015.

[30] J. R. Uijlings, K. E. van de Sande, T. Gevers, and A. W. Smeulders. Selective search for object recognition. *International journal of computer vision*, 104(2):154–171, 2013.

[31] J. van Gemert, M. Jain, E. Gati, and C. G. M. Snoek. APT: Action localization proposals from dense trajectories. In *BMVC*, 2015.

[32] H. Wang and C. Schmid. Action recognition with improved trajectories. In *ICCV*, 2013.

[33] L. Wang, Y. Qiao, and X. Tang. Action recognition and detection by combining motion and appearance features. *THUMOS14 Action Recognition Challenge*, 2014.

[34] J. Yang, K. Yu, Y. Gong, and T. Huang. Linear spatial pyramid matching using sparse coding for image classification. In *CVPR*, 2009.

[35] G. Yu and J. Yuan. Fast action proposals for human action detection and search. In *CVPR*, 2015.

[36] J. Yuan, Z. Liu, and Y. Wu. Discriminative subvolume search for efficient action detection. In *CVPR*, 2009.

[37] X. Yuan and S. Yan. Visual classification with multi-task joint sparse representation. In *CVPR*, 2010.

[38] T. Zhang, B. Ghanem, S. Liu, and N. Ahuja. Robust visual tracking via structured multi-task sparse learning. *International Journal of Computer Vision*, 101(2):367–383, 2013.

[39] T. Zhang, S. Liu, C. Xu, S. Yan, B. Ghanem, N. Ahuja, and M.-H. Yang. Structural Sparse Tracking. In *CVPR*, 2015.

[40] B. Zhao and E. Xing. Quasi real-time summarization for consumer videos. In *CVPR*, 2014.

[41] C. L. Zitnick and P. Dollár. Edge boxes: Locating object proposals from edges. In *ECCV*. 2014.