End-to-end Learning of Action Detection from Frame Glimpses in Videos

Serena Yeung[1], Olga Russakovsky[1,2], Greg Mori[3], Li Fei-Fei[1]

[1]Stanford University, [2]Carnegie Mellon University, [3]Simon Fraser University

serena@cs.stanford.edu, olgarus@cmu.edu, mori@cs.sfu.ca, feifeili@cs.stanford.edu

Abstract

In this work we introduce a fully end-to-end approach for action detection in videos that learns to directly predict the temporal bounds of actions. Our intuition is that the process of detecting actions is naturally one of observation and refinement: observing moments in video, and refining hypotheses about when an action is occurring. Based on this insight, we formulate our model as a recurrent neural network-based agent that interacts with a video over time. The agent observes video frames and decides both where to look next and when to emit a prediction. Since backpropagation is not adequate in this non-differentiable setting, we use REINFORCE to learn the agent's decision policy. Our model achieves state-of-the-art results on the THUMOS'14 and ActivityNet datasets while observing only a fraction (2% or less) of the video frames.

1. Introduction

Action detection in long, real-world videos is a challenging problem in computer vision. Algorithms must reason not only about whether an action occurs somewhere in a video, but also on the temporal extent of *when* it occurs. Most existing work [22, 39, 13, 46] take the approach of building frame-level classifiers, running them exhaustively over a video at multiple temporal scales, and applying post-processing such as duration priors and non-maximum suppression. However, this indirect modeling of action localization is unsatisfying in terms of both accuracy as well as computational efficiency.

In this work, we introduce an end-to-end approach to action detection that reasons directly on the temporal bounds of actions. Our key intuition (Fig. 1) is that the process of detecting an action is one of continuous, iterative observation and refinement. Given a single or a few frame observations, a human can already formulate hypotheses about when an action may occur. We can then skip ahead or back some frames to verify, and quickly narrow down the action location (e.g. swinging a baseball bat in Fig. 1). We are able to sequentially decide where to look and how to refine our
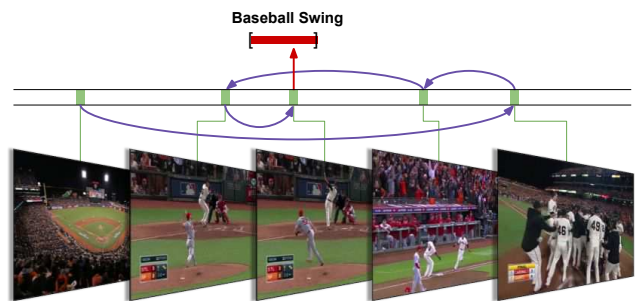

Figure 1: Action detection is a process of observation and refinement. Effectively choosing a sequence of frame observations allows us to quickly narrow down when the baseball swing occurs.

hypotheses to obtain precise localization of the action with far less exhaustive search compared to existing algorithms.

Based on this intuition, we present *a single coherent model* that takes a long video as input, and outputs the temporal bounds of detected action instances. Our model is formulated as an agent that learns a policy for sequentially forming and refining hypotheses about action instances. Casting this into a recurrent neural network-based architecture, we train the model in a fully end-to-end fashion using a combination of backpropagation and REINFORCE [42].

Our model draws inspiration from works that have used REINFORCE to learn spatial glimpse policies for image classification and captioning [19, 1, 30, 43]. However, action detection presents the additional challenge of how to handle a variable-sized set of structured detection outputs. *To address this, we present a model that decides both which frame to observe next as well as when to emit a prediction, and we introduce a reward mechanism that enables learning this policy.* To the best of our knowledge, this is the first end-to-end approach for learning to detect actions in video.

We show that our model is able to reason effectively on the temporal bounds of actions, and achieve state-of-the-art performance on the THUMOS'14 [11] and ActivityNet [3] datasets. Moreover, because it learns policies for which frames to observe, or temporally glimpse, it is able to do so while observing only a fraction (2% or less) of the frames.

2. Related Work

There is a long history of work in video analysis and activity recognition [20, 49, 2, 31, 17, 8, 10, 12, 50]. For a survey we refer to Poppe [24] and Weinland et al. [40]. Here we review recent work relevant to temporal action detection.

Temporal action detection. Canonical work in this vein is Ke et al. [14]. Rohrbach et al. [27] and Ni et al. [21] use hand-centric and object-centric features, respectively, to detect fine-grained cooking actions in a fixed-camera kitchen setting. More related to our work is the unconstrained and untrimmed setting of the THUMOS'14 action detection dataset. Oneata et al. [22], Wang et al. [39], Karaman et al. [13], and Yuan et al. [46] use fusions of dense trajectories, frame-level CNN features, and/or sound features in a sliding window framework to perform temporal action detection. Sun et al. [34] uses web images as a prior to improve detection performance. Pirsiavash and Ramanan [23] build grammars over complex actions and additionally detect sub-components in time.

Methods for spatio-temporal action detection have also been developed. Within the context of "unconstrained" internet videos, this includes a body of work on spatio-temporal action proposals [44, 16, 36, 9, 7, 45, 41]. Analysis of broader surveillance scenes for action detection is also an active area of research. Shu et al. [32] reason about groups of people, Loy et al. [18] across multi-camera setups, and Kwak et al. [15] based on quadratic programming-based instantiations of rules. Common among these works is reasoning on spatio-temporal action proposals or human tracks, typically using sliding window-based approaches in the temporal dimension. Furthermore, these works are in the context of trimmed or constrained-setting video clips. In contrast, we address the task of temporal action detection in untrimmed, unconstrained videos, with an efficient method for determining which frames to examine.

End-to-end detection. Our goal of directly reasoning on the temporal bounds of actions shares philosophy with work in object detection that has regressed from full images to object bounds [29, 35, 5, 6, 26, 25]. In contrast, existing action detection methods typically use exhaustive sliding-window approaches and post-processing to produce action instances [22, 39, 13, 46]. To the best of our knowledge, our work is the first to address learning of temporal action detection in an end-to-end framework.

Learning task-specific policies. We draw inspiration from recent approaches that have used REINFORCE [42] to learn task-specific policies. Mnih et al. [19], Ba et al. [1], and Sermanet et al. [30] learn spatial attention policies for image classification, and Xu et al. [43] for image caption generation. In a non-visual task, Zaremba et al. [47] learn policies for a Reinforcement Learning Neural Turing Machine. Our method builds on these directions and uses REINFORCE to learn policies addressing the task of action detection.

3. Method

Our goal is to take a long sequence of video and output any instances of a given action. Fig. 2 shows our model structure. The model is formulated as a reinforcement learning agent that interacts with a video over time. The agent receives a sequence of video frames $\mathbf{V} = \{v_1, ..., v_T\}$ as input, and can observe a fixed proportion of the frames. It must learn to effectively utilize these observations, or frame glimpses, to reason on the temporal bounds of actions.

3.1. Architecture

The model consists of two main components: an observation network (Sec. 3.1.1), and a recurrent network (Sec. 3.1.2). The *observation network* encodes visual representations of video frames. The *recurrent network* sequentially processes these observations and decides both which frame to observe next and when to emit a prediction. We now describe each of these in more detail. Later in Sec. 3.2, we explain how we use a combination of backpropagation and REINFORCE to train the model in end-to-end fashion.

3.1.1 Observation Network

As shown in Fig. 2, the observation network f_o, parameterized by θ_o, observes a single video frame at each timestep. It encodes the frame into a feature vector o_n and provides this as input to the recurrent network.

Importantly, o_n encodes information about both *where* in the video an observation was taken as well as *what* was seen. The inputs to the observation network therefore consist of the normalized temporal location of the observation, $l_n \in [0, 1]$, and the corresponding video frame v_{l_n}.

The architecture of the observation network is inspired by the spatial glimpse network of [19]. Both l_n and v_{l_n} are mapped to a hidden space and then combined with a fully connected layer. v_{l_n} is typically mapped with a sequence of convolutional, pooling, and fully connected layers; in our experiments we extract fc7 features from a fine-tuned VGG-16 network [33] and use $o_n \in \mathbb{R}^{1024}$.

3.1.2 Recurrent Network

The recurrent network f_h, parameterized by θ_h, forms the core of the learning agent. As can be seen in Fig. 2, the input to the network at each timestep n is observation feature vector o_n. The network's hidden state h_n, a function of both o_n and the previous hidden state h_{n-1}, models temporal hypotheses about action instances.

As the agent reasons on a video, three outputs are produced at each timestep: candidate detection d_n, binary indicator p_n signaling whether to emit d_n as a prediction, and temporal location l_{n+1} indicating the frame to observe next. We now describe each of these in more detail.
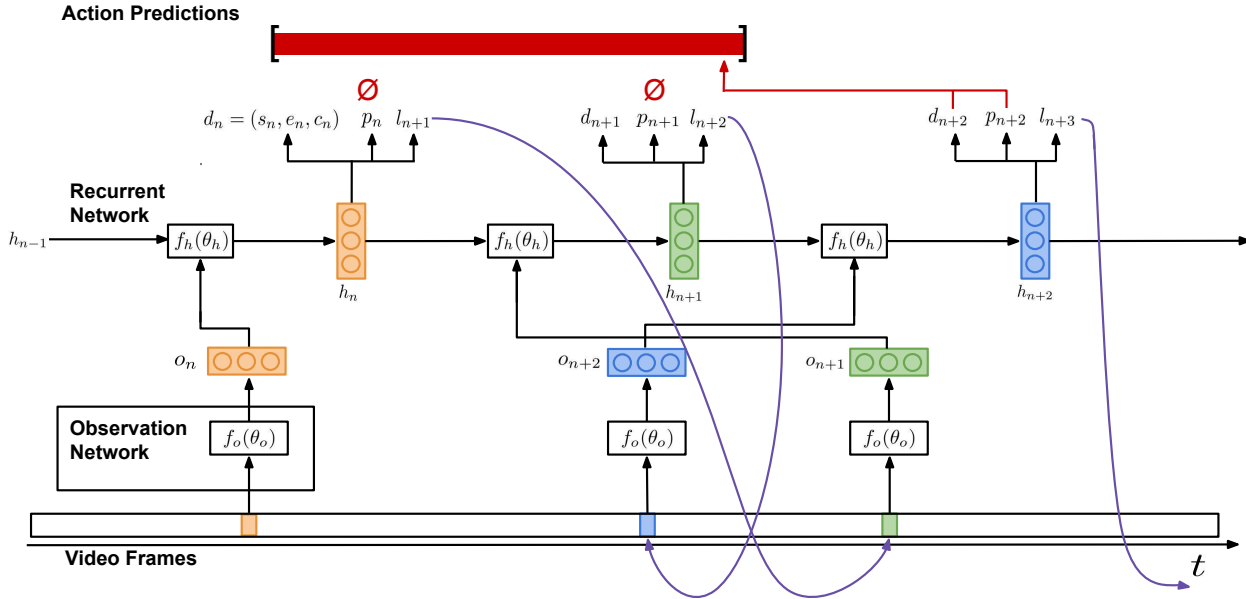
Figure 2: The input to the model is a sequence of video frames, and the output is a set of action predictions. We illustrate an example of a forward pass. At timestep n, the agent observes the orange video frame and produces candidate detection d_n, however prediction indicator output p_n suppresses it from being emitted into the prediction set. Observation location output l_{n+1} signals to observe the the green video frame at the next timestep. The process repeats, and here again p_{n+1} suppresses d_{n+1} from being emitted. l_{n+2} signals to now go backwards in the video to observe the blue frame. At timestep $n + 2$, the action hypothesis is sufficiently refined, and the agent uses prediction indicator p_{n+2} to emit d_{n+2} into the prediction set (red arrow). The agent then continues proceeding through the video.

Candidate detection. A candidate detection d_n is produced using the function $d_n = f_d(h_n; \theta_d)$, where f_d is a fully connected layer. d_n is a tuple $(s_n, e_n, c_n) \in [0, 1]^3$, where s_n and e_n are the normalized start and end locations of the detection, and c_n is the confidence level of the detection. This candidate detection represents the agent's hypothesis surrounding a current action instance. However, it is not emitted as a *prediction* at each timestep, which would lead to a large amount of noise and many false positives. Instead, the agent uses a separate prediction indicator output to signal when a candidate detection should be emitted as a prediction.

Prediction indicator. The binary prediction indicator p_n signals whether corresponding candidate detection d_n should be emitted as a prediction. $p_n = f_p(h_n; \theta_p)$, where f_p is a fully connected layer followed by a sigmoid non-linearity. At training time, f_p is used to parameterize a Bernoulli distribution from which p_n is sampled; at test time, the maximum a posteriori estimate is used.

The combination of the candidate detection and prediction indicator is crucial for the detection problem, where positive instances may occur anywhere or not at all. It enables the network to indicate when it has identified a unique action instance to add to the prediction set, and essentially folds non-maximum suppression in as a learnable component of our end-to-end framework.

Location of next observation. The temporal location $l_{n+1} \in [0, 1]$ indicates the video frame that the agent chooses to observe next. This location is not constrained, and the agent may skip both forwards and backwards around a video.

The location is computed as $l_{n+1} = f_l(h_n; \theta_l)$, where f_l is a fully connected layer, such that the agent's decision is a function of its past observations and their temporal locations. At training time, l_{n+1} is sampled from a Gaussian distribution with a mean of $f_l(h_n; \theta_l)$ and a fixed variance; at test time, the maximum a posteriori estimate is used.

Fig. 2 further illustrates the roles of these outputs and their interaction with an example of a forward pass through the network.

3.2. Training

Our end goal is to learn to output a set of detected actions. To achieve this, we need to train the three outputs at each step of the agent's recurrent network: candidate detection d_n, prediction indicator p_n, and next observation location l_{n+1}. Given supervision from temporal action annotations in long videos, training these involves challenges of designing suitable loss and reward functions, and handling non-differentiable model components. We now explain how we address these challenges. We use standard backpropagation to train d_n, and REINFORCE to train p_n and l_{n+1}.

3.2.1 Candidate detections

Candidate detections are trained using backpropagation to maximize the correctness of each candidate. We wish to maximize correctness regardless of whether a candidate is ultimately emitted, since the candidates encode the agent's hypotheses about actions. This requires matching each candidate with a ground truth instance during training. We use the insight that at each timestep, the agent should form a hypothesis around the action instance (if any) nearest its current location in the video. This enables us to design a simple yet effective matching function.

Matching to ground truth. Given a set of candidate detections $D = \{d_n | n = 1, ..., N\}$ produced by a recurrent network of N timesteps, and given ground truth action instances $g_{1,...,M}$, each candidate is matched to one ground truth instance, or none if $M = 0$.

We define matching function

$$y_{nm} = \begin{cases} 1 & \text{if } m = \arg\min_{j=1,...,M} dist(l_n, g_j) \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

In other words, candidate d_n is matched to ground truth g_m if the agent's temporal location l_n at timestep n is closer to g_m than any other ground truth instance. Defining $g_m = (s_m, e_m)$ as the start and end location of a ground truth instance, $dist(l_n, g_m)$ is simply $\min(|s_m - l_n|, |e_m - l_n|)$.

Loss function. Once candidate detections have been matched to ground truth instances, we optimize a multi-task classification and localization loss function over the set D:

$$L(D) = \sum_n L_{cls}(d_n) + \gamma \sum_n \sum_m \mathbb{1}[y_{nm} = 1]L_{loc}(d_n, g_m) \quad (2)$$

Here the classification term $L_{cls}(d_n)$ is a standard cross-entropy loss on the detection confidence c_n, encouraging the confidence to be closer to 1 if detection d_n is matched to a ground truth instance, and 0 otherwise.

If the detection is matched to a ground truth g_m (i.e. $y_{nm} = 1$), the localization term $L_{loc}(d_n, g_m)$ is an L_2-regression loss that further encourages minimizing the distance $\|(s_n, e_n) - (s_m, e_m)\|$ between the two segments.

We optimize this loss function using backpropagation.

3.2.2 Observation and emission sequences

The observation location and prediction indicator outputs are non-differentiable components of our model that cannnot be trained with standard backpropagation. However, REINFORCE [42] is a powerful approach that enables learning in non-differentiable settings. We first briefly describe REINFORCE below. We then introduce a reward function that we use with REINFORCE to learn effective policies for observation and prediction emission sequences.

REINFORCE. Given $\mathcal{A}$, a space of action sequences, and $p_\theta(a)$, a distribution over $a \in \mathcal{A}$ and parameterized by θ, the REINFORCE objective can be expressed as

$$J(\theta) = \sum_{a \in \mathcal{A}} p_\theta(a)r(a) \quad (3)$$

Here $r(a)$ is a reward assigned to each possible action sequence, and $J(\theta)$ is the expected reward under the distribution of possible action sequences. In our case we wish to learn network parameters θ that maximize the expected reward of a sequence of location and prediction indicator outputs.

The gradient of the objective is

$$\nabla J(\theta) = \sum_{a \in \mathcal{A}} p_\theta(a)\nabla \log p_\theta(a)r(a) \quad (4)$$

This leads to a non-trivial optimization problem due to the high-dimensional space of possible action sequences. REINFORCE addresses this by learning network parameters using Monte Carlo sampling and an approximation to the gradient equation:

$$\nabla J(\theta) \approx \frac{1}{K} \sum_{i=1}^{K} \sum_{n=1}^{N} \nabla \log \pi_\theta(a_n^i | h_{1:n}^i, a_{1:n-1}^i)R_n^i \quad (5)$$

Given an agent interacting with an environment, in our case a video, π_θ is the agent's policy. This is a learned distribution over actions conditioned on the interaction sequence thus far. At timestep n, a_n is the policy's current action (e.g. location l_{n+1} or prediction indicator p_n), $h_{1:n}$ is the history of past states including the current, and $a_{1:n-1}$ is the history of past actions. $R_n = \sum_{t=n}^{N} r_t$ is the cumulative future reward obtained from the current timestep onward, for a sequence of N timesteps. The approximate gradient is computed by running an agent's current policy in its environment to obtain K interaction sequences.

To reduce the variance of the gradient estimate, a baseline reward b_n^i is often estimated, e.g. via a separate network, and subtracted so that the gradient equation becomes:

$$\nabla J(\theta) \approx \frac{1}{K} \sum_{i=1}^{K} \sum_{n=1}^{N} \nabla \log \pi_\theta(a_n^i | h_{1:n}^i, a_{1:n-1}^i)(R_n^i - b_n^i) \quad (6)$$

REINFORCE learns model parameters according to this approximate gradient. The log-probability $\log \pi_\theta$ of actions leading to high future reward are increased, and those leading to low reward are decreased. Model parameters can then be updated using backpropagation.

Reward function. Training with REINFORCE requires designing an appropriate reward function. Our goal is to learn policies for the location and prediction indicator outputs that lead to action detection with both high recall and high precision. We therefore introduce a reward function

	α=0.5	α=0.4	α=0.3	α=0.2	α=0.1
Karaman et al. [13]	0.9	1.4	2.1	3.4	4.6
Wang et al. [39]	8.3	11.7	14.0	17.0	18.2
Oneata et al. [22]	14.4	20.8	27.0	33.6	36.6
Ours (full)	**17.1**	**26.4**	**36.0**	**44.0**	**48.9**
Ablation Experiments					
Ours w/o d_{pred}	12.4	19.3	26.0	32.5	37.0
Ours w/o d_{obs}	9.3	15.2	20.6	26.5	31.2
Ours w/o d_{obs} w/o d_{pred}	8.6	14.6	20.0	27.1	33.3
Ours w/o loc	5.5	9.9	16.2	22.7	27.5
CNN with NMS	6.4	9.6	12.8	16.7	18.5
LSTM with NMS	5.6	7.8	10.3	13.9	15.7

Table 1: Action detection results on THUMOS'14. Comparison with the top 3 performers on the THUMOS'14 challenge leaderboard is shown, as well as with ablation models. mAP is reported for different intersection-over-union (IOU) thresholds α.

that seeks to maximize true positive detections while minimizing false positives:

$$r_N = \begin{cases} R_p & \text{if } M > 0 \text{ and } N_p = 0 \\ N_+ R_+ + N_- R_- & \text{otherwise} \end{cases} \quad (7)$$

All reward is provided at the Nth (final) timestep, and is 0 for $n < N$, since we want to learn policies that jointly lead to high overall detection performance. M is the number of ground truth action instances, and N_p is the number of predictions emitted by the agent. N_+ is the number of true positive predictions, N_- is the number of false positive predictions, and R_+ and R_- are positive and negative rewards contributed by each of these predictions, respectively. A prediction is considered correct if its overlap with a ground truth is both greater than a threshold and higher than that of any other prediction. In order to encourage the agent not to be overly conservative, a negative reward R_p is provided if the the video contains ground truth instances ($M > 0$) but the model did not emit any predictions ($N_p = 0$).

We use this function with REINFORCE to train the location and prediction indicator outputs, and learn observation and emission policies optimized for action detection.

4. Experiments

We evaluate our model on two datasets - THU-MOS'14 [11] and ActivityNet [3]. We show that our end-to-end approach enables the model to outperform state-of-the-art results by a large margin on both datasets. Furthermore, the learned policy of frame observations is both effective and efficient; the model achieves these results while observing in total only 2% or less of the video frames.

4.1. Implementation Details

We learn a 1-vs-all model for each action class. In the observation network, we use a VGG-16 network [33] fine-tuned on the dataset to extract visual features from observed video frames. Fc7-layer features are extracted and embedded with the temporal location of the frame into a 1024-dimensional observation vector.

For the recurrent network, we use a 3-layer LSTM network with 1024 hidden units in each layer. Videos are downsampled to 5fps in THUMOS'14 and 1fps in ActivityNet, and processed in sequences of 50 frames. The agent is given a fixed number of observations for each sequence, typically 6 in our experiments. All temporal locations are normalized to $[0, 1]$ in a video sequence. Any predictions overlapping or crossing sequence bounds are merged with a simple union rule. We learn with mini-batches of 256 sequences, and use RMSProp [37] to modulate the per-parameter learning rate during optimization. Other hyper-parameters are learned through cross-validation. The ratio of sequences containing positive examples in each mini-batch is an important hyperparameter to prevent the model from being overly conservative. Approximately one-third to one-half positive examples is typically used.

4.2. THUMOS'14 Dataset

The action detection task of THUMOS'14 [11] consists of 20 classes of sports, and Table 1 shows results on this dataset. Since the task comprises only 20 of the 101 action classes in the dataset, we first coarsely filter the full set of test videos for these classes, using video-level average pooling over class probabilities that are computed every 300 frames (0.1 fps). We report mAP for different IOU thresholds α, and compare with the top 3 performers on the THUMOS'14 challenge leaderboard [11]. All these methods compute dense trajectories [38] and/or CNN features over temporal windows, and use a sliding window approach with non-maximum suppression to obtain predictions. [13] uses dense trajectories only, [39] uses temporal windows of combined dense trajectories and CNN features, and [22] uses temporal windows of dense trajectories with video-level CNN classification predictions.
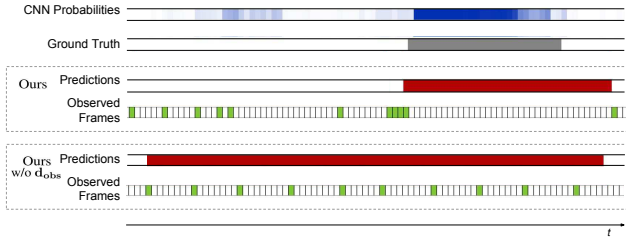
Figure 3: Comparison of Our full model with the Ours w/o d_{obs} model. Refer to Fig. 5 caption for explanation of figure structure and color scheme. Each model's observed frames are shown in green, and the prediction extent in red. Allowing the model to choose which frames to observe enables the necessary resolution to reason precisely on action bounds.

Our model outperforms existing methods at all values of α. The relative margin increases as we decrease α, indicating that our model more frequently predicts actions near ground truth instances even when not precisely localized. Our model achieves these results while processing only 2% of videos frames using its learned observation policy.

Ablation experiments. Table 1 also shows results for ablation experiments analyzing the contributions of different model components. The ablation models are as follows:

- **Ours w/o d_{pred}** removes the prediction indicator output. The candidate detection at every timestep is emitted, and merged with non-maximum suppression.
- **Ours w/o d_{obs}** removes the location output indicating where to observe next. Observations are instead determined by uniform sampling with the same total number of observations.
- **Ours w/o d_{obs} w/o d_{pred}** removes both the prediction indicator and location observation outputs.
- **Ours w/o loc** removes localization regression. All emitted detections are of median length from the training set, and centered on the currently observed frame.
- **CNN with NMS** removes direct prediction of temporal action bounds. Per-frame class probabilities from the VGG-16 Network [33] used in our observation network are densely obtained at multiple temporal scales and aggregated with non-maximum suppression, similar to existing work [13, 39, 22].

Ours w/o d_{pred} obtains lower performance compared to the full model, due to many false positives. Ours w/o d_{obs} also lowers performance since uniform sampling does not provide sufficient resolution to localize action boundaries (Fig. 3). Interestingly, removing d_{obs} cripples the model more than removing d_{pred}, highlighting the importance of the observation policy. As expected, removing both outputs in Ours w/o d_{obs} w/o d_{pred} decreases performance further. Ours w/o loc is the poorest performing model at $\alpha = 0.5$, even below the CNN, showing the importance of temporal regression. The relative difference with the CNN decreases and then flips when we decrease α, indicating that

	[22]	Ours		[22]	Ours
Baseball Pitch	8.6	**14.6**	Hamm. Throw	**34.7**	28.9
Basket. Dunk	1.0	**6.3**	High Jump	17.6	**33.3**
Billiards	2.6	**9.4**	Javelin Throw	**22.0**	20.4
Clean and Jerk	13.3	**42.8**	Long Jump	**47.6**	39.0
Cliff Diving	**17.7**	15.6	Pole Vault	**19.6**	16.3
Cricket Bowl.	9.5	**10.8**	Shotput	11.9	**16.6**
Cricket Shot	2.6	**3.5**	Soccer Penalty	**8.7**	8.3
Diving	4.6	**10.8**	Tennis Swing	3.0	**5.6**
Frisbee Catch	1.2	**10.4**	Throw Discus	**36.2**	29.5
Golf Swing	**22.6**	13.8	Volley. Spike	1.4	**5.2**
mAP				14.4	**17.1**

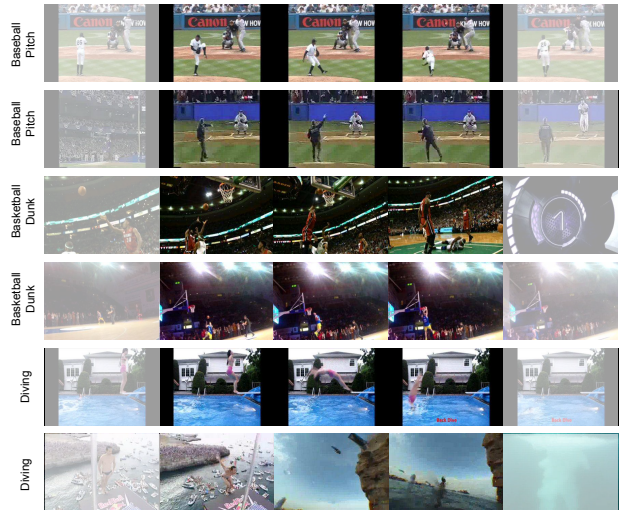Table 2: Per-class breakdown (AP) on THUMOS'14, at IOU of $\alpha = 0.5$.

Figure 4: Examples of predicted action instances on THUMOS'14. Each row shows sampled frames during or just outside the temporal extent of a detected action. Faded frames indicate location outside the detection and illustrate localization ability.

the model still detects the rough location of actions but suffers from imprecise localization. Finally, the CNN with NMS achieves significantly lower performance than all ablation models except the Ours w/o loc model, quantifying the contribution of our end-to-end framework. Its performance is also in the range of but lower than [39], which uses dense trajectories [38] and Imagenet-pretrained [28] CNN features. This suggests that additionally incorporating motion-based features would further improve the performance of our model.

As an additional baseline, we perform NMS on top of an LSTM, a standard temporal network which produces frame-level smoothing and consistency [4]. The LSTM with NMS achieves lower performance than the CNN with NMS, despite adding greater temporal consistency. The main reason appears to be that increasing the temporal smoothness of frame-level class probabilities is actually harmful, not beneficial, to the task of action instance detection, where precise localization of temporal boundaries is required.
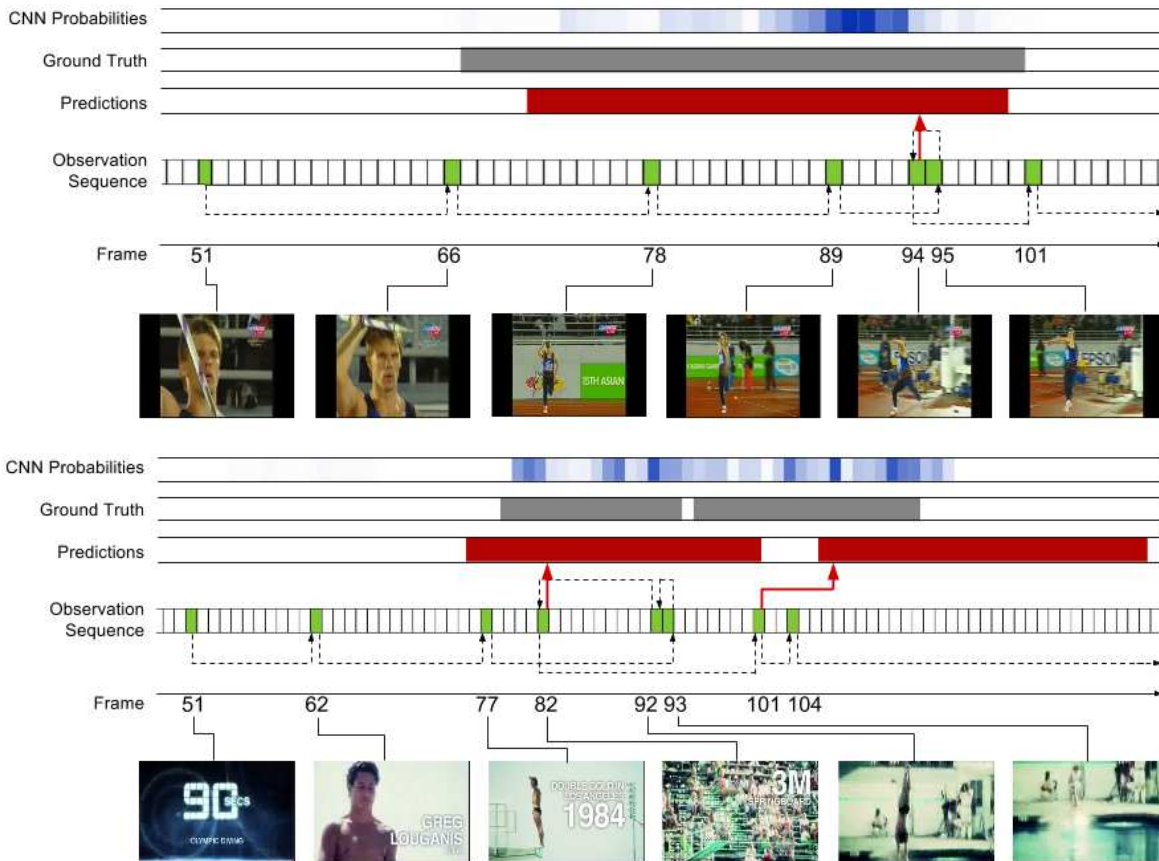
Figure 5: Examples of learned observation policies on THUMOS'14. The top example shows a javelin throw and the bottom example shows diving. Observed frames are colored in green and labeled with the frame index. Prediction extents are shown in red, and ground truth in grey. For reference, we also show frame-level CNN probabilities from the VGGNet used in our observation network; higher intensity indicates higher probability and provides insight into frame-level signal for the class. Dashed arrows indicate the observation sequence, and red arrows indicate frames where a prediction was emitted.

Finally, we experimented with different numbers of observations per video sequence, e.g. 4, 8, and 10. Detection performance was not substantially different across this range. This is consistent with other work on CNNs for action recognition using max-pooling [48], highlighting the importance of learning effective frame observation policies. **Per-class breakdown.** Table 2 shows the per-class AP breakdown of our model, and comparison with the top performer [22] on the THUMOS'14 leaderboard. Our model outperforms [22] on 12 out of 20 classes. Notably, it shows significant improvement on some of the most challenging classes in the dataset such as basketball dunk, diving, and frisbee catch. Fig. 4 shows examples of our model's predictions, including several from these challenging classes. The model's ability to reason holistically on action extents enables it to infer temporal boundaries even when frame appearance is challenging: e.g. similar pose and environment, or abrupt scene change in the second diving example. **Observation policy analysis.** Fig. 5 shows examples of ob-

servation policies that our model learns, as well as accompanying predictions. For reference, we also show frame-level CNN probabilities from the VGGNet used in our observation network, to provide insight into frame-level signal for the action. In the top example of a javelin throw, the model begins to take more frequent observations once the person begins running. Near the end boundary of the action, it takes a step backwards to refine its hypothesis and then emits a prediction before moving on.

The lower example of diving is a challenging case where two action instances occur in quick succession. While the strength of the frame-level CNN probabilities over the sequence would be difficult for standard sliding-window approaches to handle, our model is able to discern the two separate instances. The model once again takes steps backwards to refine its prediction, including once (frame 93) when motion blur makes it difficult to discern much from the frame. However, the predictions are also somewhat longer than the ground truth, and upon observing its first
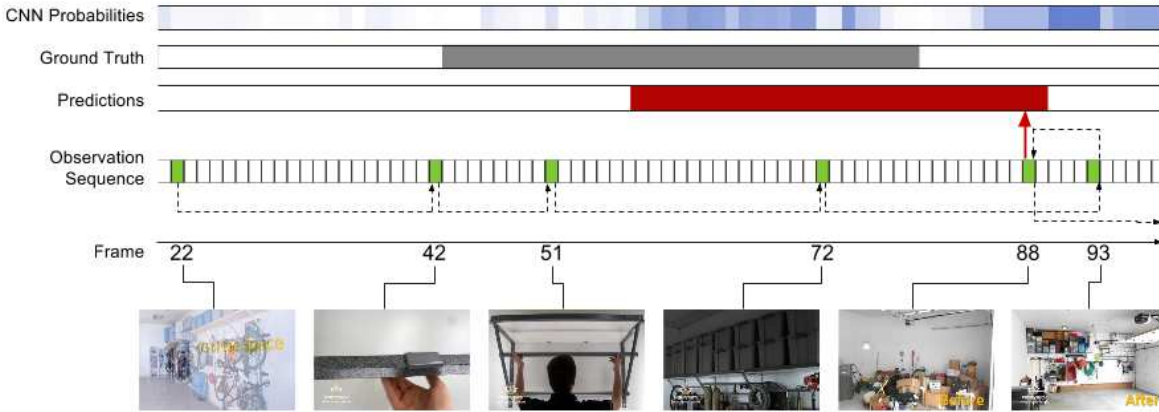
Figure 6: Example of a learned observation policy on the Work subset of ActivityNet. The action is Organizing Boxes. Refer to Fig. 5 for explanation of figure structure and color scheme.

frame of the second instance (frame 101), the model immediately emits a prediction of comparable but slightly longer duration than the first. This suggests that the model may have learned duration priors that, while generally beneficial, were overly strong in this case.

4.3. ActivityNet Dataset

The ActivityNet action detection dataset [3] consists of 68.8 hours of temporal annotations in 849 hours of untrimmed, unconstrained video. There are 1.41 action instances per video and 193 instances per class. Tables 3 and 4 show per-class and mAP performance on the "Playing sports" and "Work, main job" subsets of ActivityNet, respectively. Evaluation uses the ActivityNet validation set, and hyperparameters are cross-validated on the training set.

Our model outperforms existing work [3], which is based on a combination of dense trajectories, SIFT, and ImageNet-pretrained CNN features, by significant margins. It outperforms [3] in 13 out of 21 classes on the Sports subset and in 10 out of 15 classes on the Work subset. The improvement is particularly large on the Work subset. This is partially attributable to the fact that work activities are generally less well-defined and have less discriminative movements. In the example sequence of the Organizing Boxes action in Fig. 6, this is evident in the weaker, more diffuse frame-level CNN probabilities for the action. While this creates a challenge for approaches that rely on post-processing, our model's direct reasoning on action extents enables it to still produce strong predictions.

5. Conclusion

In conclusion, we have introduced an end-to-end approach for action detection in videos that learns to directly predict the temporal bounds of actions. Our model achieves state-of-the-art results on the THUMOS'14 and ActivityNet

	[3]	Ours		[3]	Ours
Archery	**34.7**	5.2	Long Jump	41.1	**56.8**
Bowling	51.3	**52.2**	Mountain Climb.	31.0	**53.0**
Bungee	42.6	**48.9**	Paintball	**31.2**	12.5
Cricket	27.9	**38.4**	Playing Kickball	33.8	**60.8**
Curling	16.4	**30.1**	Playing Volley.	32.1	**40.2**
Discus Throw	**26.2**	17.6	Pole Vault	**47.7**	35.5
Dodgeball	26.6	**61.3**	Shot put	29.4	**50.9**
Doing Moto.	30.2	**46.2**	Skateboarding	21.3	**34.4**
Ham. Throw	**22.2**	13.7	Start Fire	25.3	**38.4**
High Jump	**41.3**	21.9	Triple Jump	**36.4**	16.1
Javelin Throw	**48.1**	35.7			
mAP				33.2	**36.7**

Table 3: Per-class breakdown and mAP on the ActivityNet Sports subset, at IOU of $\alpha = 0.5$.

	[3]	Ours		[3]	Ours
Attend Conf.	28.3	**56.5**	Phoning	34.7	**52.1**
Search Security	24.5	**33.9**	Pumping Gas	**54.7**	34.0
Buy Fast Food	34.4	**45.8**	Setup Comp.	**37.4**	30.3
Clean Laptop Fan	26.0	**35.8**	Sharp. Knife	**36.3**	35.2
Making Copies	18.2	**41.7**	Sort Books	**29.3**	16.7
Organizing Boxes	**29.6**	19.1	Using Comp.	37.4	**50.2**
Organiz. Cabin.	19.0	**43.7**	Using ATM	29.5	**64.9**
Packing	28.0	**39.1**			
mAP				31.1	**39.9**

Table 4: Per-class breakdown and mAP on the ActivityNet Work subset, at IOU of $\alpha = 0.5$.

action detection datasets while observing only a fraction of frames. A direction for future work is to extend our framework to learn joint spatio-temporal observation policies.

6. Acknowledgments

References

[1] J. Ba, V. Mnih, and K. Kavukcuoglu. Multiple object recognition with visual attention. *arXiv preprint arXiv:1412.7755*, 2014. 1, 2

[2] M. Blank, L. Gorelick, E. Shechtman, M. Irani, and R. Basri. Actions as space-time shapes. In *Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on*, volume 2, pages 1395–1402. IEEE, 2005. 2

[3] F. Caba Heilbron, V. Escorcia, B. Ghanem, and J. Carlos Niebles. Activitynet: A large-scale video benchmark for human activity understanding. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 961–970, 2015. 1, 5, 8

[4] J. Donahue, L. A. Hendricks, S. Guadarrama, M. Rohrbach, S. Venugopalan, K. Saenko, and T. Darrell. Long-term recurrent convolutional networks for visual recognition and description. *arXiv preprint arXiv:1411.4389*, 2014. 6

[5] D. Erhan, C. Szegedy, A. Toshev, and D. Anguelov. Scalable object detection using deep neural networks. In *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, pages 2155–2162. IEEE, 2014. 2

[6] R. Girshick. Fast r-cnn. *arXiv preprint arXiv:1504.08083*, 2015. 2

[7] G. Gkioxari and J. Malik. Finding action tubes. *arXiv preprint arXiv:1411.6031*, 2014. 2

[8] A. Gupta, P. Srinivasan, J. Shi, and L. S. Davis. Understanding videos, constructing plots learning a visually grounded storyline model from annotated videos. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 2012–2019. IEEE, 2009. 2

[9] M. Jain, J. Van Gemert, H. Jégou, P. Bouthemy, and C. G. Snoek. Action localization with tubelets from motion. In *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, pages 740–747. IEEE, 2014. 2

[10] H. Jhuang, J. Gall, S. Zuffi, C. Schmid, and M. J. Black. Towards understanding action recognition. In *Computer Vision (ICCV), 2013 IEEE International Conference on*, pages 3192–3199. IEEE, 2013. 2

[11] Y.-G. Jiang, J. Liu, A. Roshan Zamir, G. Toderici, I. Laptev, M. Shah, and R. Sukthankar. THUMOS challenge: Action recognition with a large number of classes. http://crcv.ucf.edu/THUMOS14/, 2014. 1, 5

[12] V. Kantorov and I. Laptev. Efficient feature extraction, encoding, and classification for action recognition. In *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, pages 2593–2600. IEEE, 2014. 2

[13] S. Karaman, L. Seidenari, and A. Del Bimbo. Fast saliency based pooling of fisher encoded dense trajectories. 1, 2, 5, 6

[14] Y. Ke, R. Sukthankar, and M. Hebert. Event detection in crowded videos. In *ICCV*, 2007. 2

[15] S. Kwak, B. Han, and J. H. Han. Multi-agent event detection: Localization and role assignment. In *CVPR*, 2013. 2

[16] T. Lan, Y. Wang, and G. Mori. Discriminative figure-centric models for joint action localization and recognition. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 2003–2010. IEEE, 2011. 2

[17] I. Laptev, M. Marszałek, C. Schmid, and B. Rozenfeld. Learning realistic human actions from movies. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8. IEEE, 2008. 2

[18] C. C. Loy, T. Xiang, and S. Gong. Incremental activity modeling in multiple disjoint cameras. *TPAMI*, 34(9):1799–1813, 2012. 2

[19] V. Mnih, N. Heess, A. Graves, et al. Recurrent models of visual attention. In *Advances in Neural Information Processing Systems*, pages 2204–2212, 2014. 1, 2

[20] D. Moore and I. Essa. Recognizing multitasked activities from video using stochastic context-free grammar. 2002. 2

[21] B. Ni, V. R. Paramathayalan, and P. Moulin. Multiple granularity analysis for fine-grained action detection. In *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, pages 756–763. IEEE, 2014. 2

[22] D. Oneata, J. Verbeek, and C. Schmid. The lear submission at thumos 2014. 2014. 1, 2, 5, 6, 7

[23] H. Pirsiavash and D. Ramanan. Parsing videos of actions with segmental grammars. In *Computer Vision and Pattern Recognition (CVPR)*, 2014. 2

[24] R. Poppe. A survey on vision-based human action recognition. *IVC*, 28:976–990, 2010. 2

[25] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. *arXiv preprint arXiv:1506.02640*, 2015. 2

[26] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *arXiv preprint arXiv:1506.01497*, 2015. 2

[27] M. Rohrbach, S. Amin, M. Andriluka, and B. Schiele. A database for fine grained activity detection of cooking activities. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 1194–1201. IEEE, 2012. 2

[28] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, pages 1–42, 2014. 6

[29] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun. Overfeat: Integrated recognition, localization and detection using convolutional networks. *arXiv preprint arXiv:1312.6229*, 2013. 2

[30] P. Sermanet, A. Frome, and E. Real. Attention for fine-grained categorization. *arXiv preprint arXiv:1412.7054*, 2014. 1, 2

[31] Y. Shi, A. Bobick, and I. Essa. Learning temporal sequence model from partially labeled data. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 2, pages 1631–1638. IEEE, 2006. 2

[32] T. Shu, D. Xie, B. Rothrock, S. Todorovic, and S.-C. Zhu. Joint inference of groups, events and human roles in aerial videos. In *CVPR*, 2015. 2

[33] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 2, 5, 6

[34] C. Sun, S. Shetty, R. Sukthankar, and R. Nevatia. Temporal localization of fine-grained actions in videos by domain

transfer from web images. *arXiv preprint arXiv:1504.00983*, 2015. 2

[35] C. Szegedy, S. Reed, D. Erhan, and D. Anguelov. Scalable, high-quality object detection. *arXiv preprint arXiv:1412.1441*, 2014. 2

[36] Y. Tian, R. Sukthankar, and M. Shah. Spatiotemporal deformable part models for action detection. In *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, pages 2642–2649. IEEE, 2013. 2

[37] T. Tieleman and G. E. Hinton. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude., 2012. 5

[38] H. Wang and C. Schmid. Action recognition with improved trajectories. In *Computer Vision (ICCV), 2013 IEEE International Conference on*, pages 3551–3558. IEEE, 2013. 5, 6

[39] L. Wang, Y. Qiao, and X. Tang. Action recognition and detection by combining motion and appearance features. 1, 2, 5, 6

[40] D. Weinland, R. Ronfard, and E. Boyer. A survey of vision-based methods for action representation, segmentation and recognition. In *Computer Vision and Image Understanding, Vol. 115, Issues 2, pp. 224,241*, 2010. 2

[41] P. Weinzaepfel, Z. Harchaoui, and C. Schmid. Learning to track for spatio-temporal action localization. *arXiv preprint arXiv:1506.01929*, 2015. 2

[42] R. J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256, 1992. 1, 2, 4

[43] K. Xu, J. Ba, R. Kiros, A. Courville, R. Salakhutdinov, R. Zemel, and Y. Bengio. Show, attend and tell: Neural image caption generation with visual attention. *arXiv preprint arXiv:1502.03044*, 2015. 1, 2

[44] A. Yao, J. Gall, and L. Van Gool. A hough transform-based voting framework for action recognition. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 2061–2068. IEEE, 2010. 2

[45] G. Yu and J. Yuan. Fast action proposals for human action detection and search. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1302–1311, 2015. 2

[46] J. Yuan, Y. Pei, B. Ni, P. Moulin, and A. Kassim. Adsc submission at thumos challenge 2015. 1, 2

[47] W. Zaremba and I. Sutskever. Reinforcement learning neural turing machines. *arXiv preprint arXiv:1505.00521*, 2015. 2

[48] S. Zha, F. Luisier, W. Andrews, N. Srivastava, and R. Salakhutdinov. Exploiting image-trained cnn architectures for unconstrained video classification. *arXiv preprint arXiv:1503.04144*, 2015. 7

[49] H. Zhong, J. Shi, and M. Visontai. Detecting unusual activity in video. In *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, volume 2, pages II–819. IEEE, 2004. 2

[50] Y. Zhu, R. Kiros, R. Zemel, R. Salakhutdinov, R. Urtasun, A. Torralba, and S. Fidler. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. *arXiv preprint arXiv:1506.06724*, 2015. 2