Action Search: Learning to Search for Human Activities in Untrimmed Videos

Humam Alwassel*, Fabian Caba Heilbron*, and Bernard Ghanem
King Abdullah University of Science and Technology (KAUST), Thuwal, Saudi Arabia
{humam.alwassel, fabian.caba, bernard.ghanem}@kaust.edu.sa

Abstract

Traditional approaches for action detection use trimmed data to learn sophisticated action detector models. Although these methods have achieved great success at detecting human actions, we argue that huge information is discarded when ignoring the process, through which this trimmed data is obtained. In this paper, we propose Action Search, a novel approach that mimics the way people annotate activities in video sequences. Using a Recurrent Neural Network, Action Search can efficiently explore a video and determine the time boundaries during which an action occurs. Experiments on the THUMOS14 dataset reveal that our model is not only able to explore the video efficiently but also accurately find human activities, outperforming state-of-the-art methods.

1. Introduction

We have recently seen an exponential growth in the number of videos uploaded online. For example, more than 300 hours of video are uploaded to YouTube every minute. Many applications, such as video retrieval, video surveillance, and patient monitoring, require temporal action localization in long videos. Thus, it is crucial today with this high volume of video to develop approaches that efficiently process a video and accurately detect the human activities appearing in this video. This problem is widely known in the computer vision community as temporal action localization.

Traditional approaches that address the problem of temporal action localization rely on trimmed data to learn sophisticated models [5, 8, 18, 26, 28]. These methods have achieved great success at detecting human actions. However, despite the encouraging progress, the goal of accurate detection remains elusive to automated systems. One of the main drawback of current detection methods is that they discard the sequence of steps human annotators follow to produce the final video annotations, which are inherently the only information from the annotation process used to train the detection models.

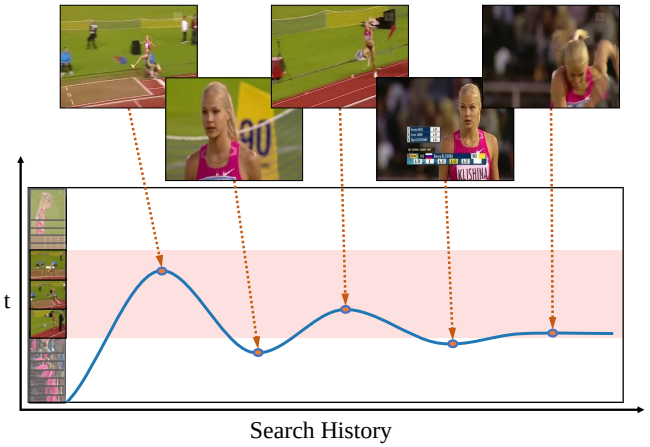

Figure 1. The search steps of a human finding the starting time of a *Long Jump* action in a video. The shaded area is where the *Long Jump* action happens. Notably, humans remarkably do an efficient search for finding human activities in video sequences.

Consider the illustration in Figure 1. It depicts the search process of a human finding the starting time of a *Long Jump* action in a video. The sequence of search steps reveals that the person can quickly find the activity and then smartly refine the beginning of the action. Since humans are remarkably good in finding activities in a video sequence, the process in which they tackle this task must be an efficient and informative search process. Therefore, inspired by human performance in this search task, we seek to develop a computational method that that mimics this human search process accurately, which we expect to lead to better and more efficient detection performance. As such, we introduce a new approach that focuses on learning to search for human activities by imitating the way human annotators do.

Contributions. (i) We propose *Action Search*, a novel approach that mimics the way humans annotate activities in untrimmed video sequences. It uses a Recurrent Neural Network to efficiently search for activities in a video and determine the time boundaries during which these activities

*Both authors contributed equally to this work. Author ordering determined by coin flip.

occur. *Action Search* achieves state-of-the-art results on the THUMOS14 dataset [15]. (ii) To address the lack of data on the behavior of human annotators, we put forward *action searches*, a new dataset composed of the search histories of human annotators for the THUMOS14 dataset [15].

2. Related Work

Datasets. Recognizing and localizing human activities in video often require an extensive collection of annotated data. In recent years, several datasets for temporal action localization have become available. For instance, Jiang *et al*. [15] introduce THUMOS14, a large-scale dataset of untrimmed video sequences that includes 20 different sports categories. At the same time, ActivityNet [4] establishes a large benchmark of long YouTube videos with 200 daily activities annotated. More recently, Sigurdsson *et al*. [29] release *Charades*, a novel video database of day-to-day indoors actions. All these three datasets use human annotators to label the starting and ending points of intended activities in a video. Although this generation of new datasets is opening new challenges in the field, all of them lack an important component: the sequence of steps the human annotator follows to produce the final annotation. In Section 3, we introduce the *Action Searches* dataset, which allows us to disrupt the current paradigm on how action datasets are structured/collected and how action detection models are trained.

Temporal Action Localization. A large number of works have successfully tackled the task of action recognition [21, 30, 33] and Spatio-temporal localization of actions [6, 11, 20, 24, 27, 31]. Here, we briefly review some of the most influential works on temporal action localization. Traditional methods for temporal action localization have relied on the sliding-window-plus-classifier combination to produce predictions about the temporal boundaries of an activity [7, 10, 14, 22]. Recently, a series of works have explored the idea of action proposals to reduce the computational complexity incurred by sliding window-based approaches. Notably, Shou *et al*. [28] introduce a carefully designed multi-stage system that can find regions of interest and classify these regions to produce the temporal action locations. Meanwhile, Caba Heilbron *et al*. [5] propose a sparse learning framework to rank temporal segments based on its similarity to training samples. In the same spirit, Escorcia *et al*. [8] produce action proposals by exploiting the efficacy of Recurrent Neural Networks. Another line of research includes language models [26] or action progression analysis [18] to produce action detections with high fidelity. More related to our work, Yeung *et al*. [35] introduce *Frame Glimpses*, a method to predict the temporal bounds of an action by observing very few frames in the videos. Using Reinforcement Learning, their method can learn a policy to smartly jump through the video. Our model avoids the subtleties of such policies by exploiting ground truth depicting how humans sequentially annotate activities in videos. To the best of our knowledge, no previous approach has used this kind of information for action localization.

Sequence Prediction. Recurrent Neural Networks and specifically Long Short Term Memory (LSTM) Networks have successfully tackled several sequence prediction problems [2, 3, 12, 13]. For instance, Alahi *et al*. [2] introduce an LSTM based model to predict human motion trajectories in crowded scenes. Graves *et al*. [12] propose a recurrent model that learns to predict the next stroke for the task of on-line handwriting prediction. Inspired by these approaches, in Section 4, we introduce a novel Learning-to-Search strategy, which recasts the problem of temporal action localization as a sequence prediction problem.

3. Learning to Search for Actions Like Humans Do

"It is good to have an end to journey toward, but it is the journey that matters in the end." — Ernest Hemingway

Searching for actions in videos is a task that humans can efficiently do. Seminal works in the study of the human visual attention [9, 19] have shown that humans are able to stand out from surrounding distractors when requested to search for a known target. However, current methods for automatic action search lack this ability. We argue that this inability lies on the fact that existing action detection models are trained without an intelligent mechanism of search. This is in part due to the limitations of existing datasets that only provide supervision about where the action is in the video, while ignoring the entire process followed by the annotator in finding the action.

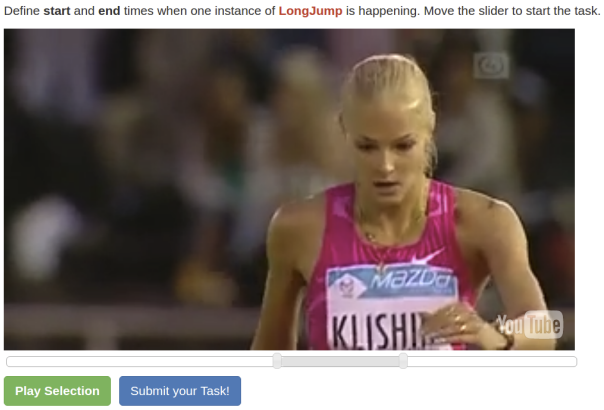

Figure 2. Screen-shot of the human annotation tool used to gather the *action searches* dataset. Our user interface allows the human annotator to slide a time bar to the left or right to quickly search for the start and ending time respectively, as well as, jumping directly to any temporal point in the video.

To address the limitations of the existing datasets, we

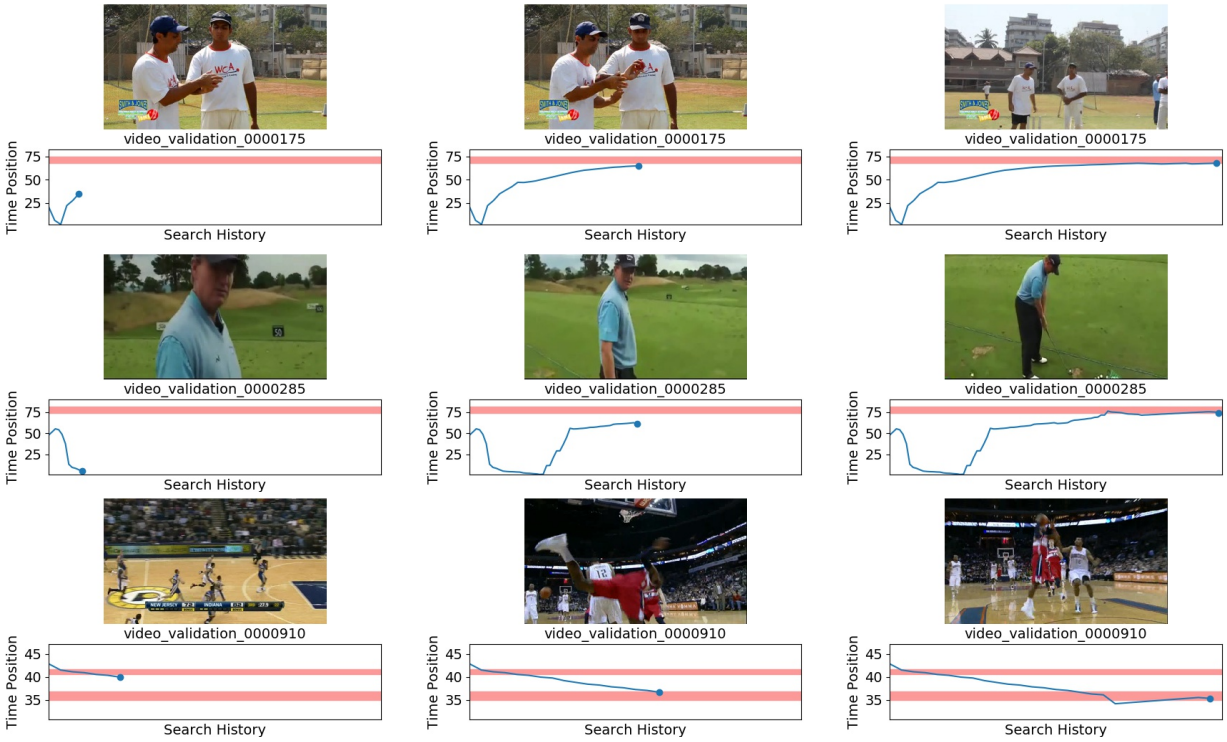

Figure 3. Illustration of *action searches* gathered from our annotation campaign. Each row depicts one search history for a THUMOS14 video. The shaded areas are the regions where an action occurs.

collected a novel dataset of *action searches*, which comprises 200 THUMOS14 videos [15], where each video is annotated with the sequence of steps the annotator follows to define the temporal bounds of the action. We refer to such sequence of steps as the search history of the annotator. To collect the annotations, we rely on Amazon Mechanical Turk (AMT) workers with at least $1,000$ accepted HITs. Low-quality annotations were easily filtered by comparing the final prediction of the annotator with the THUMOS14 ground truth. Our user interface allowed the annotator to slide a time bar to the left or right to quickly search for starting and ending bounds of actions, as well as, jumping directly to any temporal location in the video. Figure 2 shows a snapshot of the human annotation tool used to gather the *action searches* dataset.

Our *action searches* dataset consists of a total of $1,761$ human annotations (search histories). Table 1 shows per-class statistics about *action searches*. Interestingly, human annotators find the activities in 6 jumps and define the starting and ending points in 22 steps. This translates into the observation of only 5% of the video. Figure 3 shows several examples of the collected data. With the release of *action searches*, we hope that new research opportunities can be enabled in the field of video understanding.

Action Class	# Histories	# Steps	# Videos
Baseball Pitch	89	31.1	11
Basketball Dunk	67	34.6	10
Billiards	100	37.9	10
Clean and Jerk	67	27.7	10
Cliff Diving	93	33.0	10
Cricket Bowl	106	39.2	17
Cricket Shot	72	34.9	16
Diving	79	31.2	20
Frisbee Catch	94	38.2	10
Golf Swing	105	36.4	11
Hammer Throw	98	27.7	10
High Jump	64	44.2	10
Javelin Throw	97	24.5	10
Long Jump	94	39.1	11
Pole Vault	85	28.9	10
Shot put	98	34.8	10
Soccer Penalty	103	34.0	10
Tennis Swing	91	34.2	10
Throw Discuss	92	21.0	11
Volleyball Spike	67	22.8	10
All	1761	31.0	200

Table 1. *Action Searches* dataset annotation details. For each activity class, we count the number of collected search histories (# Histories), average number of steps over the collected search histories (# Steps), and the number of videos annotated (# Videos).

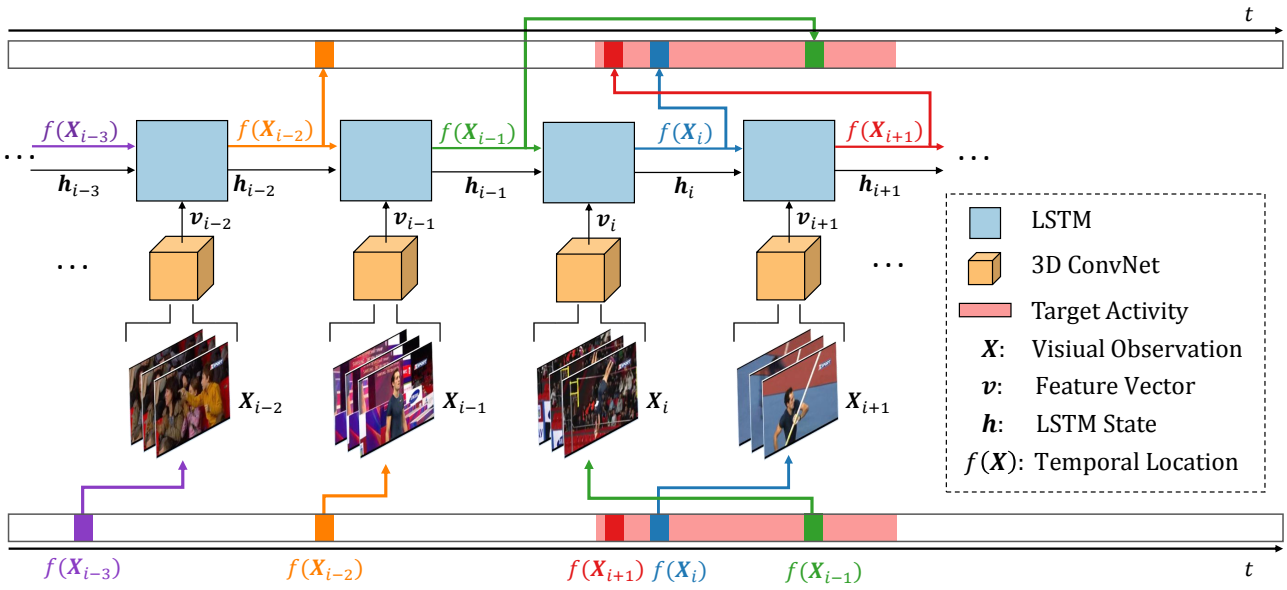

Figure 4. Our model uses the visual information of its current location in the video and the history of what it has observed before to efficiently predict the next location to search in the video. At each step, (i) a 3D convolution visual encoder consumes a small video volume from the model's current temporal location in the video and transfers that volume to a meaningful feature vector; (ii) an LSTM consumes this feature vector along with the state and temporal location produced by the previous step; (ii) the LSTM outputs its updated state and the next prediction of where to search in the video; (iv) the model then moves to the new temporal location, repeating the process.

4. Model

In this section, we provide details for the model architecture of *Action Search*, the approach employed for the training *Action Search*, the prediction process for the testing phase, as well as, some implementation details. *Action Search* determines the temporal boundaries of action locations in two stages: the **Search Stage** and the **Classification and Filtering Stage**. Figure 4 gives an overview of the main architecture of our approach.

Search Stage. The input to this stage is a sequence of the visual observations $(\mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_n)$, while the output is a sequence of temporal locations $(f(\mathbf{X}_1), f(\mathbf{X}_2), \ldots f(\mathbf{X}_n))$ used in the search process. For a given video and at the i^{th} step of the search stage, a 3D convolution visual encoder transforms a small video volume $\mathbf{X}_i$, extracted from the model's current temporal location in the video, to a feature vector $\mathbf{v}_i$. We choose a 3D convolution visual encoder over other types of encoders because 3D ConvNets encode motion, which has proven to be advantageous in action detection [30, 32]. The output of the visual encoder is passed to an LSTM search network. The LSTM search network takes three inputs $(\mathbf{h}_{i-1}, f(\mathbf{X}_{i-1}), \mathbf{v}_i)$, where $\mathbf{h}_{i-1}$ is the LSTM state from the previous step, $f(\mathbf{X}_{i-1})$ is the temporal location outputted by the previous step (*i.e.* the model's current temporal location), and $\mathbf{v}_i$ is the feature vector of the visual observation from this step. After consuming its inputs, the LSTM network outputs its updated state $\mathbf{h}_i$. This updated state is then transformed by a fully-connected layer to produce $f(\mathbf{X}_i)$, the next temporal location in the video that the search model will move to.

Classification and Filtering Stage. The input to this stage is the search sequence $(f(\mathbf{X}_1), f(\mathbf{X}_2), \ldots f(\mathbf{X}_n))$ obtained from the search stage, while the output is a set of predictions for temporal bounds of the activities in the given video. From the search sequence, we generate a set of proposals $\mathcal{P}$ such that

$$\mathcal{P} = \{\, [f(\mathbf{X}_i), d + f(\mathbf{X}_i)] \mid i \in \{1, 2, \ldots, n\} \text{ and } d \in \mathcal{D} \,\}, \quad (1)$$

where $\mathcal{D}$ is a set of activity duration priors that are precomputed from the validation videos and is activity class-specific. The set of proposals $\mathcal{P}$ is filtered using *non-maximum-suppression* (NMS). *Action Search* then classifies each remaining proposal in the given video and ranks them by the classification score.

4.1. Learning to Search

We employ a multi-layer LSTM network in training *Action Search* to produce the search sequence. We train a separate LSTM network for each activity class. For

a training video and at each step, the network consumes visual observation of its current temporal location, along with the temporal location of the previous step. Running for n steps, the network produces a search sequence $(f(\mathbf{X}_1), f(\mathbf{X}_2), \dots f(\mathbf{X}_n))$. Given a search sequence, $(y_1, y_2, \dots, y_n)$, of a human annotator for the same video from our *action searches* dataset, we compute the loss L as the average Huber loss at each step

$$L = \frac{1}{n}\sum_{i=1}^{n} H_\delta(y_i, f(\mathbf{X}_i)), \tag{2}$$

$$H_\delta(y, f(\mathbf{X})) = \begin{cases} \frac{1}{2}(y - f(\mathbf{X}))^2 & \text{if } |y - f(\mathbf{X})| \leq \delta, \\ \delta\,|y - f(\mathbf{X})| - \frac{1}{2}\delta^2 & \text{otherwise.} \end{cases} \tag{3}$$

where $\delta > 0$. We choose mean Huber loss over the mean squared loss, $\frac{1}{n}\sum_{i=1}^{n}(y_i - f(\mathbf{X}_i))^2$, because the latter tends to be dominated by outliers. Moreover, the Huber loss is convex and differentiable in a neighborhood of its minimum, giving it an advantage over the mean absolute loss, $\frac{1}{n}\sum_{i=1}^{n}|y_i - f(\mathbf{X}_i)|$.

4.2. Prediction

In the testing phase, *Action Search* uniformly samples a set of 24 random fixed-length segments from a given test video. We cross-validated the number of random fixed-length segments on the validation subset, and 24 gave the highest average recall of all the activity labels in a given video while maintaining a small number of segments. These 24 segments are then classified in order to obtain the top-k likely global class labels for the given test video. To reduce the computation time and complexity, *Action Search* then only runs the LSTM search models of these k activity classes to produce a set of temporal search sequences. These sequences are then aggregated and transformed into a set of proposals in the same manner described in the **Classification and Filtering Stage** section. *Action Search* filters the resulting set of proposals using NMS, and then classifies each of the remaining proposals. These proposals are then ranked based on their classification score, and finally, *Action Search* produces the activity temporal boundary predictions for the test video based on these ranked proposals.

4.3. Implementation Details

Although our pipeline is differentiable and can be trained end-to-end, we choose to fix the 3D convolution visual encoder and use precomputed C3D features. We also fix the classification part of our pipeline and use the S-CNN classifier introduced by Shou *et al*. [28] to classify the temporal locations of activities. For simplicity, we only train the multi-layer LSTM network in our pipeline.

For a more stable training process, we normalize the ground truth output search sequence in a per-class manner. Each LSTM network is trained using Adam optimizer [17] with an exponential learning rate decay. We unroll the LSTM for a fixed number of steps for the backpropagation computation during training. To regularize the multi-layer LSTM network, we follow the Recurrent Neural Network dropout techniques introduced by Pham *et al*. [25] and Zaremba *et al*. [37]. We use $k = 5$ when choosing the top-k global class labels of a test video during the prediction process (Section 4.2). We cross-validated this value of k using the validation subset of our dataset. Figure 6 shows the effects of the top-k parameter on the recall as a function of the number of proposals. *Action Search* code base is implemented using TensorFlow [1].

5. Experiments

In this section, we show that our *Action Search* model is able to achieve state-of-the-art detection performance in THUMOS14 [15], one of the most challenging datasets for temporal action localization. We first introduce our experimental protocol including a brief description of the dataset and metrics used. Then, we compare several variants of our approach and analyze the impact of the most important parameters. As an intermediary step, we compare the recall performance of *Action Search* against basic baseline models. Finally, we conduct a comparison against state-of-the-art approaches and also present qualitative results depicting some success and failure cases of our model.

5.1. Experimental Settings

Dataset. We conduct our experiments using THUMOS14 [15], which is among the largest available annotated action detection datasets. This dataset contains $13,320,200$ and 213 videos for training, validation and testing respectively. Given that the training set contains only trimmed videos, we use the validation set to train our model. Additionally, we augment our training set with the collected *action searches* described in Section 3.

Metrics. We compare our model against other approaches according to two main metrics, namely Recall and Mean Average Precision (mAP). To compute the recall, we consider a detection as a true positive if the temporal Intersection-over-Union (tIoU) is greater than 0.5. The mAP is computed using the official toolkit provided by THUMOS14 [15].

Pre-processing. To reduce the training time and memory usage of our implementation, we reduce the dimensionality of the C3D features to a 500 dimensional vector using PCA. We collapse *action searches* that contain less than 8 search steps when training our model. We also discard searches that reach final temporal boundary predictions that have a tIoU less than 0.5 with the groundtruth temporal boundaries of the action.

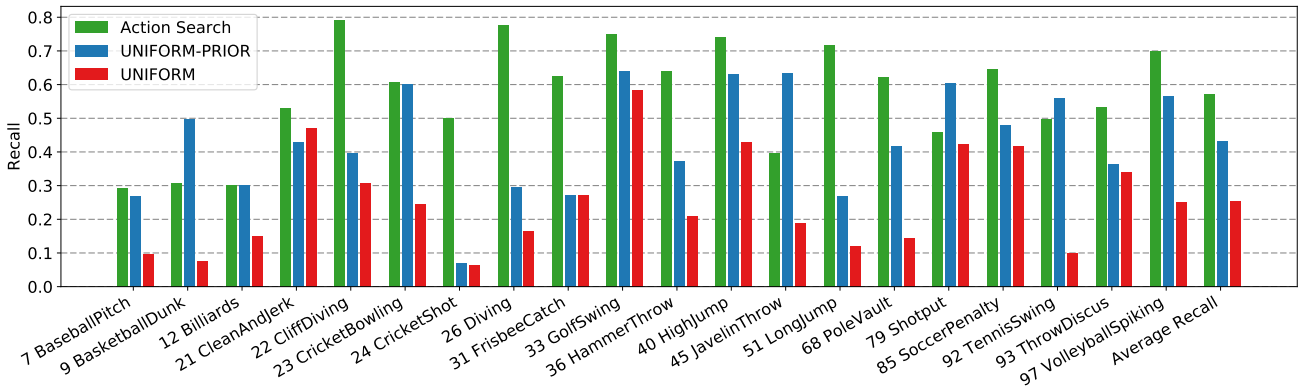

Figure 5. Histogram of recall for each class in THUMOS14 with a tIoU $\alpha = 0.5$

Baseline methods. To demonstrate the effectiveness of our approach in terms of the recall metric, we consider two baseline models, *UNIFORM* and *UNIFORM-PRIOR*.

- *UNIFORM*: This model produces proposals by picking the proposal start time s randomly form a uniform distribution on the interval $[0, d]$, where d is the duration of the video. Then, it picks the proposal end time t randomly form a uniform distribution on the interval $[s, d]$.
- *UNIFORM-PRIOR*: This model produces proposals by picking the proposal start time s randomly form a uniform distribution on the interval $[0, d]$, where d is the duration of the video. Then, it sets the end time to $t = s + p$, where $p > 0$ is drawn from a set of class-specific activity length priors.

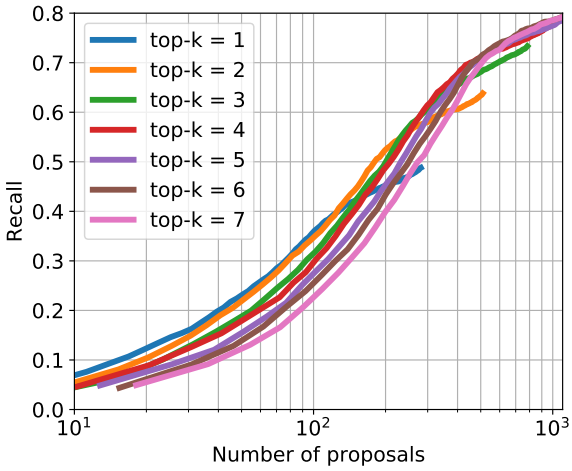

Figure 6. Cross-validation results on the effects of the top-k parameter on recall. k fixed to 5 offers a good trade-off between recall and false positive rate.

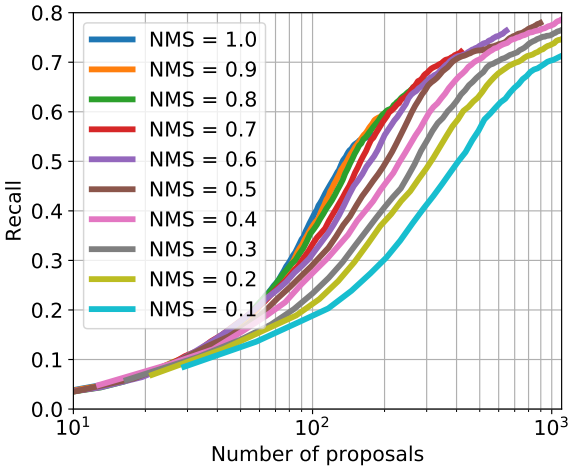

Figure 7. Cross-validation results on the effects of the NMS threshold on recall. NMS threshold fixed to 0.6 offers a good trade-off between recall and false positive rate.

UNIFORM-PRIOR uses the same class-specific activity length priors that *Action Search* uses as described in the **Classification and Filtering Stage** in Section 4.

5.2. Cross Validation

In this section, we consider multiple variants of our approach and analyze the impact of two of the most important *Action Search* model parameters: the k value of the top-k global class labels selected for search and the NMS threshold value used for filtering the proposals. We study the effect of the different values of these two parameters on recall using our validation subset, as summarized in Figures 6 and 7. We observe that choosing bigger k maximizes the ratio of true positives at the cost of introducing a large rate of false positives. To achieve a desirable ratio of true positives *vs* false positives, we fix k to 5 for all experiments in what fol-

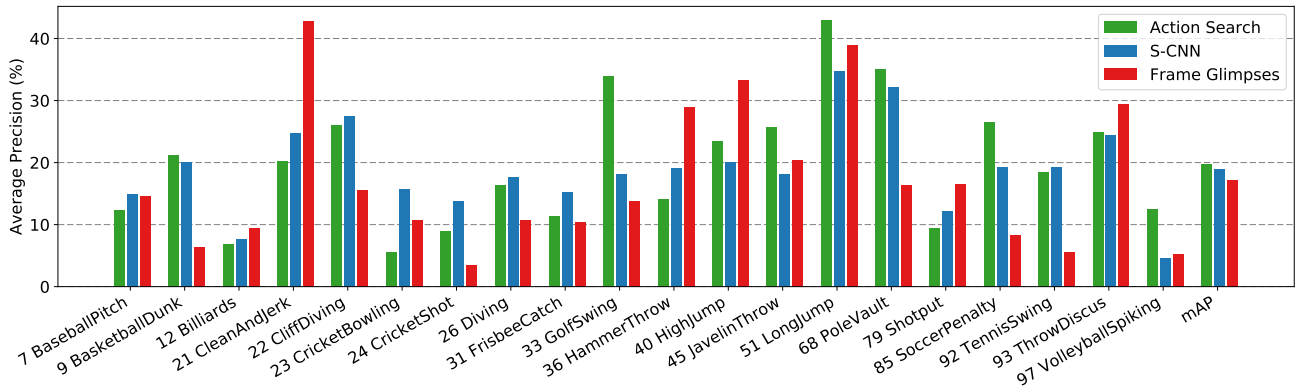

Figure 8. Histogram of average precision (%) for each class in THUMOS14 with a tIoU $\alpha = 0.5$. *Action Search* improves the mAP on classes which contains instances with short durations as compared to the average length of a video (*e.g. VolleyballSpiking*).

	mAP@0.1	mAP@0.2	mAP@0.3	mAP@0.4	mAP@0.5
Karaman *et al.* [16]	1.5	0.9	0.5	0.3	0.2
Wang *et al.* [34]	19.2	17.8	14.6	12.1	8.5
Caba Heilbron *et al.* [5]	36.1	32.9	25.7	18.2	13.5
Escorcia *et al.* [8]	*	*	*	*	13.9
Oneata *et al.* [23]	39.8	36.2	28.8	21.8	15.0
Richard and Gall [26]	39.7	35.7	30.0	23.2	15.2
Yeung *et al.* [35]	48.9	44.0	36.0	26.4	17.1
Yuan *et al.* [36]	**51.4**	42.6	33.6	26.1	18.8
Shou *et al.* [28]	47.7	43.5	36.3	**28.7**	19.0
Action Search	49.6	**44.3**	**38.1**	28.4	**19.8**

Table 2. Results for state-of-the-art action detection methods on the THUMOS14 testing set. We report the performance using mAP at tIoU = α (mAP@α). *Action Search* is in the two top-performers over all tIoU thresholds.

lows. Similarly, we also observe that higher NMS threshold values increase the ratio of true positives but at the cost of allowing a larger number of false positives. We fix the NMS threshold to 0.6, which strikes a desirable tradeoff between high recall and a low false positive rate.

5.3. Recall Analysis

In Figure 5, we compare *Action Search* against the *UNIFORM* and *UNIFORM-PRIOR* models in terms of the recall for each activity class in THUMOS14 (or equivalently the detection performance if an ideal action classifier is used). The average recall of *Action Search* is higher than those of the two baseline models. Moreover, *Action Search* preforms significantly better in classes like *Cricket Shot* and *Long Jump*. Interestingly, these classes have average activity lengths that are short and their activity instances are sparsely distributed across the video duration. The two baseline models achieve their best performance on classes, such as *Golf Swing*, which have average activity lengths that are long and their activity instances cover most of the video.

5.4. Comparison Against State-of-the-Art Methods

Table 2 compares *Action Search* against state-of-the-art detection approaches on the THUMOS14 testing set. Our model improves the action detection performance by 0.8%, while keeping an attractive computational cost. We achieve a significant improvement of 2.7% over Frame Glimpses [35]. We attribute this to the fact that our model is trained to imitate how humans search for activities instead of being left to learn and explore by itself, as is the case in Frame Glimpses [35]. When compared to S-CNN [28], our model gains 0.8% mAP. This is an indication of the quality of the proposals generated by our method. In Figure 8, we further compare the AP performance per class against the top two performers. Interestingly, our approach outperforms state-of-the-art methods in classes such as *VolleyballSpiking*, which contains instances with short durations as compared to the average length of a video.

5.5. Qualitative Results

Our model can find the starting temporal point of actions in most cases. Here, we provide some illustrative qualitative

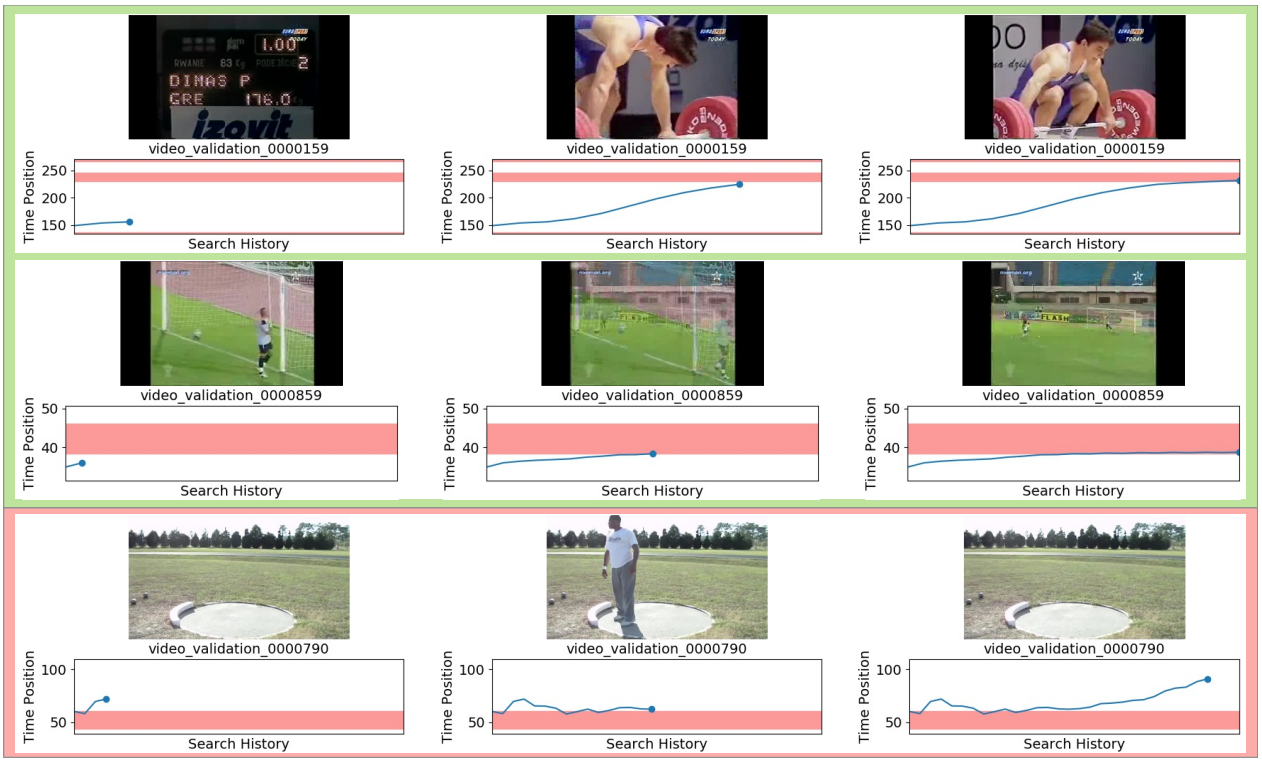

Figure 9. The first two rows show examples when our model correctly retrieves the start time of an action. Last row depicts an example when our model fails at finding the action. The shaded areas are regions where an action occurs.

results. Figure 9 depicts 2 examples in which *Action Search* correctly retrieves an action and 1 example in which our approach fails at finding the starting time. We see that our model fails in some cases where it overshoots the action. In those cases, the model keeps exploring the video till the end without finding the action of interest.

6. Conclusion

We introduced *Action Search* a new learning model to imitate how people search for human activities in video sequences. We have introduced a new dataset called *action searches* to train our model. Our experimental analysis demonstrates that *Action Searches* produces reliable action detections when tested on one of the largest available datasets for action detection. We plan to release our *action searches* dataset to the vision community and expect that further works can extend the use of search processes for action detection.

References

[1] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[2] A. Alahi, K. Goel, V. Ramanathan, A. Robicquet, L. Fei-Fei, and S. Savarese. Social lstm: Human trajectory prediction in crowded spaces. In *CVPR*, 2016.

[3] S. Bengio, O. Vinyals, N. Jaitly, and N. Shazeer. Scheduled sampling for sequence prediction with recurrent neural networks. In *NIPS*, 2015.

[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, J. C. Niebles, and B. Ghanem. Fast temporal activity proposals for efficitent detection of human actions in untrimmed videos. In *CVPR*, 2016.

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

[7] O. Duchenne, I. Laptev, J. Sivic, F. Bach, and J. Ponce. Automatic annotation of human actions in video. In *ICCV*, 2009.

[8] V. Escorcia, F. Caba Heilbron, J. C. Niebles, and B. Ghanem. Daps: Deep action proposals for action understanding. In *ECCV*, 2016.

[9] J. A. Fodor. *The modularity of mind: An essay on faculty psychology*. MIT press, 1983.

[10] A. Gaidon, Z. Harchaoui, and C. Schmid. Actom sequence models for efficient action detection. In *CVPR*, 2011.

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

[12] A. Graves. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*, 2013.

[13] A. Graves and N. Jaitly. Towards end-to-end speech recognition with recurrent neural networks. In *International Conference on Machine Learning*, 2014.

[14] M. Jain, J. C. van Gemert, and C. G. Snoek. What do 15,000 object categories tell us about classifying and localizing actions? In *CVPR*, 2015.

[15] 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.

[16] S. Karaman, L. Seidenari, and A. Del Bimbo. Fast saliency based pooling of fisher encoded dense trajectories.

[17] D. P. Kingma and J. Ba. Adam: A Method for Stochastic Optimization. *ArXiv e-prints*, Dec. 2014.

[18] S. Ma, L. Sigal, and S. Sclaroff. Learning activity progression in lstms for activity detection and early detection. In *CVPR*, 2016.

[19] B. McElree and M. Carrasco. The temporal dynamics of visual search: evidence for parallel processing in feature and conjunction searches. *Journal of Experimental Psychology: Human Perception and Performance*, 25(6):1517, 1999.

[20] P. Mettes, J. C. van Gemert, and C. G. M. Snoek. Spot on: Action localization from pointly-supervised proposals. In *ECCV*, 2016.

[21] J. C. Niebles, C.-W. Chen, and L. Fei-Fei. Modeling temporal structure of decomposable motion segments for activity classification. In *ECCV*, 2010.

[22] D. Oneata, J. Verbeek, and C. Schmid. Efficient action localization with approximately normalized fisher vectors. In *CVPR*, 2014.

[23] D. Oneata, J. Verbeek, and C. Schmid. The lear submission at thumos 2014. 2014.

[24] X. Peng and C. Schmid. Multi-region two-stream r-cnn for action detection. In *ECCV*, 2016.

[25] V. Pham, T. Bluche, C. Kermorvant, and J. Louradour. Dropout improves Recurrent Neural Networks for Handwriting Recognition. *ArXiv e-prints*, Nov. 2013.

[26] A. Richard and J. Gall. Temporal action detection using a statistical language model. In *CVPR*, 2016.

[27] S. Saha, G. Singh, M. Sapienza, P. H. Torr, and F. Cuzzolin. Deep learning for detecting multiple space-time action tubes in videos. In *BMVC*, 2016.

[28] Z. Shou, D. Wang, and S.-F. Chang. Temporal action localization in untrimmed videos via multi-stage cnns. In *CVPR*, 2016.

[29] G. A. Sigurdsson, G. Varol, X. Wang, A. Farhadi, I. Laptev, and A. Gupta. Hollywood in homes: Crowdsourcing data collection for activity understanding. In *ECCV*, 2016.

[30] 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.

[31] K. Soomro, H. Idrees, and M. Shah. Predicting the where and what of actors and actions through online action localization. In *CVPR*, 2016.

[32] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri. Learning spatiotemporal features with 3d convolutional networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4489–4497, 2015.

[33] H. Wang, A. Kläser, C. Schmid, and C.-L. Liu. Action recognition by dense trajectories. In *CVPR*, 2011.

[34] L. Wang, Y. Qiao, and X. Tang. Action recognition and detection by combining motion and appearance features.

[35] S. Yeung, O. Russakovsky, G. Mori, and L. Fei-Fei. End-to-end learning of action detection from frame glimpses in videos. In *CVPR*, 2016.

[36] J. Yuan, B. Ni, X. Yang, and A. A. Kassim. Temporal action localization with pyramid of score distribution features. In *CVPR*, 2016.

[37] W. Zaremba, I. Sutskever, and O. Vinyals. Recurrent Neural Network Regularization. *ArXiv e-prints*, Sept. 2014.