StartNet: Online Detection of Action Start in Untrimmed Videos

Mingfei Gao[1*] Mingze Xu[2] Larry S. Davis[1] Richard Socher[3] Caiming Xiong[3†]
[1]University of Maryland [2]Indiana University [3]Salesforce Research
{mgao,lsd}@umiacs.umd.edu, mx6@indiana.edu, {rsocher,cxiong}@salesforce.com

Abstract

We propose StartNet to address Online Detection of Action Start (ODAS) where action starts and their associated categories are detected in untrimmed, streaming videos. Previous methods aim to localize action starts by learning feature representations that can directly separate the start point from its preceding background. It is challenging due to the subtle appearance difference near the action starts and the lack of training data. Instead, StartNet decomposes ODAS into two stages: action classification (using ClsNet) and start point localization (using LocNet). ClsNet focuses on per-frame labeling and predicts action score distributions online. Based on the predicted action scores of the past and current frames, LocNet conducts class-agnostic start detection by optimizing long-term localization rewards using policy gradient methods. The proposed framework is validated on two large-scale datasets, THUMOS'14 and ActivityNet. The experimental results show that StartNet significantly outperforms the state-of-the-art by 15%-30% *p-mAP under the offset tolerance of* 1-10 *seconds on THUMOS'14, and achieves comparable performance on ActivityNet with* $\times$10 *smaller time offset.*

1. Introduction

Temporal action localization (TAL) in untrimmed videos has been widely studied in offline settings, where start and end times of an action are recognized after the action is fully observed [4, 7, 8, 13, 32, 42]. With the emerging applications that require identifying actions in real time, *e.g.*, autonomous driving, surveillance system, and collaborative robots, online action detection (OAD) methods [9,12,31,41] have been proposed. They typically pose the TAL problem as a per-frame class labeling task.

However, in some time-sensitive scenarios, detecting accurate action starts in a timely manner is more important than successfully detecting every frame containing actions.

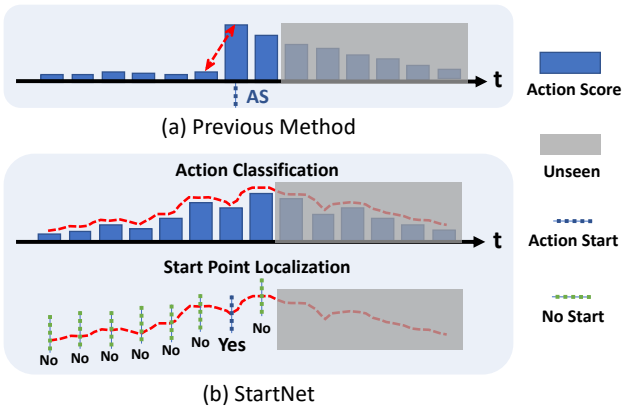

Figure 1. Comparison between (a) the previous method [31] and (b) the proposed framework. [31] aims to generate an action score sequence which produces low score for background and high score for the correct action immediately when the action starts. We propose a two-stage framework: the first stage only focuses on per-frame action classification and the second stage learns to localize the start points given the historical trend of the action scores generated by the first stage.

For example, an autonomous driving car needs to detect the start of "pedestrian crossing" as soon as it happens to avoid collision; a surveillance system should generate alert as soon as a dangerous event is initiated. Online Detection of Action Start (ODAS) was proposed to address this problem specifically [31]. Instead of classifying every frame, ODAS detects the occurrence and category of an action start as soon as possible. Thus, it addresses two sub-tasks: (i) if an action starts at time t and (ii) its associated action class.

The existing method [31] handles the two sub-tasks jointly by training a classification network that is capable of localizing the starts of different action classes. The network attempts to make the representation of a start point close to that of its associated action class and far from its preceding background. As shown in Fig. 1 (a), the network is encouraged to react immediately when an action starts. However, it is hard to achieve this goal due to the subtle appearance difference near start points and the lack of labeled training data (one action only contains one start point).

Our method is inspired by three key insights. First, de-

*Work done when the author was an intern at Salesforce Research.
†Corresponding author.

composing a complex task properly allows sub-modules to focus on their own sub-tasks and makes the learning process easier. A good example is the success of the two-stage object detection framework [15, 16, 29]. Second, as mentioned in [16], when training data is scarce, learning from a representation that is pre-trained on an auxiliary task may lead to a significant performance boost. Third, OAD (per-frame labeling) is very related to ODAS. Comparing to the scarce labeled data of action starts, the amount of per-frame action labels is much larger. Thus, there may be potential benefits if we take advantage of the per-frame labeling task.

Instead of focusing on learning subtle difference near start points, we propose an alternative framework, *i.e*. startNet, and address ODAS in two stages: classification (using ClsNet) and localization (using LocNet). ClsNet conducts per-frame labeling as an auxiliary task based on the spatial-temporal feature aggregation from input videos, and generates score distributions of action classes as a high-level representation. Based on the historical trend of score distributions, LocNet predicts class-agnostic start probability at each time (see Fig 1 (b)). At the end, late fusion is applied on the outputs of both modules to generate the final result. When designing LocNet, we consider the implicit temporal constraint between action starts – two start point are unlikely to be close by. To impose the temporal constraint into the framework under the online setting, historical decisions are taken into account for later predictions. To optimize the long-term reward for start detection, LocNet is trained using reinforcement learning techniques. The proposed framework and its variants are validated on THUMOS'14 [21] and ActivityNet [11]. Experimental results show that our approach significantly outperforms the state-of-the-art by 10%-30% p-mAP under offsets of 1-10 seconds on THUMOS'14, and achieves comparable p-mAP with 10 times smaller time offset on ActivityNet.

2. Related Work

Temporal Action Detection. Most existing methods [4, 7, 8, 13, 32, 42] on temporal action detection formulate the problem in an offline manner. These methods segment actions from long, untrimmed videos and require observing the entire video before making a decision. S-CNN [32] localizes actions with three stages: action proposal generation, proposal classification, and proposal regression. Dai *et al*. [8] proposed TCN which incorporates local context of each proposal for proposal ranking. By sharing features between proposal generation and classification, R-C3D [40] reduces computational cost significantly. Buch *et al*. [4] propose an efficient proposal generation model that avoids working on overlapping regions. Instead of treating temporal action detection as segment-level classification, Shou *et al*. [30] propose CDC network to produce per-frame predictions using 3D convolutional networks.

Online Action Detection. Online action detection is usually solved as a per-frame labeling task [9] on live, streaming videos. As soon as a video frame arrives, it is classified to an action class or background without accessing future frames. De Geest *et al*. [9] first introduced the problem and proposed several models as baselines. Gao *et al*. [12] propose a Reinforced Encoder-Decoder network for action anticipation and treat online action detection as a special case of their framework. Temporal Recurrent Networks [41] set a new state-of-the-art performance by conducting current and future action detection jointly. With the same goal of online per-frame labeling, these methods can serve as ClsNet in our framework.

Early Action Detection. Early action detectors detect actions after only processing a fraction of videos. The earlier a detector recognizes an action, the better it performs. Hoai *et al*. [18] solve this problem by proposing a max-margin framework with structured SVMs. However, this method works on simple scenarios, *e.g*., one video contains only one action. Ma *et al*. [26] design a ranking loss for training assuming that the gaps of predicted scores between correct and incorrect actions should be non-decreasing when an model observes more of an activity.

Online Detection of Action Start (ODAS). As with early action detection, ODAS also aims to recognize actions as soon as possible. Specifically, it focuses on detecting action starts and tries to minimize the time delay of identifying the start point of an action. To the best of our knowledge, [31] is the first and only work that is designed to address ODAS. They solve the problem by encouraging a classification network to learn a representation that can separate action starts from their preceding backgrounds. To achieve the goal, they force the learned representation of an action start window to be similar to that of the following action window and different from that of the preceding background.

Sequential Search with RL. Reinforcement learning (RL) is popular for sequential search problems, since it allows models to be optimized for long-term rewards. Caicedo *et al*. [5] propose a framework based on Deep Q-learning [28] that transforms an initial bounding box iteratively until it lands on an object. Huang *et al*. propose a self-adaptive model [20] which continuously adjusts the boundary of the temporal localization window for action detection. In order to speed up object detection on large images, Gao *et al*. [14] design a coarse-to-fine framework with Deep Q-learning that sequentially selects regions to zoom in only when it is needed. Wu *et al*. [38] propose BlockDrop that trains with policy gradient [35] and improved computational efficiency by dropping unnecessary blocks of ResNets [17]. AdaFrame [39] is also optimized with policy gradient to reduce computations of LSTM by skipping input frames. Our method is related to the above approaches in terms of using

similar RL techniques, but our contributions are mainly formulating ODAS as a two-stage framework and start point detection as a long-term selection process.

3. Action Start Detection Network (StartNet)

The input of an ODAS system is untrimmed, streaming video frames $\{I_1, I_2, ..., I_t\}$. The system processes each video frame sequentially and detects the start of each action instance. At time step t, it outputs a probability distribution, $\mathbf{as}_t^k$, which indicates the start probability of the action class k, without accessing any future information.

The overview of the proposed framework is illustrated in Fig. 2. The framework contains two sub-networks, *i.e.*, a classification network (ClsNet) and a localization network (LocNet). ClsNet focuses on per-frame class labeling. It takes the raw video frames as input and outputs action class probabilities at every time step in an online manner. ClsNet serves two purposes. First, it learns simpler but useful representation for localizing action starts. Second, the classification results can be combined later with the localization results to produce the action starts for each class. LocNet takes the output of ClsNet together with the historical decision vector as inputs. At each time step, it outputs a two-dimensional probability distribution indicating the probability that this frame contains an action start. The historical decision vector records its predictions in the previous n steps in order to model the effect of historical decisions on later ones. Finally, the results of the two networks are fused to construct the final output.

3.1. Classification Network (ClsNet)

Inspired by recent online action detection methods [9,12, 41], we utilize recurrent networks, specifically, LSTM [19], to construct ClsNet. At each time t, it uses the previous hidden state $\mathbf{h}_{t-1}^{(cls)}$, the cell $\mathbf{c}_{t-1}^{(cls)}$, and the feature, $\mathbf{f}_t$, extracted from the current video frame, I_t, as inputs, to update its hidden state $\mathbf{h}_t^{(cls)}$ and cell $\mathbf{c}_t^{(cls)}$. Then, the likelihood distribution over all the action classes can be obtained in Eq. 1,

$$\mathbf{p}_t = softmax(\mathbf{W}_{cls}^T \mathbf{h}_t^{(cls)} + \mathbf{b}), \tag{1}$$

where $\mathbf{p}_t$ is a K dimensional vector and K indicates the number of action classes including background.

To learn ClsNet, action class label for each frame is needed. The cross-entropy loss, $L_{cls}(\mathbf{W}_c)$, is used for optimization during training, where $\mathbf{W}_c$ represents the parameter set of ClsNet.

We observe that ClsNet can be implemented with different architectures. Thus, we validate our framework using two additional structures as the backbone of ClsNet, *i.e.*, CNN and C3D [36]. CNN conducts action classification based only on the arriving frame, I_t. It focuses on the spatial information of the current frame without considering temporal patterns of actions. C3D labels I_t based on each temporal segment consisting of 16 consecutive video frames, from I_{t-15} to I_t. It captures spatial and temporal information jointly using 3D convolutional operations. Comparisons and explanations are discussed in Sec. 4.

3.2. Localization Network (LocNet)

As discussed in Sec. 1, historical action scores can provide useful cues for identifying action starts. At time t, LocNet observes the action score distribution over classes of each frame, $\mathbf{p}_t$, obtained from ClsNet and outputs a two-dimensional vector, $\mathbf{s}_t$, indicating the start and non-start probability distribution.

The start probability is generated sequentially. In general, if an action starts at time step t, there is a low probability that another action also starts at time $t+1$, given reasonable frames per second (FPS). Thus, there are implicit temporal constraints between nearby start points. To enable the model to consider constraints between decisions, we record the historical decisions made by LocNet and use the history to influence later decisions. To enable long-term decision planning, we formulate the problem as a Markov Decision Process (MDP) and use reinforcement learning to optimize our model. When making a decision[1], the model not only considers the effect of the decision at the current step, but also how it will influence the later ones by maximizing the expected long-term reward. In the following, we first discuss the inference phase of LocNet and then the training phase in detail.

Inference Phase. LocNet is built upon a LSTM structure. It acts as an agent which interacts with historical action scores recurrently. During testing, at each state, the agent makes a decision (predicts start probability) that produces the maximum expected long-term reward and updates the state according to the decision. To model the dependency between decisions, we incorporate the record of historical decisions (the decisions made by the agent at previous steps) as a part of the state. The state update procedure is described in Eq. 2 and 3, where $\mathbf{H}_{t-1} = \mathbf{s}_{t-n:t-1}$ indicates historical decisions from step $t-n$ to $t-1$ and $[\mathbf{p}_t, \mathbf{H}_{t-1}]$ indicates the concatenation of the vectors. At the beginning, $\mathbf{H}$ is initialized with zeros.

$$\mathbf{h}_t^{(loc)}, \mathbf{c}_t^{(loc)} = LSTM(\mathbf{h}_{t-1}^{(loc)}, \mathbf{c}_{t-1}^{(loc)}, [\mathbf{p}_t, \mathbf{H}_{t-1}]). \tag{2}$$

$$\mathbf{s}_t = softmax(\mathbf{W}_{loc}^T \mathbf{h}_t^{(loc)} + \mathbf{b}). \tag{3}$$

Training Phase. We train an agent that acts optimally based on the state of the environment. The goal is to maximize the reward by changing the predicted start probability

[1]The term "action" is generally used in reinforcement learning, we use "decision" instead to remove the confusion with action class.

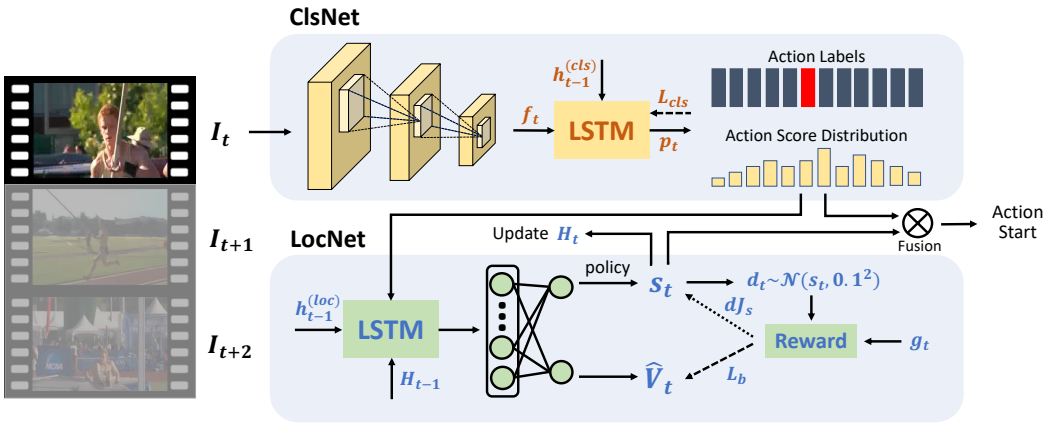

Figure 2. Our method works in two stages with ClsNet and LocNet. ClsNet: at time t, features, $\mathbf{f}_t$, are extracted by deep convolutional networks and input to an one-layer LSTM; The LSTM generates action score distributions at each time step and ClsNet is optimized with cross-entropy loss between action labels and the generated action scores. LocNet: after action score generation, it inputs together with a historical decision vector, $\mathbf{H}$, to a second one-layer LSTM which works as an agent to generate two-dimensional start probability sequentially; $\mathbf{H}$ is updated and the state is changed accordingly; The agent is trained using policy gradient mechanism to optimize long-term reward of start localization. At the end, results from ClsNet and LocNet are fused to obtain the final action start detection results at each time step. Here, ClsNet is implemented with LSTM. CNN and C3D can also be used to construct ClsNet (see Sec. 3.1 for details).

distribution: at a given state, the start probability should be increased when the decision introduces bigger reward and be decreased otherwise. The start prediction procedure is formulated as a decision making policy defined using Gaussian distribution. Following [27, 39], the policy is trained by optimizing with d_t, where d_t, is sampled from $\pi(.|\mathbf{h}_t^{(loc)}, \mathbf{p}_t, \mathbf{H}_{t-1}) = \mathcal{N}(s_t, 0.1^2)$ and s_t indicates the output start probability.

Reward function. Each decision at a given state is associated with an immediate reward to measure the decision made by the agent at the current time. With the goal of localizing start points, we define the immediate reward function in Eq. 4, where $g_t \in \{0, 1\}$ indicates the ground-truth label of action start and d_t is the sampled start probability. The reward function encourages a high probability when there is an actual start and a low probability when there is not by giving a negative reward. Considering the sample imbalance between start points and background, weighted rewards are used by setting a parameter α. In particular, we set α to be the ratio between the number of negative samples to positive samples for each dataset.

$$r_t = \alpha g_t d_t - (1 - g_t) d_t. \tag{4}$$

The long-term reward is the summation of discounted future rewards. In order to maximize the expected long-term reward, the policy is trained by maximizing the objective in Eq. 5, where $\mathbf{W}_s$ represents the parameters of the network and γ is a constant scalar for calculating the discounted rewards over time.

$$J_s(\mathbf{W}_s) = \mathop{\mathbb{E}}_{d_t \sim \pi(.|\mathbf{W}_s)} [\sum_{i=0} \gamma^i r_{t+i}]. \tag{5}$$

Optimization. When optimizing Eq. 5, it is not possible to train the network using error back propagation directly, since the objective is not differentiable. Following [35], we use policy gradient to calculate the expected gradient of J_s as in Eq. 6, where $R_t = \sum_{i=0} \gamma^i r_{t+i}$ indicates the long-term reward at time step t and $\hat{V}_t$ is a baseline value (generated by an fully-connected (FC) layer as shown in Fig. 2) which is widely used in policy gradient frameworks to reduce the variance of the gradient. The principle of policy gradient is to maximize the probability of an action with high reward given a state.

$$\bigtriangledown_{\mathbf{W}_s} J_s = \mathbb{E}[\sum_{t=0}^{\infty} (R_t - \hat{V}_t) \bigtriangledown_{\mathbf{W}_s} log\pi(.|\mathbf{W}_s)]. \tag{6}$$

Following [39], we use the expected long-term reward at the current state as the baseline value and approximate it by minimizing the l_2 loss: $L_b(\mathbf{W}_b) = \frac{1}{2}||R_t - \hat{V}_t||_2$. The training procedure of LocNet is summarized in Alg. 1.

Algorithm 1 Training Process of LocNet

Initialize parameters, $\mathbf{W}_s$, of LocNet
for iteration = 1:N **do**
  Obtain training sequence samples of length T_{loc}
  for t = 1:T_{loc} **do**
    Obtain s_t based on current policy
    Sample decisions: $d_t \sim \mathcal{N}(s_t, 0.1^2)$
    Obtain r_t and $\hat{V}_t$ for each sample
  end for
  Compute $R_{1:T_{loc}}$, $\bigtriangledown_{\mathbf{W}_s} J_s$ and $L_b(\mathbf{W}_b)$
  Update parameters, $\mathbf{W}_s$, of LocNet
end for

The full objective including the loss term in ClsNet is shown in Eq. 7, where λ_1 and λ_2 are constant scalars.

$$\min L_{cls}(\mathbf{W}_c) + \lambda_1 L_b(\mathbf{W}_b) - \lambda_2 J_s(\mathbf{W}_s). \quad (7)$$

Late Fusion. ClsNet outputs an action score distribution and LocNet produces class-agnostic start probabilities at each time step. Then, late fusion is applied to obtain the start probability for each action class, $\mathbf{as}_t^k$, using Eq. 8, where superscript $1{:}K\text{-}1$ indicates positive classes and 0 indicates background.

$$\mathbf{as}_t^k = \begin{cases} s_t \mathbf{p}_t^{1:K-1} & k = 1:K-1 \\ (1-s_t)\mathbf{p}_t^0 & k = 0 \end{cases}. \quad (8)$$

Action start generation. Follow [31], final action starts are generated online if all of the three conditions are satisfied: (i) $c_t = \underset{k}{\operatorname{argmax}}(\mathbf{as}_t^k)$ is an action; (ii) $c_t \neq c_{t-1}$ and (iii) $\mathbf{as}_t^{c_t}$ exceeds a threshold. We set this threshold to 0 by default. An action score sequence generated by ClsNet can also generate action start points online following this procedure. LocNet can locally adjust the start point by boosting time points with higher start probabilities and suppressing those with lower start probabilities.

4. Experiments

To validate the proposed framework, we conduct extensive experiments on two large-scale action recognition datasets, *i.e.*, THUMOS'14 [21] and ActivityNet v1.3 [11].

Evaluation protocol. To permit fair comparisons, we use the point-level average precision (p-AP) proposed in [31] to evaluate our framework. Under this protocol, each action start prediction is associated with a time point. For each action class, predictions of all frames are first sorted in descending order based on their confidence scores and then measured accordingly. An action start prediction is counted as correct only if it matches the correct action class and its temporal distance from a ground-truth point is smaller than an offset threshold (offset tolerance). Similar to segment-level average precision, no duplicate detections are allowed for the same ground-truth point. p-mAP is then calculated by averaging p-AP over all the action classes.

Following [31], we use two metrics based on p-AP to evaluate our framework on THUMOS'14. First, we use p-AP under different offset tolerances, varying from 1 to 10 seconds. Also, we adopt the metric *AP depth at recall (Rec) X%* which averages p-AP on the Precision-Recall curve with the recall rate from 0% to X%. p-mAPs under different offset thresholds are then averaged to obtain the final average p-mAP at each depth. This metric is particularly used to evaluate top ranked predictions and to measure what precision a system can achieve if low recall is allowed. For ActivityNet, we evaluate our methods using p-mAP under offset thresholds of 1-10 seconds at depth Rec=1.0.

Baselines. We compare our framework with the state-of-the-art method, *i.e.*, *Shou et al.* [31] and two baselines that were presented in [31], *i.e.*, *SceneDetect* and *ShotDetect*. The numbers were obtained from the authors [31]. Comparison results with *Shou et al.* [31] demonstrate the superior performance of StartNet. *SceneDetect* and *ShotDetect* are also two-stage methods. Similar to two-stage frameworks of object detection, they first conduct localization by getting action start proposals, which are generated by soft boundary detectors, and then classify them to different classes. Comparison with *SceneDetect* and *ShotDetect* shows the effectiveness of our decomposition design. Our framework trained by policy gradient is indicated by *StartNet-PG*.

Implementation details. Following [12, 31, 41], decisions are made on short temporal chunks, $\mathcal{C}_t$, where I_t is its central frame. The appearance feature (RGB) of $\mathcal{C}_t$ is extracted from I_t and the motion feature (optical flow) is computed using the whole chunk as input. Following [12, 41], chunk size is fixed to 6 and image frames are obtained at 24 FPS. Two adjacent chunks are not overlapping, thus, there are exactly 4 chunks per second. Following [41], for ClsNet, we set the size of LSTM's hidden state to 4096 and the length of each training sequence to 64. When using CNN, we finetune an FC layer with different CNN features as input (see feature descriptions for each dataset). C3D is pretrained on Sports-1M [23] and finetuned for the per-frame labeling task on each dataset. Hidden state of LocNet is set to 128 and the length of each training sequence, T_{loc}, is fixed to 16. Following [39], γ in Eq. 5 is fixed to 0.9. The length of the historical decision vector, n, is set to 8. λ_1 and λ_2 in Eq. 7 are fixed to 1. We adopt an alternating strategy for classification and localization training: ClsNet is first trained and fixed afterwards, and then LocNet is trained upon the pretrained ClsNet. We implement the models in PyTorch [3], and set batch size to 32 for THUMOS'14 and 64 for ActivityNet. For parameter optimization, we used the Adam [24] optimizer with learning rate $5e^{-4}$ and weight decay $5e^{-4}$.

4.1. Experiments on THUMOS'14

Dataset. THUMOS'14 [21] is a popular benchmark for temporal action detection. It contains 20 action classes related to sports. There are only trimmed videos in the training set which makes it not appropriate for training ODAS methods. Following [31], we use the validation set (including 200 untrimmed videos, 3K action instances) for training and the test set (including 213 untrimmed videos, 3.3K action instances) for testing.

Feature description. Two types of features are adopted on THUMOS'14 dataset, *RGB* and *Two-Stream (TS)* features. Following [12, 41], we extract appearance (RGB) feature at

	Offsets (second)	1	2	3	4	5	6	7	8	9	10
Baselines	SceneDetect [1]	1.0	2.0	2.3	3.1	3.6	4.1	4.7	5.0	5.1	5.2
	ShotDetect [2]	1.1	1.9	2.3	3.0	3.4	3.9	4.3	4.5	4.6	4.9
	Shou et al. [31]	3.1	4.3	4.7	5.4	5.8	6.1	6.5	7.2	7.6	8.2
StartNet-PG	C3D [36] + LocNet	6.8	8.0	9.4	10.1	10.6	10.9	10.9	11.1	11.2	11.2
	CNN [37] + LocNet	17.0	23.6	27.6	29.9	31.3	32.1	33.2	33.5	33.9	34.5
	LSTM [19] + **LocNet**	**19.5**	**27.2**	**30.8**	**33.9**	**36.5**	**37.5**	**38.3**	**38.8**	**39.5**	**39.8**

Table 1. Comparisons using p-mAP at depth Rec=1.0 on THUMOS'14. Results are under different offset thresholds. ClsNet is implemented with different structures, *i.e.*, C3D, CNN and LSTM. CNN and LSTM are using TS features.

	Depth Rec.	@0.1	@0.2	@0.3	@0.4	@0.5	@0.6	@0.7	@0.8	@0.9	@1.0
Baselines	SceneDetect [1]	30.0	18.3	12.2	9.1	7.2	6.1	5.2	4.6	4.0	3.6
	ShotDetect [2]	26.3	15.9	11.3	8.6	6.8	5.8	4.9	4.3	3.8	3.4
	Shou et al. [31]	42.7	27.3	19.8	14.9	11.8	10.0	8.5	7.4	6.6	5.9
StartNet-PG	C3D [36] + LocNet	34.8	27.7	22.6	19.0	16.3	14.4	12.9	11.8	10.8	10.0
	CNN [37] + LocNet	71.8	64.7	58.0	52.4	47.2	43.3	39.5	35.9	32.5	29.6
	LSTM [19] + **LocNet**	**77.4**	**70.2**	**64.5**	**59.1**	**54.2**	**49.3**	**45.1**	**41.2**	**37.6**	**34.2**

Table 2. Comparisons using average p-mAP at different depths on THUMOS'14. Average p-mAP means averaging p-mAP over offsets from 1 to 10 seconds. ClsNet is implemented with different structures, *i.e.*, C3D, CNN and LSTM. CNN and LSTM are using TS features.

the *Flatten 673* layer of ResNet-200 [17] and motion feature at the *global pool* layer of BN-Inception [22] with optical flows of 6 consecutive frames as inputs. The TS feature is the concatenation of appearance and motion features, which are extracted with models[2] pre-trained on ActivityNet.

4.1.1 Evaluation Results

Comparisons with previous methods are shown in Table 1 and Table 2. Table 1 shows comparisons based on p-mAP at depth Rec=1.0 under different offset thresholds. All previous methods are under 4% p-mAP at 1 second offset, while StartNet with LSTM achieves 19.5% p-mAP, outperforming the state-of-the-arts largely by over 15%. At 10 seconds offset, previous methods obtain less than 9% p-mAP and StartNet (LSTM) improves over *Shou et al.* [31] by 30% p-mAP. Table 2 shows comparisons based on average p-mAP (averaging over offsets from 1 to 10 seconds) at different depths. The results demonstrate that StartNet with LSTM outperforms previous methods significantly (by around 30%-20% average p-mAP) at depth from Rec=0.1 to Rec=1.0. Obviously, under both metrics, StartNet outperforms previous methods by a very large margin.

To measure the performance gap between online and offline methods. We obtain scores of two recent offline methods [42] and [25] from the authors and evaluate start detection using p-mAP. The p-mAP are 32.7 and 35.7 (Rec=1.0, offset is 1 second). As expected, they outperform StartNet, since they observe the entire action before prediction.

4.1.2 Ablation Experiments

ClsNet implemented with different structures. Comparisons among StartNet with different ClsNet's backbones are shown in Table 1 and Table 2. *LSTM+LocNet* achieves the best performance among the three structures and *C3D* performs worse than *CNN* and *LSTM*. *Shou et al.* [31] chose *C3D* as its backbone and proposed sophisticated training strategies for optimization. With *C3D*, StartNet still significantly outperforms *Shou et al.*, which demonstrates the effectiveness of our framework. Since *LSTM+LocNet* achieves the best performance, the following ablation studies are conducted using ClsNet implemented with LSTM.

[2]https://github.com/yjxiong/anet2016-cuhk.

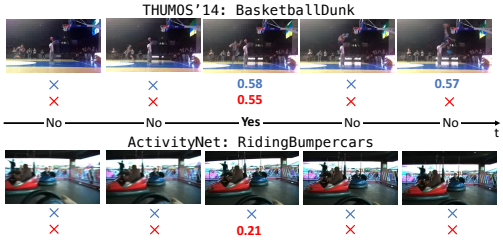

Figure 3. Qualitative results on THUMOS'14 and ActivityNet after action start generation in late fusion. × means no starts are detected at those times. Numbers indicate the scores of detected action starts. Results of ClsNet and StartNet are marked in blue and red, respectively. Yes/No (ground-truth) indicates if an action of the associated class starts at the time. Best viewed in color.

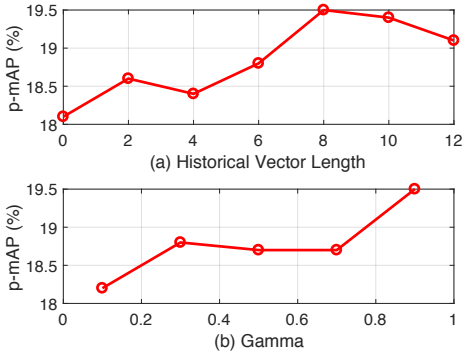

Figure 4. Ablation study of LocNet: (a) effect of length of historical decision vector (b) effect of different gamma values in Eq. 5. Generally, the model performs better with bigger gamma and longer historical decision vector.

Features	Offsets (second)	1	2	3	4	5	6	7	8	9	10
RGB	ClsNet-only	11.8	17.2	21.3	24.9	27.9	28.7	29.5	30.0	30.4	30.7
	StartNet-CE	13.7	20.7	23.8	27.2	29.4	30.7	31.9	32.5	33.2	33.6
	StartNet-PG	**15.9**	**21.0**	**24.8**	**28.4**	**30.7**	**31.8**	**33.0**	**33.5**	**34.0**	**34.4**
Two Stream	ClsNet-only	13.9	21.6	25.8	28.9	31.1	32.5	33.5	34.3	34.8	35.2
	StartNet-CE	17.4	25.4	29.8	33.0	34.6	36.3	37.2	37.7	38.6	38.8
	StartNet-PG	**19.5**	**27.2**	**30.8**	**33.9**	**36.5**	**37.5**	**38.3**	**38.8**	**39.5**	**39.8**

Table 3. Ablation study of our framework using p-mAP at depth Rec=1.0 on THUMOS'14. LSTM is used to implement ClsNet. Different offset thresholds are used to evaluate our framework with different features. Best performance is marked in bold.

Features	Depth Rec.	@0.1	@0.2	@0.3	@0.4	@0.5	@0.6	@0.7	@0.8	@0.9	@1.0
RGB	ClsNet-only	71.2	61.1	52.8	47.0	42.0	37.7	34.0	30.6	27.5	25.3
	StartNet-CE	73.2	64.5	56.8	50.2	45.1	40.5	36.6	33.5	30.5	27.7
	StartNet-PG	**73.6**	**65.0**	**58.0**	**51.2**	**45.9**	**41.5**	**37.8**	**34.3**	**31.5**	**28.8**
Two Stream	ClsNet-only	71.3	63.0	56.9	52.0	46.9	42.3	38.7	35.0	31.8	29.2
	StartNet-CE	72.7	65.6	60.2	55.3	51.0	46.8	43.0	39.2	36.0	32.9
	StartNet-PG	**77.4**	**70.2**	**64.5**	**59.1**	**54.2**	**49.3**	**45.1**	**41.2**	**37.6**	**34.2**

Table 4. Ablation study of our framework using average p-mAP at different depths on THUMOS'14. At each depth, we average p-mAP over offset thresholds from 1 to 10 seconds. LSTM is used to implement ClsNet. Best performance is marked in bold.

Effectiveness of LocNet. The results from ClsNet alone can be used to generate action starts by following the *action start generation procedure* in late fusion. To evaluate the contribution of LocNet, we construct *ClsNet-only* by removing LocNet from our framework. Results of *ClsNet-only* can also demonstrate the performance of OAD methods if applied on the ODAS task directly. As shown in Table 3, *ClsNet-only* has already achieved good results, outperforming *C3D* based methods. When adding *LocNet*, *StartNet-PG* improves *ClsNet-only* by 5%-6% p-mAP with TS feature and by 4%-5% p-mAP with RGB features under varying offsets. We can also observe a trend that the gaps between *StartNet-PG* and *ClsNet-only* are larger when the offset is smaller. As shown in Table 4, *StartNet-PG* outperforms *ClsNet-only* by 5%-6% p-mAP with TS features and about 3%-5% p-mAP with RGB features at different depths. The qualitative comparison in Fig. 3 shows an example that *ClsNet-only* generates a false positive at the last frame. It may be because that the frame contains a classic appearance of the action, *i.e.*, *Basketball Dunk*. With the help of LocNet, the false positive is corrected by *StartNet-PG*.

Effectiveness of long-term planning. In order to investigate the effect of long-term planning, we replace the policy gradient training strategy with simple cross-entropy loss – $-\beta g_t log(s_t) - (1-g_t)log(1-s_t)$ – such that every frame is considered independently. This baseline is referred as *StartNet-CE*. Similar to *StartNet-PG*, weight factor, β, is used to handle sample imbalance. Same as α in Eq. 4, we set β equal to the ratio between the number of negative samples and positive ones. As shown in Table 3 and 4, *StartNet-PG* significantly outperforms *StartNet-CE* under each offset threshold and at different depths, which proves the usefulness of the long-term planning.

In order to further investigate effects of parameter settings for LocNet, we conduct an ablation study on different values of the length of historical decision vector, n, and gamma in Eq. 5 when offset threshold is set to 1 second and depth Rec=1.0. Results are shown in Fig. 4. Increasing the length of the historical decision vector means increasing the dependency of later decisions on previous ones. As is shown, the model performs much better when incorporating historical decisions and it reaches its highest performance when 8 historical decisions are considered. Increasing gamma indicates increasing the effect of future rewards to the total long-term reward. It shows that when increasing values of gamma, the model performs better.

Results with different features. To investigated the performance of our framework when using different features, we add experiments with *ClsNet-only*, *StartNet-CE* and *StartNet-PG* using appearance features (RGB) only. Results are displayed in Table 3 and Table 4. We see that when using only RGB features, performance of the three models drops. However, even with RGB features, our method still outperforms *Shou et al.* [31] largely.

Effectiveness of two-stage design. We validate our two-stage design by comparing with *one-stage network* which has similar structure as ClsNet (LSTM) except that we modify it to directly predict action starts for all classes and optimize it with cross-entropy loss. We get 6.5% and 10.2% p-mAP at 1 second offset (depth Rec=1.0) using RGB and TS features, respectively. The results are much worse than *StartNet-CE* and *StartNet-PG* (drops about 7% and 9%), demonstrating that simply learning classification and localization of action starts jointly is not a good strategy.

Learning from low-level features. Our framework uses action score distributions pretrained on an auxiliary task as inputs of LocNet. We believe that learning from this high-level representation is better than learning from low-level noisy features for our task due to the lack of training data.

	Offsets (second)	1	2	3	4	5	6	7	8	9	10
Baselines	SceneDetect [1]	–	–	–	–	–	–	–	–	–	4.7
	ShotDetect [2]	–	–	–	–	–	–	–	–	–	6.1
	Shou et al. [31]	–	–	–	–	–	–	–	–	–	8.3
StartNet	ClsNet-only-VGG	2.7	4.1	5.1	5.9	6.7	7.5	8.1	8.7	9.2	9.8
	StartNet-CE-VGG	4.2	6.1	7.4	8.7	9.7	10.5	11.4	12.0	12.6	13.1
	StartNet-PG-VGG	**6.0**	**7.6**	**8.8**	**9.8**	**10.7**	**11.5**	**12.2**	**12.6**	**13.1**	**13.5**
	ClsNet-only-TS	4.2	6.1	7.7	8.8	9.8	10.7	11.3	12.2	13.0	13.6
	StartNet-CE-TS	6.0	8.3	10.1	11.7	12.9	13.9	15.0	15.8	16.7	17.5
	StartNet-PG-TS	**8.1**	**10.2**	**11.8**	**13.3**	**14.4**	**15.3**	**16.1**	**16.7**	**17.4**	**18.0**

Table 5. Comparisons using p-mAP under various offset thresholds at depth Rec=1.0 on ActivityNet. ClsNet is implemented with LSTM. Numbers of baseline methods are cited from [31]. – indicates that numbers are not provided in [31].

To prove this point, we construct *StartNet-img* where LocNet learns directly from the low-level image features. The p-mAP using RGB and TS features under offsets of 1 second (depth is 1.0) is 10.2% and 14.0%, respectively, which much under perform our framework (drops about 5%).

Efficiency analysis. We test our method with a single Quadro P6000 GPU. It takes 8ms and 0.3ms on average to forward pass ClsNet(C3D) and LocNet. When using ClsNet (LSTM-TS), LSTM takes 0.3ms. The bottleneck is RGB and motion feature extraction including flow computation with FlowNet-V2 (97ms). Even so, our method can process each frame within 0.1s in total. One can reduce time largely by using real-time flow extractors, e.g. PWC-Net [34].

4.2. Experiments on ActivityNet

Dataset. ActivityNet v1.3 [11] is one of the largest datasets for action recognition. It contains annotations of 200 action classes. There are around 10K untrimmed videos (15K action instances) in the training set and 5K (7.6K action instances) untrimmed videos in the validation set. Averagely, there are around 1.6 action instances in each video. Following [31], we train our models on the train set and test them on the validation set.

Feature description. TS feature is constructed by concatenating appearance and motion features that are extracted from TSN model (with BN-Inception) [37] pretrained on Kinetics [6]. Besides, we validate our method using appearance features extracted from *fc6* layer of VGG-16 [33]. The VGG-16 model is pretrained on ImageNet [10]. VGG-16 features are not as good as ResNet and InceptionNet features for action recognition tasks. We use VGG-16 features to show that our framework can produce reasonable results even when using simple features pretrained only on images.

Training sample strategy of LocNet. Unlike THUMOS'14 which contains around 16 action instances per video in average, ActivityNet has only one action instance in most of the videos. Thus, ActivityNet has much severer imbalance problem between start and non-start classes. To balance the samples, we randomly select equal numbers of positive and negative sequences for each training batch. Positive sequence is defined as containing at least one action start and negative one contains no action start. Then, α is set to the ratio between the number of negative samples over the number of positive ones after the sample balance.

Evaluation results. Comparisons of StartNet with previous methods on ActivityNet are shown in Table 5. StartNet significantly outperforms previous methods. Specifically, StartNet with TS feature achieves similar performance under 1 second offset tolerance compared to *Shou et al.* [31] under 10 seconds offset. At offset of 10 seconds, our method improves *Shou et al.* [31] by around 10%. It also outperforms *SceneDetect* and *ShotDetect* largely by 13.3% and 11.9%, respectively. Even with VGG features pretrained on only images, our method significantly outperforms the state-of-the-arts. Besides, we demonstrate the contribution of each module by comparing with *ClsNet-only* and *StartNet-CE*. Results show that by adding LocNet, *StartNet-PG* improves *ClsNet-only* by over 3% (using VGG) and around 4% (using TS) p-mAP. With long-term planning, *StartNet-PG* significantly outperforms *StartNet-CE* under both features, especially when the offset tolerance is small. Qualitative results in Fig. 3 shows a hard case where *ClsNet-only* misses an action start due to the subtle appearance difference near the start point. With LocNet, *StartNet-PG* successfully captures the start point although the score is low.

5. Conclusion

We proposed StartNet to handle Online Detection of Action Starts. StartNet consists of two networks, *i.e.*, ClsNet and LocNet. ClsNet processes the input streaming video and generates action scores for each video frame. LocNet localizes start points by optimizing long-term planning rewards using policy gradient methods. At the end, results from the two sub-networks are fused to produce the final action start predictions. Experimental results on THUMOS'14 and ActivityNet demonstrate that our framework significantly outperforms the state-of-the-arts. Extensive ablation studies were conducted to show the effectiveness of each module of our method.

References

[1] https://github.com/Breakthrough/PySceneDetect. 6, 8

[2] https://github.com/johmathe/Shotdetect. 6, 8

[3] http://pytorch.org/. 5

[4] Shyamal Buch, Victor Escorcia, Chuanqi Shen, Bernard Ghanem, and Juan Carlos Niebles. SST: Single-stream temporal action proposals. In *CVPR*, 2017. 1, 2

[5] Juan C Caicedo and Svetlana Lazebnik. Active object localization with deep reinforcement learning. In *ICCV*, 2015. 2

[6] Joao Carreira and Andrew Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *CVPR*, 2017. 8

[7] Yu-Wei Chao, Sudheendra Vijayanarasimhan, Bryan Seybold, David A Ross, Jia Deng, and Rahul Sukthankar. Rethinking the faster r-cnn architecture for temporal action localization. In *CVPR*, 2018. 1, 2

[8] Xiyang Dai, Bharat Singh, Guyue Zhang, Larry S Davis, and Yan Qiu Chen. Temporal context network for activity localization in videos. In *ICCV*, 2017. 1, 2

[9] Roeland De Geest, Efstratios Gavves, Amir Ghodrati, Zhenyang Li, Cees Snoek, and Tinne Tuytelaars. Online action detection. In *ECCV*, 2016. 1, 2, 3

[10] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *CVPR*, 2009. 8

[11] Bernard Ghanem Fabian Caba Heilbron, Victor Escorcia and Juan Carlos Niebles. Activitynet: A large-scale video benchmark for human activity understanding. In *CVPR*, 2015. 2, 5, 8

[12] Jiyang Gao, Zhenheng Yang, and Ram Nevatia. RED: Reinforced encoder-decoder networks for action anticipation. In *BMVC*, 2017. 1, 2, 3, 5

[13] Jiyang Gao, Zhenheng Yang, Chen Sun, Kan Chen, and Ram Nevatia. TURN TAP: Temporal unit regression network for temporal action proposals. *ICCV*, 2017. 1, 2

[14] Mingfei Gao, Ruichi Yu, Ang Li, Vlad I Morariu, and Larry S Davis. Dynamic zoom-in network for fast object detection in large images. In *CVPR*, 2018. 2

[15] Ross Girshick. Fast R-CNN. In *ICCV*, 2015. 2

[16] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*, 2014. 2

[17] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 2, 6

[18] Minh Hoai and Fernando De la Torre. Max-margin early event detectors. In *IJCV*, 2014. 2

[19] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 1997. 3, 6

[20] Jingjia Huang, Nannan Li, Tao Zhang, Ge Li, Tiejun Huang, and Wen Gao. SAP: Self-adaptive proposal model for temporal action detection based on reinforcement learning. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018. 2

[21] Haroon Idrees, Amir R Zamir, Yu-Gang Jiang, Alex Gorban, Ivan Laptev, Rahul Sukthankar, and Mubarak Shah. The THUMOS challenge on action recognition for videos "in the wild". *CVIU*, 2017. 2, 5

[22] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv:1502.03167*, 2015. 6

[23] Andrej Karpathy, George Toderici, Sanketh Shetty, Thomas Leung, Rahul Sukthankar, and Li Fei-Fei. Large-scale video classification with convolutional neural networks. In *CVPR*, 2014. 5

[24] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv:1412.6980*, 2014. 5

[25] Tianwei Lin, Xu Zhao, Haisheng Su, Chongjing Wang, and Ming Yang. BSN: Boundary sensitive network for temporal action proposal generation. In *ECCV*, 2018. 6

[26] Shugao Ma, Leonid Sigal, and Stan Sclaroff. Learning activity progression in lstms for activity detection and early detection. In *CVPR*, 2016. 2

[27] Volodymyr Mnih, Nicolas Heess, Alex Graves, et al. Recurrent models of visual attention. In *NIPS*, 2014. 4

[28] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. 2015. 2

[29] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In *NIPS*, 2015. 2

[30] Zheng Shou, Jonathan Chan, Alireza Zareian, Kazuyuki Miyazawa, and Shih-Fu Chang. CDC: Convolutional-deconvolutional networks for precise temporal action localization in untrimmed videos. In *CVPR*, 2017. 2

[31] Zheng Shou, Junting Pan, Jonathan Chan, Kazuyuki Miyazawa, Hassan Mansour, Anthony Vetro, Xavier Giro-i Nieto, and Shih-Fu Chang. Online action detection in untrimmed, streaming videos-modeling and evaluation. In *ECCV*, 2018. 1, 2, 5, 6, 7, 8

[32] Zheng Shou, Dongang Wang, and Shih-Fu Chang. Temporal action localization in untrimmed videos via multi-stage cnns. In *CVPR*, 2016. 1, 2

[33] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv:1409.1556*, 2014. 8

[34] Deqing Sun, Xiaodong Yang, Ming-Yu Liu, and Jan Kautz. PWC-Net: Cnns for optical flow using pyramid, warping, and cost volume. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8934–8943, 2018. 8

[35] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018. 2, 4

[36] Du Tran, Lubomir Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. Learning spatiotemporal features with 3d convolutional networks. In *ICCV*, 2015. 3, 6

[37] Limin Wang, Yuanjun Xiong, Zhe Wang, Yu Qiao, Dahua Lin, Xiaoou Tang, and Luc Van Gool. Temporal segment networks: Towards good practices for deep action recognition. In *ECCV*, 2016. 6, 8

[38] Zuxuan Wu, Tushar Nagarajan, Abhishek Kumar, Steven Rennie, Larry S Davis, Kristen Grauman, and Rogerio Feris. Blockdrop: Dynamic inference paths in residual networks. In *CVPR*, 2018. 2

[39] Zuxuan Wu, Caiming Xiong, Chih-Yao Ma, Richard Socher, and Larry S Davis. Adaframe: Adaptive frame selection for fast video recognition. *arXiv:1811.12432*, 2018. 2, 4, 5

[40] Huijuan Xu, Abir Das, and Kate Saenko. R-C3D: Region convolutional 3d network for temporal activity detection. In *ICCV*, 2017. 2

[41] Mingze Xu, Mingfei Gao, Yi-Ting Chen, Larry S Davis, and David J Crandall. Temporal recurrent networks for online action detection. *ICCV*, 2019. 1, 2, 3, 5

[42] Yue Zhao, Yuanjun Xiong, Limin Wang, Zhirong Wu, Xiaoou Tang, and Dahua Lin. Temporal action detection with structured segment networks. In *ICCV*, 2017. 1, 2, 6