Action Search: Spotting Actions in Videos and Its Application to Temporal Action Localization

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
http://www.humamalwassel.com/publication/action-search/

Abstract. State-of-the-art temporal action detectors inefficiently search the entire video for specific actions. Despite the encouraging progress these methods achieve, it is crucial to design automated approaches that only explore parts of the video which are the most relevant to the actions being searched for. To address this need, we propose the new problem of *action spotting* in video, which we define as finding a specific action in a video while observing a small portion of that video. Inspired by the observation that humans are extremely efficient and accurate in spotting and finding action instances in video, we propose *Action Search*, a novel Recurrent Neural Network approach that mimics the way humans spot actions. Moreover, to address the absence of data recording the behavior of human annotators, we put forward the *Human Searches* dataset, which compiles the search sequences employed by human annotators spotting actions in the AVA and THUMOS14 datasets. We consider temporal action localization as an application of the *action spotting* problem. Experiments on the THUMOS14 dataset reveal that our model is not only able to explore the video efficiently (observing on average **17.3**% of the video) but it also accurately finds human activities with **30.8**% mAP.

Keywords: Video understanding · Action localization
Action spotting

1 Introduction

Similar to many video-related applications, such as video object detection and video surveillance, temporal action localization requires an efficient search for

The first two authors contributed equally to this work. Authors ordering was determined by three coin flips.

Electronic supplementary material The online version of this chapter (https://doi.org/10.1007/978-3-030-01240-3_16) contains supplementary material, which is available to authorized users.

V. Ferrari et al. (Eds.): ECCV 2018, LNCS 11213, pp. 253–269, 2018.
https://doi.org/10.1007/978-3-030-01240-3_16

different visual targets in videos. With the recent exponential growth in the number videos online (*e.g.* over 400 video hours are uploaded to YouTube every minute), it is crucial today to develop methods that can simultaneously search this large volume of videos efficiently and spot actions accurately. Thus, we propose the new problem of *action spotting* in video, which we define as finding a specific action in a video sequence while observing a small portion of that video. Since the computational cost is directly impacted by the number of observations made in a video, this spotting problem brings search efficiency to the forefront. Obviously, the overall computational cost of such an action search can be reduced by making the per-observation computation faster, as done in many previous work; however, as the video becomes long and the action instances sparse in the video, the number of observations needed dominates this cost.

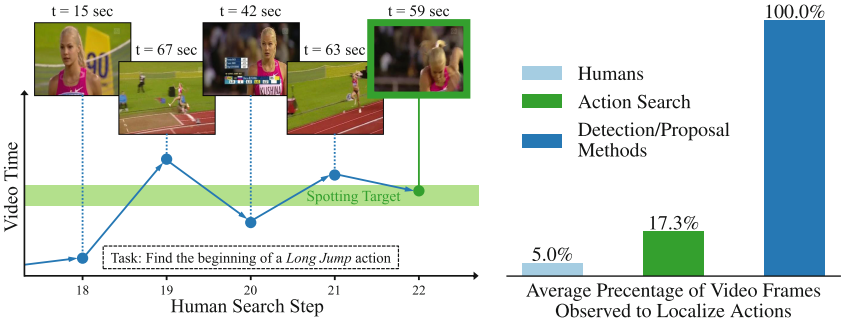

Fig. 1. Left: A partial search sequence a human performed to find the start of a *Long Jump* action (the shaded green area). Notably, humans are efficient in spotting actions without observing a large portion of the video. **Right:** An efficiency comparison between humans, *Action Search*, and other detection/proposal methods on THUMOS14 [24]. Our model is 5.8x more efficient than other methods. (Color figure online)

To get intuitions on how to automatically perform an efficient action search in video, we take notice of how humans approach the problem. In Fig. 1 (left), we show part of a search sequence a human observer carries out when asked to find the beginning of a *Long Jump* action in a long video. This sequence reveals that the person can quickly find the spotting target (in 22 search steps) without observing the entire video, which indicates the possible role temporal context plays in searching for actions. In this case, only a very small portion of the video is observed before the search successfully terminates. In fact, we observe a similar search pattern for different action targets and across different annotators.

Early approaches for temporal action localization rely on trimmed data to learn sophisticated models [29,41,46]. These methods achieve great success at detecting human actions. Despite the encouraging progress in action localization and the attention it is recently attracting, the goal of accurate detection remains elusive to automated systems. A main drawback of current detection methods is

that they do not exploit (and totally discard) the search process human annotators follow to produce the final temporal annotations, which are inherently the only information from the annotation process used to train these detection models. As such, these methods need to scan the whole video in an exhaustive manner (either by observing all video frames or a uniform temporal subsampling of them) to detect human actions. This is inefficient; thus, it is crucial to investigate methods that observe the smallest percentage of the video and maintain a state-of-the-art mAP. This naturally leads to faster, more efficient methods, which implies that *action spotting* is essential for temporal action localization.

In this paper, we focus on *action spotting* as a precursor to temporal action localization, since one can reformulate action localization as an *action spotting* problem followed by a regression task to define the action length. Based on observations drawn from the user study in Sect. 3, we believe that *action spotting* is an effective precursor that departs from the focus of the traditional action proposal precursor. Action proposal generation aims to localize the temporal bounds of actions as tightly as possible, while the goal of *action spotting* is to search for action instances as efficiently as possible. Moreover, action proposal generation is class-agnostic, while *action spotting* is class-specific. Figure 1 (right) compares the efficiency of action localization using traditional approaches (*e.g.* proposals) against using our *action spotting*-based method.

Recent temporal action localization datasets [6,24,37] use human annotators to label the action boundaries in a video. Although these datasets are opening new and exciting challenges in the field, they lack an important component: the sequence of steps the human annotator follows to produce the final annotation. These search sequences can be collected for *free* when creating new datasets or extending current ones, and they are a valuable resource for *action spotting*.

Inspired by the observation that humans are extremely efficient and accurate in finding individual action instances in a video, we aim to solve the *action spotting* problem in this paper by imitating how humans search in videos.

Contributions. (i) To address the lack of data on the behavior of human annotators, we put forward the *Human Searches* dataset, a new dataset composed of the search sequences of human annotators for the AVA [21] and THUMOS14 [24] datasets (Sect. 3). **(ii)** We propose *Action Search*, a novel Recurrent Neural Network approach that mimics the way humans spot actions in untrimmed videos (Sect. 4). **(iii)** We validate *Action Search* in the *action spotting* problem by demonstrating it requires on average **16.6**% and **22.3**% fewer observations to successfully spot an action than two baseline models (Sect. 5.1). Moreover, when our model is used in the domain of temporal action localization, it achieves state-of-the-art detection results on the THUMOS14 [24] dataset with **30.8**% mAP while only observing on average **17.3**% of the video (Sect. 5.2).

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.* [24] introduce THUMOS14, a large-scale dataset of untrimmed video sequences with 20 different sports categories. Concurrently, ActivityNet [6] establishes a large benchmark of long YouTube videos with 200 annotated daily activities. Later, Sigurdsson *et al.* [37] release *Charades*, a day-to-day indoor actions database captured in a crowdsourced manner. More recently, Google introduced AVA [21], short for *atomic visual actions*, a densely annotated dataset localizing human actions in space and time. All four datasets use human annotators to localize intended activities in a video. Although these datasets are opening new challenges in the field, they all miss an important component: the search sequence the human annotator follows to produce the final annotation, which is a form of supervised annotation that can be collected for *free* during the annotation process. In Sect. 3, we introduce the *Human Searches* dataset, which allows us to disrupt the current paradigm on how action datasets are structured/collected and how action spotting/localization models can potentially be trained.

Temporal Action Localization. A large number of works have successfully tackled the task of action recognition [8,9,28,38,42] and spatio-temporal localization [10,18,27,30,33,39]. Here, we briefly review some of the influential works in the realm of temporal action localization. Early methods have relied on the sliding-window-plus-classifier combination to produce the actions temporal boundaries [13,15,23,29]. 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.* [35] introduce a multi-stage system that finds and classifies interest regions to produce temporal action locations. Meanwhile, Caba Heilbron *et al.* [7] propose a sparse learning framework to rank a segment based on its similarity to training samples. In the same spirit, contemporary works [4,14,17] produce action proposals by exploiting the effectiveness of deep neural networks. More recently, end-to-end methods have proven to boost the performance of two-stage approaches, demonstrating the importance of jointly optimizing the feature extraction and detection process [3,41,43,46]. Other researchers have used language models [32], action progression analysis [26,34], and high-level semantics [5] to produce high-fidelity action detections.

The large body of work on temporal action localization has mostly focused on improvements in detection performance and/or speed, while very few works have targeted the development of efficient search mechanisms. The dominant search strategy in prior work has focused on exhaustively observing the entire video (either by observing all frames or a uniform temporal subsampling of them) at least once. The pioneering approach of Yeung *et al.* [44] comes as a departure from this paradigm, whereby they introduce the *Frame Glimpses* method to predict the temporal bounds of an action by observing very few frames. Their reinforcement learning-based method tries to learn a policy to intelligently jump through the video for the immediate purpose of detection. While it shares a similar goal, our model avoids the subtleties of learning such policies by exploiting ground truth data depicting how humans sequentially annotate actions. Using such information for action localization, our method outperforms *Frame*

Glimpses and achieves state-of-the-art detection results on THUMOS14 [24] (see Sect. 5.2).

Sequence Prediction. Recurrent Neural Networks (RNNs) and specifically Long Short Term Memory (LSTM) networks have successfully tackled several sequence prediction problems [1,2,19,20]. For instance, Alahi *et al.* [1] introduce an LSTM based model to predict human motion trajectories in crowded scenes. Graves *et al.* [19] propose an RNN that learns to predict the next stroke in online handwriting prediction. Motivated by the success of these approaches, Sect. 4 introduces our novel Learning-to-Search strategy, which formulates the problem of *action spotting* as a sequence prediction problem.

3 Action Spotting: What Do Humans Do?

In this section and motivated by how well humans spot and localize actions in videos, we investigate some factors that intuitively seem to play a role in this process. Specifically, we address the following questions: are humans distracted when they are asked to find multiple action classes? and is spotting an action easier than finding its temporal boundaries? Finally, we describe our key idea.

Single vs. Multiple Class Search. To investigate whether humans are distracted when asked to find multiple action classes, we conduct an online user study on Amazon Mechanical Turk. We ask participants to find an instance of a particular action class in a 15 min video. We design a user interface that includes a time bar, which allows Turkers to navigate over the video quickly until the action is found. Typically, the action instances last three seconds, are sparsely localized, and at least one occurs in the video.

We investigate two variants of the task: (i) a *single class search* to find one instance of a given action class and (ii) a *multiple class search*, which asks Turkers to find one instance from a larger set of action classes. By logging Turker interactions with the user interface, we measure the number of observed frames they require to find an action instance. As compared to *single class search*, we find that Turkers observe 190% and 210% more frames when asked to find an action instance among 10 and 20 action classes, respectively. This observation motivates our intuition that action search can be performed more efficiently when the target task is "simpler" (*i.e.* it incorporates a smaller number of action classes). See the **supplementary material** for more details about this experiment.

Spotting vs. Localization. *Action spotting* is the process of finding any temporal occurrence of an action (*e.g.* any frame inside a *Long Jump* instance) while observing as little as possible from a video. In contrast, action localization focuses on pinpointing precise temporal extents of an action (*i.e.* the exact start and end of a *Long Jump* instance), which usually leads to fine-grained and exhaustive search mechanisms. To measure the effects of searching for fine-grained targets, we ask Turkers to find the starting time of a single target action in a video. We note that Turkers tend to first spot the action and then refine its temporal boundaries. Interestingly, while searching, Turkers perform *three times* more

search steps to refine the temporal boundaries of an action, as compared to the number of search steps needed for spotting the same action. This observation motivates our intuition that *action spotting* can be performed more efficiently than action localization, especially when action instances are short in duration and sparse in frequency within a video. We partially attribute this behavior to the fact that determining precise temporal extents of some actions is ambiguous [36]. See the **supplementary material** for more details about this experiment.

Key Idea. Searching for and localizing actions in video is a task humans can do efficiently. However, current automatic methods lack such ability. This shortcoming arises primarily from the fact that existing models are trained without an intelligent search mechanism. This is in part due to the limitations of existing datasets which only provide supervision about the action's temporal location in the video, while ignoring the entire process the annotator follows to find this action. To address these limitations, we collect two novel datasets of *Human Searches*, where videos are annotated with the search steps a human follows to temporally spot/localize actions. We refer to such step sequences as *search sequences*. Below, we describe these collected datasets (refer to the **supplementary material** for the annotation process details and additional statistics).

(i) AVA searches (targets are actions): We collect search sequences from the AVA v1.0 dataset [21], which is composed of feature films and contains 192 15-min-long videos. We select 15 action classes (out of the original 80 actions) based on the two conditions: (1) the average action coverage is relatively small; and (2) the action class has at least 10 videos in the training set. Based on these two conditions, actions such as *talk to*, *stand*, or *watch (a person)* are discarded due to their extremely large coverage. Moreover, actions like *shovel*, *kick*, or *exit* are discarded because they only have a few training samples. To gather the search sequences, we assign Turkers the task of spotting action instances in the AVA training videos. In other words, the Turker's task is to spot *any* frame inside the temporal bounds of a given action. We only accept workers with more than 1000 HITs submitted. In addition, we use the existing AVA dataset ground truth to filter out noisy annotations. We use a total of 139 AVA training videos and collect 3988 search sequences. Notably, humans only observe a very small portion of the video (less than 1%) before spotting the action.

(ii) THUMOS14 searches (targets are actions starting times): We collect 1761 search sequences from the training videos of THUMOS14 [24], a large-scale dataset of untrimmed videos with 20 different sports categories. We aim to use this searches dataset for *action spotting* with the purpose of *action localization*. Using the same collection process as in the *AVA searches*, we ask Turkers to find an action's *starting time*. We choose to define the Turkers' task this way because defining the action's starting time is easier than defining its end time [36]. Analyzing the collected data, we observe that humans find the actions in 6 steps on average and define the starting points with an additional 16 steps. This translates into observing only 5% of

the video. Figure 2 shows two examples from the collected dataset. With its release, we hope *Human Searches* will enable new research directions in the video understanding field.

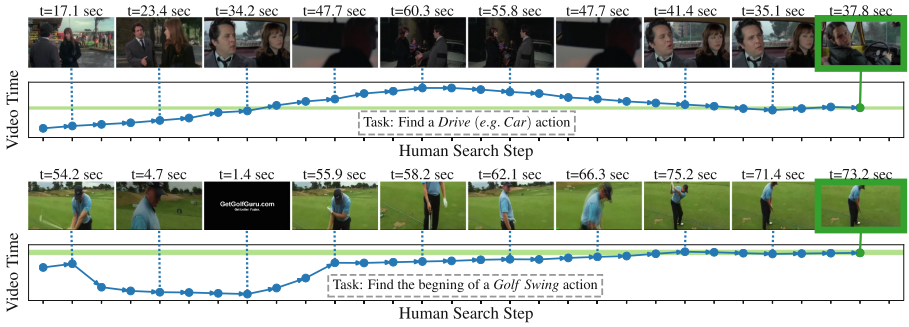

Fig. 2. Illustration of two human search sequences from our *Human Searches* dataset for an AVA [21] training video (first row) and a THUMOS14 [24] training video (second row). The shaded green areas are where the search targets occur. (Color figure online)

4 Proposed Action Search Model

In this section, we discuss the *Action Search* model architecture, the approach used to train it, and some specific implementation details. Figure 3 gives an overview of the main architecture of our approach.

4.1 Model Architecture

The input to *Action Search* is a sequence of visual observations $(\mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_n)$ and the output is a sequence of temporal locations $(f(\mathbf{X}_1), f(\mathbf{X}_2), \ldots, f(\mathbf{X}_n))$, *i.e.* a search sequence, produced by the search process. For a given video and at the i^{th} search step, a visual encoder represents the observation $\mathbf{X}_i$ extracted from the model's current temporal location in the video by a feature vector $\mathbf{v}_i$. We cast the search mechanism as a sequential decision making process modeled by an LSTM search network. This LSTM 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 (an aggregation of information from previous steps), $f(\mathbf{X}_{i-1})$ is the predicted temporal location from the previous step (the model's current location), and $\mathbf{v}_i$ is the feature vector representing the current visual observation. Finally, a fully-connected layer transforms $\mathbf{h}_i$ (the updated state) to produce $f(\mathbf{X}_i)$, the next location to search in the video.

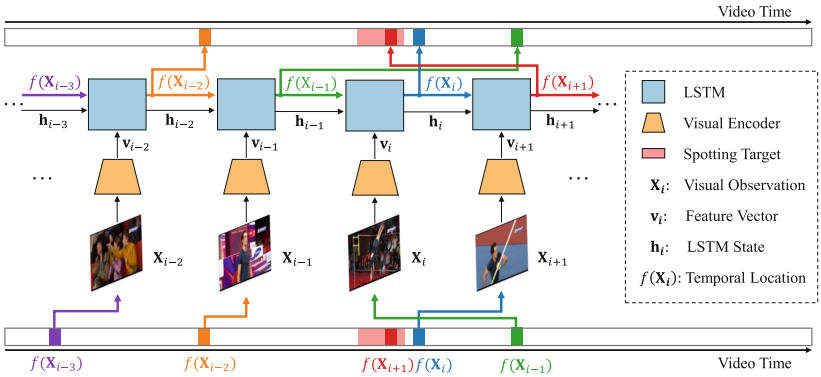

Fig. 3. Our model harnesses the temporal context from its current location and the history of what it has observed to predict the next search location in the video. At each step, (i) a visual encoder transforms the visual observation extracted from the model's current temporal location to a representative feature vector; (ii) an LSTM consumes this feature vector plus the state and temporal location produced in the previous step; (iii) the LSTM outputs its updated state and the next search location; (iv) the model moves to the new temporal location.

4.2 Learning to Search

We employ a multi-layer LSTM network in training *Action Search* to produce search sequences that align with human search sequences in the same video. Following the observations from the user study in Sect. 3, *action spotting* should be class-driven, and thus we train a separate LSTM for each action class.

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

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

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

where $\delta > 0$. We choose the mean Huber loss over the mean squared loss, $\frac{1}{n}\sum_{i=1}^{n}(y_i - f(\mathbf{X}_i))^2$, in order to overcome the effects of outliers that might arise from the different Turkers searching the same video. Moreover, the mean Huber loss is convex and differentiable in a local neighborhood around its minimum, giving it an advantage over the mean absolute loss, $\frac{1}{n}\sum_{i=1}^{n}|y_i - f(\mathbf{X}_i)|$.

4.3 Implementation Details

Training Stage. Although our pipeline is differentiable and can be trained end-to-end, we simplify and expedite training by fixing the visual encoder to precomputed ResNet-152 [22] features (pretrained on ImageNet [12]) extracted from the average-pooling layer. We reduce the feature dimensionality to 512 using PCA. We employ teacher forcing for training the LSTMs. For a stable training process, we represent each $f(\mathbf{X}_i)$ and y_i as relative steps to the previous search location and normalize the ground truth output search sequence in a per-class fashion. Each LSTM network is trained using the Adam optimizer [25] with an exponential learning rate decay. We unroll the LSTM for a fixed number of steps, ranging from 2^2 to 2^5, for the backpropagation computation. To regularize the multi-layer LSTM network, we follow the RNN dropout techniques introduced by Pham *et al.* [31] and Zaremba *et al.* [45]. We set $\delta = 1$ in the Huber loss.

Inference Stage. Since videos typically contain multiple instances of the same action class, we initialize our model at multiple random points. Each search is run for a fixed number of steps. The number of initial points and the search sequence length are cross-validated per class using the validation subset. However, we prefer to launch many short search sequences as opposed to few long ones, since LSTM states tend to saturate and become unstable after a large number of iterations [40]. Thus, one may view the *Action Search* model as a random sampler with a smart local search: the first search steps are a random sampling of the video (exploration), while the later search steps are fine-grained steps (local search) that rely on the temporal context accumulated throughout the search.

4.4 Model Variants

Here, we present different variants of *Action Search* network structure and training. Experiments using these flavors show minimal effects on the final results.

Training with Weighted Loss. In order to put more emphasis on learning the steps towards the end of the ground truth search sequence, each term in the loss L is weighted inversely proportional to how close the search step is to the target action instance. We consider this variant to give less weight to learning the first search steps a human annotator makes (seemingly random) before accumulating enough temporal context knowledge to guide subsequent search steps.

Early Stopping Confidence Score. Another flavor integrates a new module that consumes the LSTM state $\mathbf{h}$ along with the feature vector $\mathbf{v}$ at each step and produces a probability p to determine if the search sequence has reached the spotting target. To train this model variant, we assign a probability $p = 1$ to all frames inside the spotting targets and $p = 0$ elsewhere. We add to L a new term that computes the softmax cross-entropy loss of this confidence score.

5 Experiments

5.1 Action Search for Action Spotting

In this subsection, we demonstrate that our model is able to mimic the human search process when spotting actions in the AVA dataset [21]. We first introduce our experimental protocol including a brief description of the dataset and metric used. Then, we compare *Action Search* against two spotting baseline models.

Dataset. We conduct this experiment using the AVA v1.0 dataset [21], which is among the largest annotated action datasets and is composed of feature films. We pick 15 action categories out of the original 80 set of actions. We train our model using the collected *AVA searches* from our *Human Searches* dataset. However, we prune off the search sequences with less than 8 steps. Refer to Sect. 3 for a description of the *AVA searches* and details about the action categories selection criteria. We evaluate our model on 35 testing videos. We use AVA 3-s annotations to define temporal boundaries for each action instance.

Metric. We compare our model against other approaches according to the *action spotting* metric, which we define to be the expected number of unique observations made per video until spotting an action. An action is spotted if the model lands *anywhere* between its ground truth temporal bounds.

Baseline Methods. To demonstrate the effectiveness of our approach in learning to search efficiently, we consider two baseline models, *Random Baseline* and *Direction Baseline* (refer to the **supplementary material** for more baselines).

Random Baseline: This model picks both the search direction and step size randomly. In particular, if the current search step is at time t of the video, it randomly picks between searching before t in the interval $[0, t]$ or after t in the interval $[t, d]$, where d is the duration of the video. The model then picks the next search location randomly from a uniform distribution on the selected interval.

Direction Baseline: This model picks the search direction using a trained *direction network* and chooses the search step size randomly. In particular, if the current search step is at time t of the video, its *direction network* uses the visual observation of the current frame to decide between searching before or after t, and then picks the next search location randomly from a uniform distribution on the selected interval. The *direction network* is based on a ResNet-152 [22] architecture with a binary softmax classifier. To train this network, we annotated each video frame with the search direction that leads to the nearest ground truth instance boundary. The *direction network* is class-specific and achieves an average of 95% training and 91% validation accuracy per class.

Results and Analysis. We run *Action Search* and the two baseline methods for 1000 independent search trials per test video to compute the *action spotting* metric. At each trial, we initialize each model at a random temporal location in the video, and we reinitialize the search model if it fails to spot an action after 500 steps. Figure 4 shows the cumulative performance of all three models over videos with different action coverage (*i.e.* the percentage of the video

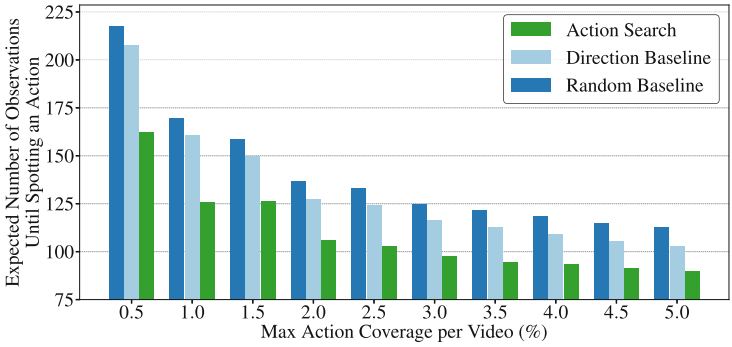

Fig. 4. *Action spotting* results for the AVA testing set for 1000 independent search trials per video. We report the cumulative spotting metric results on videos with action coverage (*i.e.* the percentage of video containing actions) $\leq$5%. *Action Search* takes 22%, 17%, and 13% fewer observations than the *Direction Baseline* on videos with at most 0.5%, 2.5%, and 5% action coverage, respectively.

length that contains action instances). Unlike both the *Direction Baseline* and *Random Baseline*, *Action Search* is able to harness temporal context to expedite the search process and, as a result, takes on average **16.6**% and **22.3**% less observations, respectively, before successfully spotting actions. Notably, there is a great performance difference between our model and the two baselines in videos with very sparse actions (*i.e.* action coverage $\leq$1%). We attribute this to the importance of temporal context when searching for sparse actions in video.

5.2 Action Search for Action Localization

In this subsection, we combine *Action Search* with an off-the-shelf action classifier to temporally localize human actions. Our approach achieves state-of-the-art detection performance on THUMOS14 [24], one of the most challenging temporal action localization datasets, while observing only **17.3**% of the video frames on average. We first explain how *action spotting* is used as a precursor to temporal action localization, and then introduce our experimental protocol, including a brief description of the dataset and metric. Finally, we compare our method against state-of-the-art detectors and present an ablation study of our model.

From Action Spotting to Localization. We train *Action Search* to spot the *start* of actions using the *THUMOS14 searches* dataset described in Sect. 3. To define the spotted action length, we use a simple heuristic based on class-specific fixed prior lengths. Specifically, from an output search sequence $(f(\mathbf{X}_1), f(\mathbf{X}_2), \ldots, f(\mathbf{X}_n))$, produced by our model, we generate the temporal segments $\mathcal{Q} = \{\, [f(\mathbf{X}_i), f(\mathbf{X}_i) + d] \mid i \in \{1, \ldots, n\};\ d \in \mathcal{D} \,\}$, where $\mathcal{D}$ is a set of class-specific fixed action duration priors precomputed from the validation videos. In pursuit of assigning a class label to each segment, we use a pretrained classifier to provide probabilistic action scores. Finally, we follow the standard practice of applying *non-maximum-suppression* to remove duplicate detections.

Prediction. To reduce the number of search models we apply on a video to obtain its temporal predictions, we perform the following steps. We uniformly sample a small random set of N fixed-length segments from a given test video. These N segments are then classified using an off-the-shelf action classifier in order to obtain the top-k likely global class labels for the test video. To reduce complexity, *Action Search* then only runs the LSTM search models associated with these k activity classes to produce a set of temporal search sequences. These sequences are then aggregated and transformed into a set of final temporal predictions using the class-specific duration priors described above.

Dataset. We conduct our experiment using THUMOS14 [24], one of the most popular datasets for temporal localization. It contains 200 and 213 videos for training and testing, respectively. To train our *Action Search* model, we use the *THUMOS14 searches* dataset described in Sect. 3 (discarding the search sequences with less than 8 search steps). Following the standard evaluation protocol, we evaluate our approach on the testing videos. We choose to do our experiments in THUMOS14 instead of other large-scale datasets such as Charades [37] or ActivityNet [6] due to the expensive costs of *re-annotating* such datasets with search sequences. However, we provide an experiment in the **supplementary material** where we evaluate *Action Search* (trained on THUMOS14) on the ActivityNet v1.2 validation videos with the same THUMOS14 classes.

Metric. We compare our model against other methods according to the mean Average Precision (mAP) and penalize duplicate detections. We report the mAP at multiple temporal Intersection-over-Union (tIoU) thresholds.

Implementation Details. We initialize *Action Search* at random temporal locations in the video. We use the Res3D (+ S-CNN) classifier [35,41] as an off-the-shelf pretrained action classifier to classify the N random fixed-length segments, as well as, to classify the temporal action segments generated by our method. After cross-validating on the validation subset, we set $N = 24$, as it achieves the highest average recall while maintaining a small number of segments. Empirically, we find setting $k = 4$ provides the best trade-off between global video classification accuracy and number of search models to run.

Results and Analysis. In Table 1a, we compare our localization approach with state-of-the-art techniques. We assess methods in terms of mAP and the average percentage of observed frames (**S**). Our approach achieves state-of-the-art detection performance while observing much less of the video. For instance, at 0.5 tIoU, we achieve a competitive 30.8% mAP (compared to [16]'s 31.0% mAP) and observe on average 17.3% of each video. To understand the strengths of our method, we break down the results and discuss the following findings:

(1) Our method outperforms its baseline (Res3D + S-CNN [41]) by 8.3% mAP (0.5 tIoU). This baseline uses the same classifier *Action Search* uses, but its action proposals are generated by a CNN. We attribute this improvement to our approach's ability to discard irrelevant portions of the video, which allows the detector to prune false positive detections. This finding indicates the importance of *Action Search* as a precursor to localization.

Table 1. Temporal localization results (mAP at tIoU) on the THUMOS14 testing set. We assign '–' to unavailable mAP values. We report the average percentage of observed frames (**S**) for each approach. (a) Comparison against state-of-the-art methods: Our method (*Action Search + Priors* + Res3D + S-CNN) achieves state-of-the-art results while observing only 17.3% of the video; (b) Video features effect: We compare C3D for *Action Search* visual encoder + the C3D-based classifier from [35] vs. ResNet for *Action Search* visual encoder + the Res3D-based classifier from [41]; (c) The trade-off between *Action Search* training size and performance: mAP and **S** score improve as we increase the training size.

(a)

Method	mAP at tIoU 0.3	0.4	0.5	0.6	0.7	S
Frame Glimpses [44]	36.0	26.4	17.1	–	–	40 *
Shou *et al.* [35]	36.3	28.7	19.0	–	–	100
Shou *et al.* [34]	40.1	29.4	23.3	13.1	7.9	100
Gao *et al.* [17]	44.1	34.9	25.6	–	–	100
Dai *et al.* [11]	–	33.3	25.6	15.9	9.0	100
Xu *et al.* [43]	44.8	35.6	28.9	–	–	100
Buch *et al.* [3]	45.7	–	29.2	–	9.6	100
Zhao *et al.* [46]	**51.9**	41.0	29.8	–	–	100
Gao *et al.* [16]	50.1	41.3	**31.0**	19.1	9.9	100
Res3D + S-CNN [41]	40.6	32.6	22.5	12.3	6.4	100
Our method	51.8	**42.4**	30.8	**20.2**	**11.1**	**17.3**

(b)

Backbone Arch.	mAP at tIoU 0.3	0.4	0.5	0.6	0.7	S
C3D	43.7	35.2	23.9	18.8	4.3	45.1
ResNet	51.8	42.4	30.8	20.2	11.1	17.3

(c)

Training Size	mAP at tIoU 0.3	0.4	0.5	0.6	0.7	S
0% [41]	40.6	32.6	22.5	12.3	6.4	100
25%	41.1	32.9	22.4	12.1	6.4	63.5
50%	47.5	38.3	26.1	16.6	8.1	34.4
75%	50.2	41.0	29.5	18.1	9.4	24.1
100%	51.8	42.4	30.8	20.2	11.1	17.3

*We assume each of the 20 *Frame Glimpses* [44] models observes 2% of the video and report an upper-bound of 40% frames observed.

(2) Our method outperforms *Frame Glimpses* [44] in both mAP and **S**. We observe a 13.7% mAP (0.5 tIoU) improvement and a 22.7% reduction in **S**. We attribute the improvement on **S** to the strategy we follow to train *Action Search*. Instead of relying on a reinforcement policy to explore the video, as *Frame Glimpses* does, we learn to search by imitating humans. Thus, we argue that their search policy is unable to learn temporal context reasoning as well as *Action Search*. This finding justifies the need for the *Human Searches* dataset (Sect. 3) and the supervised strategy we follow to train our model.

(3) Although it uses a naive approach to detect actions, our method surpasses state-of-the-art approaches [3,16,46]. Opting for simplicity, we build our detector by combining *Action Search* with a pretrained off-the-shelf classifier. This straightforward approach allows us to achieve state-of-the-art results while only observing 17.3% of video frames. We argue further improvements can be obtained by combining *Action Search* with sophisticated models such as [16,46].

Ablation Study. We investigate two aspects of our model: **(i)** the backbone architecture for the *Action Search* visual encoder and the off-the-shelf classifier

(refer to Table 1b), and **(ii)** the trade-off between *Action Search* training set size and performance in terms of mAP and **S** (refer to Table 1c).

(i) Table 1b shows using a C3D architecture (*i.e.* C3D for *Action Search* visual encoder + the C3D-based classifier from [35]) improves the performance of the baseline [35] by 4.9% mAP (0.5 tIoU) while observing 54.9% less frames. However, a ResNet backbone architecture gives better results in both mAP and **S**. We attribute this to the facts that ResNet offers a richer feature space, the off-the-shelf classifier from [41] is more sophisticated compared to the classifier from [35], and ResNet uses 1 frame per observation while C3D needs 16 frames.

(ii) *Action Search* is trained on *THUMOS14 searches* dataset, which contains on average 8.8 human searches per video. Training on more human searches would lead to a better performing model, but collecting human searches for *existing* datasets is expensive (although *free* for new datasets). Nonetheless, we observe in Table 1c that training *Action Search* on half of the *THUMOS14 searches* dataset (*i.e.* 4.4 human searches per video), the model improves the detection performance of [41] by 3.6% mAP (0.5 tIoU) while observing only **S** = 34.4% of the frames. Notably, training on as little as 2.2 human searches per video cuts the percentage of observed frames by 36.5% while keeping a similar mAP performance. In general, as we decrease the training set size, the efficiency of the search degrades (**S** increases) since the model is

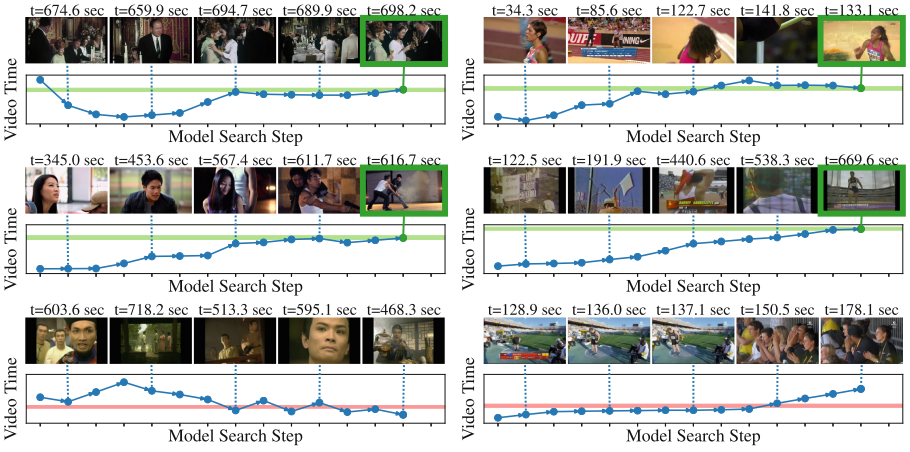

Fig. 5. Qualitative search sequences produced by *Action Search*. The left column corresponds to AVA [21] testing videos, and the right column corresponds to THUMOS14 [24] testing videos. The top two rows depict examples when our model successfully spots the target action location (in green). The last row illustrate failure cases, *i.e.* when the action location (in red) is not spotted exactly. We observe that *Action Search* uses temporal context to reason about where to search next. In failure cases, we notice that our model often oscillates around actions without spotting frames within the exact temporal location. (Color figure online)

not exposed to as many human searches variations. These findings further justify the need for the *Human Searches* dataset and the supervised strategy we follow to train our model.

5.3 Spotting at a Glance

Figure 5 depicts qualitative search sequences of our *Action Search* model, which exploits temporal context to spot the actions quickly. For example, it uses information about scenes/concepts such as gala dinner and fights to spot actions such as *clink glass* and *shoot*, respectively (top two rows, left column). Furthermore, when spotting the start times of actions, our model seems to understand the inherent temporal structures of actions. *Action Search* surprisingly understands when an action finishes and is able to rewind (jump back) and spot the beginning of the action (top row, right column). Typical failure cases (actions are not spotted) occur when the search sequences oscillate around the target action but they miss its exact location (last row, left column). Additionally, *Action Search* may fail when action boundaries are ambiguous. When our model finds content visually similar to the target action, it remains static for a few search steps and then decides to explore the video further (last row, right column).

6 Conclusion

In this paper, we introduced *Action Search*, a new learning model to imitate how humans search for actions in videos, and a new dataset called *Human Searches* to train such model. Extensive experiments demonstrated that *Action Search* produces reliable action detections. We plan to release our *Human Searches* dataset to the vision community and expect that further works can extend the use of search processes for action detection and other applications.

Acknowledgments. This publication is based upon work supported by the King Abdullah University of Science and Technology (KAUST) Office of Sponsored Research (OSR) under Award No. OSR-CRG2017-3405.

References

1. Alahi, A., Goel, K., Ramanathan, V., Robicquet, A., Fei-Fei, L., Savarese, S.: Social LSTM: human trajectory prediction in crowded spaces. In: CVPR (2016)
2. Bengio, S., Vinyals, O., Jaitly, N., Shazeer, N.: Scheduled sampling for sequence prediction with recurrent neural networks. In: NIPS (2015)
3. Buch, S., Escorcia, V., Ghanem, B., Fei-Fei, L., Niebles, J.C.: End-to-end, single-stream temporal action detection in untrimmed videos. In: BMVC (2017)
4. Buch, S., Escorcia, V., Shen, C., Ghanem, B., Carlos Niebles, J.: SST: single-stream temporal action proposals. In: CVPR, July 2017
5. Caba Heilbron, F., Barrios, W., Escorcia, V., Ghanem, B.: SCC: semantic context cascade for efficient action detection. In: CVPR (2017)

6. Caba Heilbron, F., Escorcia, V., Ghanem, B., Carlos Niebles, J.: ActivityNet: a large-scale video benchmark for human activity understanding. In: CVPR (2015)
7. Caba Heilbron, F., Carlos Niebles, J., Ghanem, B.: Fast temporal activity proposals for efficient detection of human actions in untrimmed videos. In: CVPR (2016)
8. Caba Heilbron, F., Thabet, A., Carlos Niebles, J., Ghanem, B.: Camera motion and surrounding scene appearance as context for action recognition. In: Cremers, D., Reid, I., Saito, H., Yang, M.-H. (eds.) ACCV 2014. LNCS, vol. 9006, pp. 583–597. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-16817-3_38
9. Carreira, J., Zisserman, A.: Quo vadis, action recognition? A new model and the kinetics dataset. In: CVPR, July 2017
10. Chen, W., Xiong, C., Xu, R., Corso, J.J.: Actionness ranking with lattice conditional ordinal random fields. In: CVPR (2014)
11. Dai, X., Singh, B., Zhang, G., Davis, L.S., Chen, Y.Q.: Temporal context network for activity localization in videos. In: ICCV (2017)
12. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: ImageNet: a large-scale hierarchical image database. In: CVPR (2009)
13. Duchenne, O., Laptev, I., Sivic, J., Bach, F., Ponce, J.: Automatic annotation of human actions in video. In: ICCV (2009)
14. Escorcia, V., Caba Heilbron, F., Carlos Niebles, J., Ghanem, B.: DAPs: deep action proposals for action understanding. In: ECCV (2016)
15. Gaidon, A., Harchaoui, Z., Schmid, C.: Actom sequence models for efficient action detection. In: CVPR (2011)
16. Gao, J., Yang, Z., Nevatia, R.: Cascaded boundary regression for temporal action detection. In: BMVC (2017)
17. Gao, J., Yang, Z., Sun, C., Chen, K., Nevatia, R.: TURN TAP: temporal unit regression network for temporal action proposals. In: ICCV (2017)
18. Gkioxari, G., Malik, J.: Finding action tubes. In: CVPR (2015)
19. Graves, A.: Generating sequences with recurrent neural networks. arXiv preprint arXiv:1308.0850 (2013)
20. Graves, A., Jaitly, N.: Towards end-to-end speech recognition with recurrent neural networks. In: International Conference on Machine Learning (2014)
21. Gu, C., et al.: AVA: a video dataset of spatio-temporally localized atomic visual actions. In: CVPR (2018)
22. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: CVPR (2016)
23. Jain, M., van Gemert, J.C., Snoek, C.G.: What do 15,000 object categories tell us about classifying and localizing actions? In: CVPR (2015)
24. Jiang, Y.G., et al.: THUMOS challenge: action recognition with a large number of classes (2014). http://crcv.ucf.edu/THUMOS14/
25. Kingma, D.P., Ba, J.: Adam: a method for stochastic optimization. ArXiv e-prints, December 2014
26. Ma, S., Sigal, L., Sclaroff, S.: Learning activity progression in LSTMs for activity detection and early detection. In: CVPR (2016)
27. Mettes, P., van Gemert, J.C., Snoek, C.G.M.: Spot on: action localization from pointly-supervised proposals. In: Leibe, B., Matas, J., Sebe, N., Welling, M. (eds.) ECCV 2016. LNCS, vol. 9909, pp. 437–453. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46454-1_27
28. Carlos Niebles, J., Chen, C.W., Fei-Fei, L.: Modeling temporal structure of decomposable motion segments for activity classification. In: ECCV (2010)
29. Oneata, D., Verbeek, J., Schmid, C.: Efficient action localization with approximately normalized fisher vectors. In: CVPR (2014)

30. Peng, X., Schmid, C.: Multi-region two-stream R-CNN for action detection. In: Leibe, B., Matas, J., Sebe, N., Welling, M. (eds.) ECCV 2016. LNCS, vol. 9908, pp. 744–759. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46493-0_45
31. Pham, V., Bluche, T., Kermorvant, C., Louradour, J.: Dropout improves recurrent neural networks for handwriting recognition. ArXiv e-prints, November 2013
32. Richard, A., Gall, J.: Temporal action detection using a statistical language model. In: CVPR (2016)
33. Saha, S., Singh, G., Sapienza, M., Torr, P.H., Cuzzolin, F.: Deep learning for detecting multiple space-time action tubes in videos. In: BMVC (2016)
34. Shou, Z., Chan, J., Zareian, A., Miyazawa, K., Chang, S.F.: CDC: convolutional-de-convolutional networks for precise temporal action localization in untrimmed videos. In: CVPR (2017)
35. Shou, Z., Wang, D., Chang, S.F.: Temporal action localization in untrimmed videos via multi-stage CNNs. In: CVPR (2016)
36. Sigurdsson, G.A., Russakovsky, O., Gupta, A.: What actions are needed for understanding human actions in videos? In: ICCV, October 2017
37. Sigurdsson, G.A., Varol, G., Wang, X., Farhadi, A., Laptev, I., Gupta, A.: Hollywood in homes: crowdsourcing data collection for activity understanding. In: Leibe, B., Matas, J., Sebe, N., Welling, M. (eds.) ECCV 2016. LNCS, vol. 9905, pp. 510–526. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46448-0_31
38. Simonyan, K., Zisserman, A.: Two-stream convolutional networks for action recognition in videos. In: Advances in Neural Information Processing Systems, pp. 568–576 (2014)
39. Soomro, K., Idrees, H., Shah, M.: Predicting the where and what of actors and actions through online action localization. In: CVPR (2016)
40. Sukhbaatar, S., Weston, J., Fergus, R., et al.: End-to-end memory networks. In: Advances in Neural Information Processing Systems (2015)
41. Tran, D., Ray, J., Shou, Z., Chang, S., Paluri, M.: ConvNet architecture search for spatiotemporal feature learning. CoRR abs/1708.05038 (2017). http://arxiv.org/abs/1708.05038
42. Wang, H., Kläser, A., Schmid, C., Liu, C.L.: Action recognition by dense trajectories. In: CVPR (2011)
43. Xu, H., Das, A., Saenko, K.: R-C3D: region convolutional 3D network for temporal activity detection. In: ICCV (2017)
44. Yeung, S., Russakovsky, O., Mori, G., Fei-Fei, L.: End-to-end learning of action detection from frame glimpses in videos. In: CVPR (2016)
45. Zaremba, W., Sutskever, I., Vinyals, O.: Recurrent neural network regularization. ArXiv e-prints, September 2014
46. Zhao, Y., Xiong, Y., Wang, L., Wu, Z., Lin, D., Tang, X.: Temporal action detection with structured segment networks. In: ICCV (2017)