Budget-Aware Activity Detection with A Recurrent Policy Network

Behrooz Mahasseni[†1]
https://mahasseb.github.io
Xiaodong Yang[2]
http://xiaodongyang.org
Pavlo Molchanov[2]
pmolchanov@nvidia.com
Jan Kautz[2]
http://jankautz.com

[1] Apple Inc.
[2] NVIDIA Research

Abstract

In this paper, we address the challenging problem of efficient temporal activity detection in untrimmed long videos. While most recent work has focused and advanced the detection accuracy, the inference time can take seconds to minutes in processing each single video, which is too slow to be useful in real-world settings. This motivates the proposed budget-aware framework, which learns to perform activity detection by intelligently selecting a small subset of frames according to a specified time budget. We formulate this problem as a Markov decision process, and adopt a recurrent network to model the frame selection policy. We derive a recurrent policy gradient based approach to approximate the gradient of the non-decomposable and non-differentiable objective defined in our problem. In the extensive experiments, we achieve competitive detection accuracy, and more importantly, our approach is able to substantially reduce computation time and detect multiple activities with only 0.35s for each untrimmed long video.

1 Introduction

Efficient temporal activity detection in untrimmed long videos is fundamental for intelligent video analytics including automatic categorizing, searching, indexing, segmentation, and retrieval of videos. This is a challenging problem as algorithms must (1) determine whether a specific activity occurs in an untrimmed video; (2) identify the temporal extent of each activity; and (3) maximize detection accuracy within a given time budget. In temporal activity detection, the most time consuming step is the execution of CNNs or hand-crafted feature extractors to every sliding window or proposal segment [4, 8, 30], typically taking, e.g., seconds to minutes to process one video in THUMOS14 [9]. Unfortunately, this rules out the practical use of these methods for the applications that require real-time and large-scale video processing. Although hardware solutions in some scenarios can help meet the constraints, it is equally important to establish a better understanding of how to achieve a maximal detection accuracy given the constraints on time and resource.

[†]Work done during an internship at NVIDIA Research.

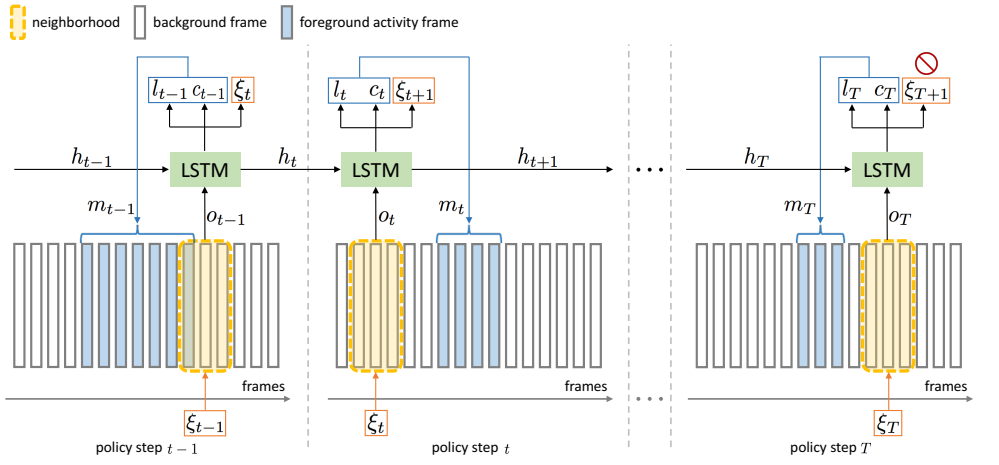


Figure 1: Given an untrimmed long video, at each step t the policy has access to the local observation o_t of a neighborhood centered around the current selected frame at ξ_t. For each step, the policy predicts a segment m_t and produces three outputs: the temporal location l_t (i.e., start and end) of the segment, the estimated class c_t associated with the segment, and the next frame to observe at ξ_{t+1}. According to a specified time budget, the policy runs for T steps then completes the detection process.

Recently, there is a fast growing interest in the research of temporal activity detection. Most existing work [11, 12, 18, 22, 30] hinges on a large set of features and classifiers that exhaustively run over every time step at multiple temporal scales. This sliding window scheme is obviously computationally prohibitive for applications such as the ones running on mobile and embedded devices. To avoid such exhaustive evaluations, a number of action proposal algorithms [4, 8, 14] have been proposed to produce a set of candidate temporal segments that are likely to contain a certain action. A separate classifier is then applied on these proposal segments for action classification. However, we argue that it is suboptimal to divide temporal activity detection into the two disjointed steps: proposal and classification. Moreover, the large number of proposal segments, e.g., thousands of proposals per video, is still unsatisfying in term of computational efficiency.

In this paper, we address this problem by introducing a fully end-to-end and budget-aware framework, which learns to optimally select a small number of video frames according to a time budget to perform temporal activity detection. We formalize our frame selection as a Markov decision process (MDP), and adopt a recurrent network to model the policy for selecting frames. We develop a policy gradient approach based on reinforcement learning to approximate the gradient of our non-decomposable and non-differentiable objective function. Figure 1 illustrates the detection process of our approach.

Our main contributions of this paper are summarized as follows. First, we achieve competitive accuracy by using only 0.35s for each untrimmed minutes-long video, speeding up detection by orders of magnitude compared to the existing methods. Second, we present a fully end-to-end policy based model with a single training phase to handle the activity classes in their entirety. Third, to our knowledge, we provide the first approach to directly optimize the mean average precision (mAP) criteria, i.e., the final evaluation metric, in the objective function for activity detection.

2 Related Work

A large family of the research in video activity understanding is about activity classification, which provides useful tools for temporal activity detection, such as the two-stream networks with separate CNNs operating on the color and optical flow modalities [16, 27], and the C3D and I3D using 3D-CNN to process short video clips [1, 21]. RNNs can be applied with CNNs to model temporal dynamics and handle variable-length video sequences [6, 28].

Another active research line is the spatio-temporal action detection, which focuses on localizing action regions over consecutive video frames. A number of methods have been proposed, from the spatio-temporal proposals such as supervoxels [19], the frame-level object detection followed by a linking or tracking algorithm [5], to the recent video SSD approach [10]. However, these methods are mostly applied on short video snippets, in contrast, temporal activity detection targets at untrimmed videos that involve complex actions, objects and scenes evolving over long sequences. Therefore, efficient inference under a certain time budget is much in demand for the temporal activity detection task.

A majority of the existing approaches [11, 12, 18, 22, 30] for temporal activity detection focus on extracting various features to represent sliding windows and subsequently classifying them with SVMs trained on the multiple features. Alternative proposal based methods can be used to generate action segments to replace the exhaustive sliding windows. A sparse learning based framework is presented in [8] to produce segment proposals with high recall. Escorcia et al. [4] introduce a proposal algorithm based on C3D and LSTM. Shou et al. [14] propose a 3D-CNN based multi-stage framework with three different networks for proposal, classification and localization. A convolutional-de-convolutional network is further introduced in [15] to boost the precision for temporal boundaries of proposal segments. These proposal based detection methods are by nature stage-wised, and therefore are not end-to-end trainable. R-C3D is recently developed in [26] to save computation through sharing convolutional features between proposal and classification pipelines.

Yeung et al. [29] present an end-to-end learning method, which directly predicts temporal boundaries from raw videos and exhibits fast inference capability. In that work, a recurrent attention model is learned to select a subset of frames to interact with, and maintains high detection accuracy. Our work differs from [29] mainly in: (1) unlike their binary model specifically trained for each action class, our approach is able to handle multiple classes; (2) rather than using a separate emission signal to identify foreground segments, we consider all predicted outputs as valid segments since we include the background as an additional class; (3) their reward function is designed to maximize true positives and minimize false positives, while our retrieval loss is directly defined on the mAP; and (4) instead of using two schemes (i.e., back-propagation for candidate detection and REINFORCE for prediction indicator and next observation) to train their learning agent, we employ a unified recurrent policy gradient to train the entire policy altogether.

3 Problem Formulation

Given a video v and a set of activity labels $\mathcal{L}$, our goal is to predict for each frame a single label from $\mathcal{L}$. We call each temporal extent consisting of consecutive frames with the same label a semantic temporal segment. As stated in Section 1, given a limited time budget, it is infeasible to process every single frame in a video. So we aim to detect and classify the foreground segments by only observing a small subset of video frames $x \in v$.

Assuming limited access to the frames of v, finding the optimal frame subset x is inherently a sequential decision making task. Accordingly, we draw on ideas from reinforcement learning—an area that focuses on learning for sequential decision making problems. Our aim is to learn a policy π, parameterized by θ, to sequentially select the frames from v and form the subset x. Alongside the selection process, π outputs the current belief about the foreground segment and the associated class label. This sequential decision making process intuitively resembles how humans search activities in a video, i.e., iteratively refine our estimated temporal boundaries by sequentially choosing a few frames to observe.

Let $\mathbb{G}$ denote the ground truth segments in v, and $\mathbb{M}^x$ be the set of estimated semantic temporal segments from observing x. We define the deterministic indicator $\mathbb{I}_{m,g}$ to identify whether an estimated segment $m \in \mathbb{M}^x$ is assigned to a ground truth segment $g \in \mathbb{G}$:

$$\mathbb{I}_{m,g} = \begin{cases} 1 & g = \arg\max_{g' \in \mathbb{G}} \alpha(m,g') \text{ subject to } \alpha > 0, \\ 0 & \text{otherwise}, \end{cases} \tag{1}$$

where α is the intersection over union (IoU). Let c_m and c_g indicate the probability distribution and the one-hot representation of class label for segments m and g. For a subset of selected frames x and a set of predicted segments $\mathbb{M}^x$, our loss is defined as:

$$L_\theta = \sum_{m \in \mathbb{M}^x} \sum_{g \in \mathbb{G}} \mathbb{I}_{m,g} \Big[\lambda_c \Delta_{cls}(c_m, c_g) + \lambda_l \Delta_{loc}(l_m, l_g) \Big] + \lambda_r \Delta_{ret}(\mathbb{M}^x, \mathbb{G}), \tag{2}$$

where Δ_{cls} is the multi-class classification error, Δ_{loc} is the localization error with l_m and l_g identifying the locations of segments m and g, and Δ_{ret} is the segment retrieval error. The most important property of Δ_{ret} is that while it encourages the model to detect all foreground segments, it also discourages the model from producing many false positives.

We now explain how to formulate each individual error defined in Eq. (2). In contrast to using a binary classification loss as in [29], we employ a multi-class cross-entropy loss $\Delta_{cls} = -c_g \log c_m$. Unlike [29] which penalizes the localization based on the absolute error, we believe this loss should also depend on the duration of a segment, i.e., the same amount of absolute error should be treated differently for short and long intervals. Intuitively, this means that if the policy makes a small error for a short segment this error should be considered relatively large, otherwise the algorithm would ignore the small segments. With this intention, we define $\Delta_{loc}(l_m, l_g) = \zeta(g) \times \|(m_s, m_e), (g_s, g_e)\|$, where $\zeta(g)$ is a scaling factor which depends on the length of segment g, $\|\cdot\|$ is the distance between two segments, m_s and m_e are the start and end of segment m, similar for segment g. To define the segment retrieval loss $\Delta_{ret}(\mathbb{M}^x, \mathbb{G})$, we use the mAP criteria, where mean is over different class labels, and AP for each individual class is defined as $\text{AP}(\mathbb{M}^x, \mathbb{G}) = \sum_i \text{Prec}(\mathbb{M}^x(i), \mathbb{G}) \times \Delta_{\text{Recall}}$, where $\mathbb{M}^x(i)$ is the subset of $\mathbb{M}^x$ till the ith segment ranked by the overlap with ground truth, $\text{Prec}(\cdot)$ is the precision of detection, and Δ_{Recall} is the change of recall from previous subset. Given a training set of N videos $\{v_1, \cdots, v_N\}$, our goal is to find θ that minimizes:

$$\theta^* = \arg\min_\theta \Big[\mathbb{E}(L_\theta) \approx \frac{1}{N} \sum_{n=1}^{N} L_\theta(\mathbb{M}_n^x, \mathbb{G}_n) \Big]. \tag{3}$$

Unfortunately, the standard back-propagation is not applicable to learn the parameters in Eq. (3), as the objective function in Eq. (2) contains the non-differentiable components. This is mainly due to the non-decomposable AP, as well as the sequential decision making process in selecting video frames. In order to solve this difficulty, we reformulate our problem as a reinforcement learning problem, which allows us to define an equivalent reward function to the original objective function.

4 Recurrent Policy Network

In this section, we describe our proposed representation for the policy π, and present the approach for learning the parameters θ by drawing on the recurrent policy gradient estimation.

4.1 Policy Representation

Our activity detection agent makes a sequence of predictions based on the local information from the most recent observed frame. At each step, the policy produces three outputs including the estimate of start and end of the current potential temporal segment, the prediction of class label associated with the segment, and the next frame to observe. Unlike the binary model used in [29], our approach enables us to define a multi-class classifier, which means that we only need to train a single policy rather than training multiple different policies. Note that this also allows us to avoid the binary prediction indicator signal used in [29], since we can directly discard those segments predicted with the background label.

Due to the local observation at each step, the policy has no access to the global state (i.e., the entire video). This resembles the partially observable Markov decision process (POMDP), which assumes that despite the existence of a global state, for practical reasons an agent does not have a full observation of the global state. We adopt the recurrent policy gradient approach [24] to maintain an approximate belief of the current state s_t by LSTM.

Particularly, suppose at step t the current frame is i, the policy π makes a decision based on (1) the local information of a neighborhood $\mathcal{N}_i$ centered around i and (2) the history of previous observations. We capture the local information through an observation feature $o_t = [\psi(\mathcal{N}_i), \phi(\mathcal{N}_i), \xi_t]$, where $\psi(\mathcal{N}_i)$ is an indicator vector that identifies whether each frame in $\mathcal{N}_i$ has been previously selected, $\phi(\mathcal{N}_j)$ is the average of per-class confidence predicted in $\mathcal{N}_i$, and $\xi_t \in [0,1]$ is the normalized location of the current frame at step t. The inclusion of ξ_t is helpful in encouraging the policy to cover broader video content. During our experiments, we find that excluding ξ_t results in a considerable number of over-selection of frames. Note that for ϕ, we compute the averaged confidence of estimated segments, which share the frames in $\mathcal{N}_i$. As for the history of the decision makings, we use the hidden state h_{t-1} of LSTM to maintain the context of previous observations up to step t.

To summarize, the global state at step t is approximated by the internal state h_t of LSTM, which depends on the current observation o_t and the previous state h_{t-1}. Given h_t the outputs of the policy π are $\nu_t = [l_t, c_t, \xi_{t+1}]$: (1) the location l_t of an estimated temporal segment, (2) the probability distribution over activity class labels c_t, and (3) the location of the next observation ξ_{t+1}. Note that our formulation allows the policy to perform both forward and backward selections. In order to further improve the exploration at training phase, instead of directly using ξ_{t+1}, the next selected location is sampled from a Gausssian distribution with a mean equal to ξ_{t+1} and a fixed variance (e.g., 0.18 used in our experiments).

4.2 Policy Learning

Our goal of policy learning is to jointly optimize the parameters of π by minimizing the loss of a sequence of policy actions as defined in Eq. (2). These actions are taken from the initial state s_0, when no frames are selected, until the final state s_T, where T is the number of steps specified according to a time budget.

The main difficulty in policy learning is that the estimated temporal segments $\mathbb{M}^x$ for a video are computed through a sequence of policy decisions, resulting in a non-decomposable

and non-differentiable objective function. Moreover, a decision that the policy makes at any step depends on the history of decisions that the policy has made in previous steps, and also impacts the decisions available to the policy in the future. Among the potential algorithms for addressing similar POMDP problems [2, 3, 24, 25], we adopt the recurrent policy gradient approach [24], which provides better theoretical bounds on learning objective to approximate the gradients of our non-decomposable and non-differentiable objective function, so that the policy can be efficiently learned with stochastic gradient descent.

To follow the general reinforcement learning formulation, let r be the immediate reward associated with a state s_t. Since $s_t \approx h_t$ in our policy, we define r as $r(h_t) = L_\theta(\mathbb{M}^x_{t-1}, \mathbb{G}) - L_\theta(\mathbb{M}^x_t, \mathbb{G})$, where L_θ is the loss associated with a set of estimated temporal segments as defined in Eq. (2). Intuitively, $r(h_t)$ states that the policy earns an immediate reward equal to the decrease in the temporal segmentation error achieved by selecting an observed frame, or pays a penalty if the temporal segmentation error increases. Let $R(H_t)$ be the discounted accumulated reward starting from the state s_t and continuing the policy up to the final state s_T: $R(H_t) = \sum_{t'=t}^{T} \tau^{t'-t} r(h_{t'})$, where $H_t = \{h_t, ..., h_T\}$ represents the history of hidden states in LSTM, and $\tau \in (0,1)$ is the discount factor. H_0 can be interpreted as the trajectory of observations for a sample run of the policy from the initial state. For notational simplicity, we use H for H_0 in the rest of this paper. The goal of policy learning is transformed to find the parameters θ^* to maximize $J(\theta)$ which is defined as:

$$J(\theta) = \mathbb{E}[R(H)] = \int p(H|\theta) R_\theta(H) dH, \tag{4}$$

where $p(H|\theta)$ is the probability of observing a sequence of hidden states H, given a policy π defined by the parameters θ. It can be shown that maximizing $J(\theta)$ implicitly minimizes L_θ along the trajectory of policy executions. We now derive how to compute the gradient with respect to the policy parameters $\nabla_\theta J$, which is given by:

$$\nabla_\theta J = \int \left[\nabla_\theta p(H|\theta) R_\theta(H) + p(H|\theta) \nabla_\theta R_\theta(H)\right] dH. \tag{5}$$

Note that given the sequence of hidden states H, which determines the history of selected frames, the reward function does not depend on the policy parameters, yielding $\nabla_\theta R_\theta(H) = 0$. To further simplify Eq. (5), we need to define $\nabla_\theta p(H|\theta)$. So we first factorize $p(H|\theta)$ as $p(H|\theta) = p(h_0) \prod_{t=1}^{T} p(h_t|h_{t-1}) \pi(v_t|h_{t-1}, o_t)$, where the same notation π is used to denote the output of the policy. Based on this we have: $\log p(H|\theta) = \text{const} + \sum_{t=1}^{T} \log \pi(v_t|h_{t-1}, o_t)$, where the first term is a sum over the log of $p(h_t|h_{t-1})$, a constant with respect to θ. This therefore results in the following gradient: $\nabla_\theta \log p(H|\theta) = \sum_{t=1}^{T} \nabla_\theta \log \pi(v_t|h_{t-1}, o_t)$.

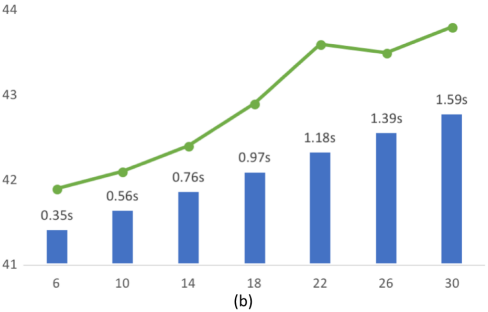
It is common to use the Monte-Carlo integration to approximate the integration over the probability of observing a sequence of hidden states. Specifically, the approximate gradient is computed by running the current policy on N training videos to generate N trajectories. Combining aforementioned derivations and Eq. (5), we can obtain the approximate gradient:

$$\nabla_\theta J \approx \frac{1}{N} \sum_{n=1}^{N} \sum_{t=1}^{T} \left[\nabla_\theta \log \pi(v_t^n|h_{t-1}^n, o_t^n) R_\theta(h_t^n)\right]. \tag{6}$$

Since the policy gradient methods usually suffer from the high variance of gradient estimates, we follow the common practice used in [25] to subtract a bias from the expected reward R. However, rather than taking a constant bias, we set the bias value to be the reward obtained from a random selection policy.

Method	0.3	0.4	0.5	0.6	0.7
CNN	12.6	9.7	6.1	4.1	2.4
CNN_{dif}	15.9	13.1	10.2	5.4	3.8
LSTM	12.8	11.3	9.4	6.4	3.9
LSTM_{dif}	18.4	19.2	15.1	9.7	4.1
Ours_{bas}	34.1	24.2	17.4	10.6	6.2
Ours_{dif}	37.6	26.7	20.9	10.9	6.6
$\text{Ours}_{\text{reg}}, \kappa = 5$	38.3	27.5	21.1	11.2	7.1
$\text{Ours}_{\text{reg}}, \kappa = 10$	38.4	28.9	22.4	11.4	7.0
$\text{Ours}_{\text{reg}}, \kappa = 20$	38.7	28.5	22.5	11.8	7.1

(a)



(b)

Figure 2: (a) Comparison of the detection accuracy of baselines and variations of our models under different IoU thresholds α on THUMOS14. (b) Comparison of the detection accuracy (green dots) and the speed (blue bars) with different policy steps on ActivityNet.

5 Experiments

In this section, we extensively evaluate our approach on the two benchmark datasets: THUMOS14 [9] and ActivityNet [7]. Experimental results demonstrate that our approach is able to substantially reduce computational time, and meanwhile provides competitive detection accuracy under varying time budgets. In the supplementary material, we explain how to calculate the detection speeds of different methods, and present the illustration of the learned policy for frame selection.

We use the pre-trained VGG16 [17] on ImageNet as our backbone CNN, and fine-tune the network on each dataset. We take the layer `fc7` of VGG16 as the per-frame feature. Our policy is based on a two-layer LSTM, and each layer has 1024 hidden units. If not otherwise specified, our policy takes $T = 6$ steps, which we believe are efficient enough to meet a reasonable time budget constraint. We empirically set the weights in our loss function of Eq. (2) as $\lambda_c = \lambda_l = 1.0$ and $\lambda_r = 0.5$. We set the batch size to 128, and train each model for 100 epochs using Adam with the default parameters. We implement our networks in TensorFlow and perform experiments on a single NVIDIA Tesla P100 GPU.

5.1 Baselines and Variations of Our Model

To provide a better insight, we first study the different configurations of baseline method and our model. We define the following two important baselines, each of them outputs a class label for a single frame including the background class, followed by a non-maximum suppression (NMS) post-processing to aggregate the class probabilities of single frames.

- **CNN with NMS**: VGG16 is fine-tuned on the single frames of each dataset and used to perform per-frame classification.
- **LSTM with NMS**: According to the setting in our model, we train LSTM on top of the layer `fc7` of VGG16, and a prediction is made for each single frame.

We can improve our base model mainly in the two ways and define the multiple variations of our model as follows.

- **Ours_{bas}** is the base model defined in Sec. 4.1.
- **Ours_{dif}** provides the simple pixel-level frame difference between consecutive frames to roughly capture motion cues. Although we can incorporate optical flow [20], given the fact that optical flow is usually computationally expensive, and the motivation of

this work is for fast inference, we believe that the pixel-level frame subtraction is a reasonable compromise. We use early fusion to concatenate RGB channels of the original frame and the frame difference as a composite input to VGG16.

- **Ours$_{\text{reg}}$** performs a simple post-processing to refine the policy output. A simple linear regressor is trained to refine the boundaries of detected temporal segments by using the current temporal extend of a segment, and the κ uniformly sampled frames along with their pixel-level frame differences within this segment.

5.2 Results on THUMOS14

We follow the standard experimental setting on THUMOS14 [9]. We first present the ablation study to understand the contribution of each individual component in our model. As expected and shown in Figure 2(a), providing the additional simple frame difference to capture coarse motion cues improves the accuracy of all baselines and our models. The baseline recurrent models with LSTM are found to produce better results than CNN. All of our ablation models significantly outperform the baselines, quantifying the contributions of our proposed approach. In particular, our results are generated by only observing 6 policy-selected frames, far more efficient than the baselines that have to densely go through all video frames.

Method	Time (s)	$\alpha = 0.3$	$\alpha = 0.4$	$\alpha = 0.5$	$\alpha = 0.6$	$\alpha = 0.7$
LEAR14 [12]	> 108	28.8	21.8	15.0	8.5	3.2
CUHK14 [22]	> 108	14.6	12.1	8.5	4.7	1.5
Pyramid of Scores [30]	> 108	33.6	26.1	18.8	-	-
Fast Temporal Proposals [8]	108	-	-	13.5	-	-
S-CNN [24]	92	36.3	28.7	19.0	10.3	5.3
CDC [15]	92	40.1	29.4	23.3	13.1	7.9
DAPs [4]	41	-	-	13.9	-	-
Language Model [13]	17	30.0	23.2	15.2	-	-
R-C3D [26]	5.3	44.8	35.6	28.9	-	-
Glimpses [29]	4.9	36.0	26.4	17.1	-	-
Ours	**0.35**	38.4	28.9	22.4	11.4	7.0

Table 1: Comparison of our approach and the state-of-the-art methods in the approximate computation time to process each video and the detection accuracy on THUMOS14.

It is interesting to observe that the simple linear regression based post-processing with κ uniformly sampled frames from estimated segments helps in refining the temporal boundaries in Figure 2(a). We conjecture that this is due to the fact that our policy is allowed to observe frames in a temporally inconsistent way, i.e., selecting frames in a mixed forward and backward fashion. LSTM thus tends to smooth out the features to some extent during this process. We hypothesize that observing the κ sampled frames in the simple regression provides a temporally consistent description complementary to the averaged latent representation that lacks temporal consistency in the policy. We also evaluate the impact of the number of sampled frames κ to the regression. As shown in Figure 2(a), we only observe marginal gains when sampling over 10 frames, which also implies that our policy has already learned to select fairly representative frames to perform the detection.

We then compare with the state-of-the-art methods in Table 1. Our approach achieves competitive detection accuracy under various IoU thresholds, and more importantly, we per-

Method	Time (s)	$\alpha = 0.5$	$\alpha = 0.75$	$\alpha = 0.95$	ave-mAP
UTS16 [23]	> 930	45.1	4.1	0.0	16.4
CDC [15]	> 930	45.3	26.0	0.2	23.8
UPC16 [11]	11	22.5	-	-	-
OBU16 [18]	10	22.7	10.8	0.3	11.3
R-C3D [26]	3.2	26.8	-	-	-
Ours	**0.35**	41.9	22.6	0.1	21.5

Table 2: Comparison of our approach and the state-of-the-art methods in the approximate computation time to process each video and the detection accuracy on ActivityNet.

form detection in only 0.35s for each untrimmed long video. This is orders of magnitude faster than most other competing algorithms relying on sliding windows or segment proposals. While R-C3D produces superior accuracy on this dataset, we significantly outperform R-C3D on ActivityNet (see Table 2), indicating the advantage of our approach to handle the more complex activities. We specifically compare the per-class breakdown AP of our model and the glimpses method [29] that also exhibits efficient inference for each binary detection. As shown in Figure 3(a), our approach largely outperforms [29] in 15 out of 20 classes, and by 5.3% in overall mAP. Notably, our method is a unified model to handle all classes, while [29] is a binary model that needs to be trained for each specific class of the 20 actions, therefore we need to run their models multiple times to detect the whole classes.

We also provide the in-depth analysis of the computational costs of our approach. Figure 3(b) shows the percentage of time for each major algorithm step and computation component. Our policy is quite efficient to run, takes only 9.4% of the time. The feature extraction that involves applying VGG16 to multiple frames dominates the computations, consuming 80.1% of the time, which again highlights the importance of effective frame selection to reduce the computational burden. In addition, it uses only 2.9% of the time to compute the frame differences, which can provide coarse but useful motion cues.

5.3 Results on ActivityNet

Compared to THUMOS14, ActivityNet [7] contains high-level semantics with complex actions, objects and scenes in videos, and is much larger in number of activities and amount of videos. Based on this large-scale dataset, we first evaluate our model with different policy steps, according to different time budgets that the detection system can afford. As shown in

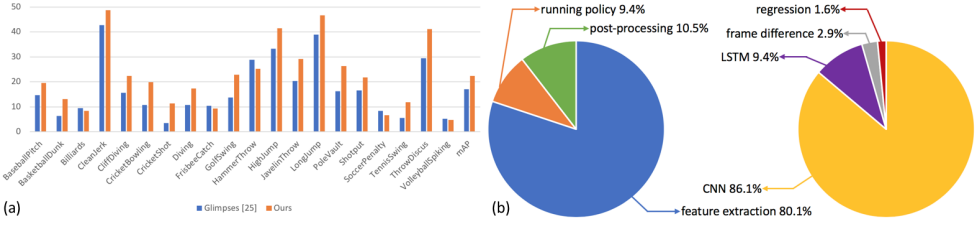


Figure 3: (a) Comparison of the per-class breakdown AP at $\alpha = 0.5$ on THUMOS14. (b) Analysis of computational time of our approach: percentage of time spent on each major algorithm step (left) and computation component (right) to perform detection.

Figure 2(b), when the policy steps move up from 6 to 30 by increasing the time budget from 0.35s to 1.59s, the detection accuracy is improved by about 2.0% under $\alpha = 0.5$.

Finally, we compare our approach with the state-of-the-art methods in Table 2. Similar to the results on THUMOS14, our approach substantially reduces the detection time by orders of magnitude compared to other methods. While CDC provides very competitive accuracy, it relies on the detection results of UTS16, i.e., CDC is primarily used to refine the predicted temporal boundaries of UTS16. If directly applying CDC on raw videos, the accuracy drops to around 15.0% at $\alpha = 0.5$. UTS16 is sliding window based, and requires multiple expensive feature extractions including iDT, C3D, ResNet152, and InceptionV3, whihch are considerably expensive for inference. We achieve significant improvement over the most recent state-of-the-art method R-C3D, demonstrating the superiority of our approach to tackle the more complex activities. Figure 4 demonstrates the prediction examples of our model on ActivityNet, including the challenging classes that involve great scale change, large viewpoint variations, crowded background, etc. Since the glimpses method [29] is a binary model and their detection result on the entire 200 classes of this dataset is not provided, we train our policy on the same two subsets (i.e., sports and work) as [29] for fair comparisons. More comparison details are provided in the supplementary material.

6 Conclusion

We have presented a fully end-to-end approach for the challenging problem of efficient temporal activity detection. We formulate the budget-aware inference for this problem to optimally select a small subset of frames within the MDP framework. We propose the LSTM based policy model to handle the whole activity classes by a single training phase. A policy gradient is further developed to approximate the gradient of our non-decomposable and non-differentiable objective. Experiments demonstrate that our approach brings substantial time saving and maintains competitive detection accuracy. This provides a practical solution for many applications that require tight runtime constraints and limited on-device computations.

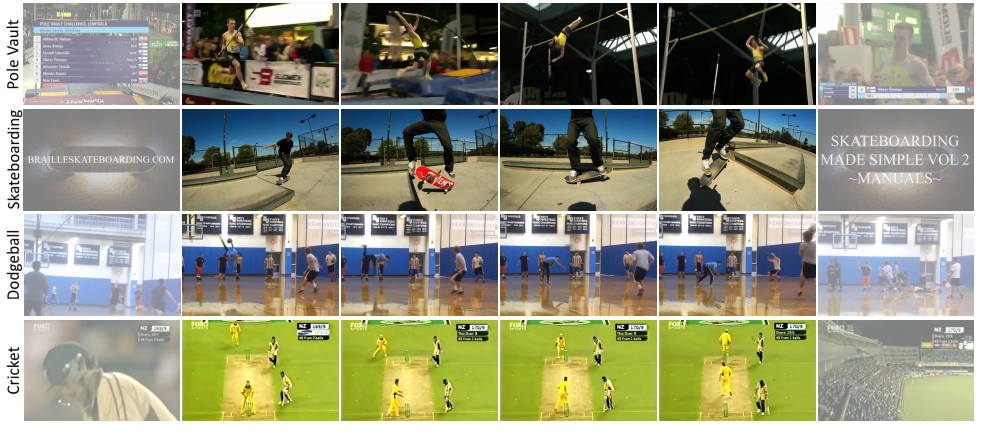


Figure 4: Examples of predicted results on ActivityNet. Each row shows four sampled frames within the temporal extent of a detected activity. Faded frames indicate the frames outside the detected temporal boundary.

References

[1] J. Carreira and A. Zisserman. Quo vadis, action recognition? A new model and the Kinetics dataset. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[2] S. Cho, S. Cho, and K.-C. Yow. A robust time series prediction model using POMDP and data analysis. *Journal of Advances in Information Technology (JAIT)*, 2017.

[3] M. Egorov, Z. Sunberg, E. Balaban, T. Wheeler, J. Gupta, and M. Kochenderfer. POMDPs.jl: A framework for sequential decision making under uncertainty. *Journal of Machine Learning Research (JMLR)*, 2017.

[4] V. Escorcia, F. Heilbron, J. Niebles, and B. Ghanem. DAPs: Deep action proposals for action understanding. In *European Conference on Computer Vision (ECCV)*, 2016.

[5] G. Gkioxari and J. Malik. Finding action tubes. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[6] J. Gu, X. Yang, S. De Mello, and J. Kautz. Dynamic facial analysis: From Bayesian filtering to recurrent neural network. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[7] F. Heilbron, V. Escorcia, B. Ghanem, and J. Niebles. ActivityNet: A large-scale video benchmark for human activity understanding. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[8] F. Heilbron, J. Niebles, and B. Ghanem. Fast temporal activity proposals for efficient detection of human actions in untrimmed videos. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[9] H. Idrees, A. Zamir, Y. Jiang, A. Gorban, I. Laptev, R. Sukthankar, and M. Shah. The THUMOS challenge on action recognition for videos “in the wild". *Computer Vision and Image Understanding (CVIU)*, 2017.

[10] V. Kalogeiton, P. Weinzaepfel, V. Ferrari, and C. Schmid. Action tubelet detector for spatio-temporal action localization. In *International Conference on Computer Vision (ICCV)*, 2017.

[11] A. Montes, A. Salvador, S. Pascual, and X. Giro. Temporal activity detection in untrimmed videos with recurrent neural networks. In *NIPS Workshop*, 2016.

[12] D. Oneata, J. Verbeek, and C. Schmid. The LEAR submission at THUMOS 2014. In *ECCV THUMOS Workshop*, 2014.

[13] A. Richard and J. Gall. Temporal action detection using a statistical language model. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[14] Z. Shou, D. Wang, and S.-F. Chang. Temporal action localization in untrimmed videos via multi-stage CNNs. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[15] Z. Shou, J. Chan, A. Zareian, K. Miyazawa, and S.-C. Chang. CDC: convolutional-de-convolutional networks for precise temporal action localization in untrimmed videos. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[16] K. Simonyan and A. Zisserman. Two-stream convolutional networks for action recognition in videos. In *Neural Information Processing Systems (NIPS)*, 2014.

[17] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations (ICLR)*, 2015.

[18] G. Singh and F. Cuzzolin. Untrimmed video classification for activity detection: Submission to ActivityNet challenge. In *CVPR ActivityNet Workshop*, 2016.

[19] K. Soomro, H. Idrees, and M. Shah. Predicting the where and what of actors and actions through online action localization. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[20] D. Sun, X. Yang, M.-Y. Liu, and J. Kautz. PWC-Net: CNNs for optical flow using pyramid, warping and cost volume. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[21] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri. C3D: Generic features for video analysis. In *International Conference on Computer Vision (ICCV)*, 2015.

[22] L. Wang, Y. Qiao, and X. Tang. Action recognition and detection by combining motion and appearance features. In *ECCV THUMOS Workshop*, 2014.

[23] R. Wang and D. Tao. UTS at ActivityNet 2016. In *CVPR ActivityNet Workshop*, 2016.

[24] D. Wierstra, A. Förster, J. Peters, and J. Schmidhuber. Recurrent policy gradients. *Logic Journal of IGPL*, 2010.

[25] R. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 1992.

[26] H. Xu, A. Das, and K. Saenko. R-C3D: Region convolutional 3D network for temporal activity detection. In *International Conference on Computer Vision (ICCV)*, 2017.

[27] X. Yang, P. Molchanov, and J. Kautz. Multilayer and multimodal fusion of deep neural networks for video classification. In *ACM Multimedia*, 2016.

[28] X. Yang, P. Molchanov, and J. Kautz. Making convolutional networks recurrent for visual sequence learning. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[29] S. Yeung, O. Russakovsky, G. Mori, and L. Fei-Fei. End-to-end learning of action detection from frame glimpses in videos. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[30] J. Yuan, B. Ni, X. Yang, and A. Kassim. Temporal action localization with pyramid of score distribution features. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.