SAP: Self-Adaptive Proposal Model for Temporal Action Detection based on Reinforcement Learning

Jingjia Huang and **Nannan Li** and **Tao Zhang** and **Ge Li** and **Tiejun Huang** and **Wen Gao**
School of Electronic and Computer Engineering, Shenzhen Graduate School, Peking University
Lishui Road 2199, Nanshan District, Shenzhen, China 518055
jjhuang@pku.edu.cn, linn@pkusz.edu.cn, t_zhang@pku.edu.com,geli@ece.pku.edu.cn,
tjhuang@pku.edu.cn, wgao@pku.edu.cn

Abstract

Existing action detection algorithms usually generate action proposals through an extensive search over the video at multiple temporal scales, which brings about huge computational overhead and deviates from the human perception procedure. We argue that the process of detecting actions should be naturally one of observation and refinement: observe the current window and refine the span of attended window to cover true action regions. In this paper, we propose a *Self-Adaptive Proposal* (SAP) model that learns to find actions through continuously adjusting the temporal bounds in a self-adaptive way. The whole process can be deemed as an agent, which is firstly placed at the beginning of the video and traverse the whole video by adopting a sequence of transformations on the current attended region to discover actions according to a learned policy. We utilize reinforcement learning, especially the Deep Q-learning algorithm to learn the agent's decision policy. In addition, we use temporal pooling operation to extract more effective feature representation for the long temporal window, and design a regression network to adjust the position offsets between predicted results and the ground truth. Experiment results on THUMOS'14 validate the effectiveness of SAP, which can achieve competitive performance with current action detection algorithms via much fewer proposals.

Introduction

Temporal action detection requires not only to determine whether an action occurs in a video but also to locate the temporal extent of when it occurs, which is a challenging problem for real-life long untrimmed videos. Most of modern approaches (Shou, Wang, and Chang 2016; Zhu and Newsam 2016; Xiong et al. 2017) usually solve the problem via a two-step pipeline: firstly generate a set of class independent action proposals, which are obtained via running a action/background classifier over a video at multiple temporal scales; then the proposals are classified by the pre-trained action detector, and post processing such as non-maximum suppression is applied. However, such extensive search for action localization is unsatisfying in terms of both accuracy and computational efficiency. Like the human detects the action through successively altering the span of attended region to narrow down the difference between the bounds of current window and that of true action region, the optimal algorithm should be the process of sequential, iterative observation and refinement consuming search steps as less as possible.

In this paper, we propose a class-specific action detection model called SAP that learns to continuously adjust the current region to cover the groundtruth more precisely in a self-adapted way. This is achieved by applying a sequence of transformations to a temporal window that is initially placed at the beginning of the video and finally finds and covers action region as large as possible. The sequence of transformation is decided by an agent that analyzes the content of the current attended region and select the next best action according to a learned policy, which is trained via reinforcement learning based on Deep Q-Learning algorithm (Mnih et al. 2015). Different from existing approaches that locate the action following a fixed path, our method generates various search trajectories for different action instances, depending on the video scenarios, the starting search position and the sequences of actions adopted. As a result, the trained agent will locate a single instance of an action in about 15 steps, which means that the model only processes 15 successive regions of an image to explore an uncover video segment, thus it is of great computational efficiency to compare with sliding window based approaches.

Our model draws the inspiration from works that have used reinforcement learning to build active models for object localization in image (Caicedo and Lazebnik 2015; Jie et al. 2016; Bellver et al. 2016). However, we can not handle the video in a top-down way that is proved to perform effectively for image object localization, as the duration of the video is usually too long (from hundreds to thousands frames). We start the search from a position initially placed at the beginning of the video, which will terminate until a instance of action has been found or the maximum transformation steps has been reached, and then a new search begins from the position on the right side away from current attended region. We incorporate temporal pooling operation with feature extraction process to better represent the long video segment and design a "*jump*" action to avoid the agent trapping itself in the region where no action occurs. We conducted a comprehensive experimental evaluation in the challenging THUMOS'14 dataset (Jiang et al. 2014), and the results demonstrate that SAP can achieve competitive performance

in terms of precision and recall via a small number of action proposals.

Related work

Action Recognition. This task has been attended for a few years, and a large amount of research work have been done (Laptev and Lindeberg 2003; Wang and Schmid 2013; Simonyan and Zisserman 2014; Tran et al. 2015). In early years, researchers often tackle the problem based on hand-crafted visual features (Laptev and Lindeberg 2003; Wang and Schmid 2013). Recently, impressed by the huge success of deep learning on image analysis task, some approaches have introduced deep models, especially Convolutional Neural Network (CNN), for better excavation the spatial-temporal information included in the video clip. Simonyan and Zisserman (Simonyan and Zisserman 2014) propose the two-stream network architecture with one branch processing RGB signal and the other one dealing with optical-flow signal. Tran *et al.* (Tran et al. 2015) construct C3D model, which operates 3D convolution in spatio-temporal video volume directly and integrates appearance and motion cues for better feature representation. There have been also other efforts (Donahue et al. 2015; Yue-Hei Ng et al. 2015) that attempt to combine frame-level CNN feature representation and long-range temporal structure to cope with input videos of long duration. Up to now, deep learning based approaches have achieved state-of-the-art performances.

Temporal Action Detection. Different from action recognition where actions are included in a trimmed video clip and the aim is to predict the category, temporal action detection needs to not only classify the action but also give out temporal localization. Most existing approaches address the problem via sliding window strategy for candidates generation and focus on feature representation and classifier construction (Shou, Wang, and Chang 2016; Gaidon, Harchaoui, and Schmid 2013; Oneata, Verbeek, and Schmid 2013; Yuan et al. 2016). Shou *et al.* (Shou, Wang, and Chang 2016) utilize a multi-stage CNN detection network for action localization, where background windows are first filtered out by a binary action/background classifier based on C3D feature, then an action detection network incorporated both classification loss and temporal localization loss is trained for candidate refinement. By the limitation of 16-frames input of C3D model, they select 16 frames in uniform from the whole video, which is inferior to temporal pooling operation utilized in our approach. Gao *et al.* (Gao et al. 2017) decompose the input video into short video units, and pool features extracted from a set of contiguous units for representation of long video clip, and meanwhile employ a coordinate regression network to refine the temporal action boundaries. Our approach also includes location regression, whose regression offsets are calculated via the relative deviation rather than the absolute value, thus it will facilitate the model to converge more efficiently. Unlike the works mentioned above, Yeung *et al.* (Yeung et al. 2016) propose an attention based model that predicts the action position through a few of glimpses, which is trained via reinforcement learning. The difference between their work and ours is that our approach locates the action through continuously adjusting the span of current window not predicting the bounds directly.

Object Detection. Most of recent approaches for object detection are built upon the paradigm of "*proposal + classification*" (Girshick et al. 2014; Ren et al. 2015). Object proposals are usually either generated by methods relied on hand-crafted low-level visual cues, such as SelectiveSearch (Uijlings et al. 2013) and Edgebox (Zitnick and Dollár 2014), or produced by fully convolutional network implemented on CNN features extracted from anchor boxes arranged uniformly on the image, such as Faster R-CNN (Ren et al. 2015). However, generating too many proposals for a image with only one or two objects is unnecessary and computational inefficiency. Some works attempt to reduce the number of proposals with an active object detection strategy (Caicedo and Lazebnik 2015; Jie et al. 2016; Mathe, Pirinen, and Sminchisescu 2016). Caicedo *et al.* (Caicedo and Lazebnik 2015) learns an optimal policy to locate one single object in the image via Deep Q-Learning, where it starts from the whole image in a top-down way and adaptively adjusts the window scale and position to focus on the true region. Jie *et al.* (Jie et al. 2016) propose an effective tree-structured reinforcement learning approach, which learns to balance the exploration of uncovered new objects and the refinement of covered ones, and can localize multiple objects in a single run. Inspired by (Caicedo and Lazebnik 2015; Jie et al. 2016), we design a reinforcement learning based approach for temporal action localization, which locates action instances within the long untrimmed video via the learned policy in a bottom-up way, and meanwhile utilizes a regression network to refine the predicted temporal window boundaries.

Self-Adaptive Proposal Model

In this section, we present *Self-Adaptive Proposal* (SAP) model, which is self-adapted and will gradually adjust its predicted results according to the content of attended window and the history of executed actions to cover the true action region as accurate as possible in a few steps. We cast the problem of temporal action localization as a Markov Decision Process (MDP), in which the agent interacts with the environment and makes a sequence of decisions to achieve the settled goal. In our formulation, the environment is the input video clip, in which the agent has an observation of the current video segment, called temporal window, and restructures the position or span of the window, to achieve the goal of locating the action precisely. The agent receives positive or negative rewards after each decision made during the train phase to learn an effective policy. Besides, we construct a regression network to refine the final detection results to promote the accuracy of localization. The framework of our proposal generation model is illustrated in Fig. 1. In the following subsections, the set of actions A, the set of states S, and the reward function $R(s, a)$ of MDP and the regression network are discussed in detail. To avoid confusion, the action performed by the actor in the video is called motion in this section, and in other sections the meaning of action is determined by the context.

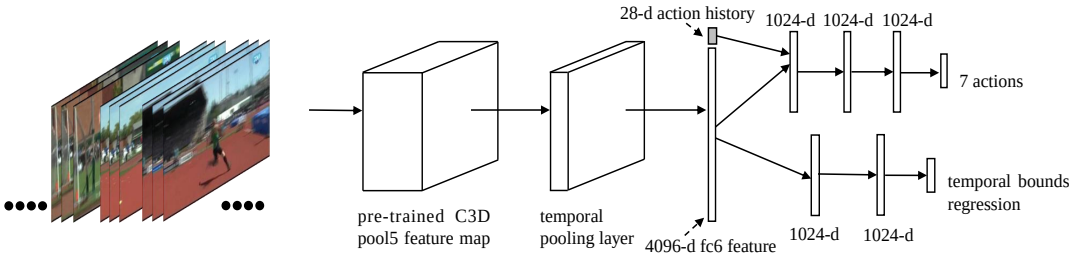


Figure 1: The framework of SAP model based on Deep Q-learning, which incorporates a regression network for better action localization.

MDP Formulation

Actions. The set of actions A can be divided into two categories: one group for transformation on temporal window, such as "*move left*", "*move right*", "*expand left*", and the remaining one for terminating the search, "*trigger*", as shown in Fig. 2. The transformation group includes regular actions that comprises of translation and scale, and one irregular action. The regular actions vary the current window in terms of position or time span around the attended region, such as "*move left*", "*expand left*" or "*shrink*", which are adopted by the agent to increase the intersection with the groundtruth that has overlaps with the current window. The irregular action, namely "*jump*", translates the window to a new position away from the current site to avoid that the agent traps itself round the present location when there is no motion occurring nearby. The change caused by any regular actions at each time to the window equals to a value in proportion to the current window size. For instance, supposing that current window is denoted as $[x_l, x_r]$, where x_l and x_r stand for the left and right boundary respectively. The action "*move left*" translates the window to a new site of $[x_{l'}, x_{r'}]$ with $x_l - x_{l'} = x_{r'} - x_r = \alpha * (x_r - x_l)$, while for action "*expand left*" scales the window with the change of $x_l - x_{l'} = \alpha * (x_r - x_l)$ and $x_{r'} = x_r$. Here, $\alpha \in [0, 1]$ is a parameter that can give a trade-off between search speed and localization accuracy. In this paper, we set $\alpha = 0.2$. The action "*jump*" selects a new window randomly from the left or right side, which has the same size with the current window, being a distance away from the present site. The regular actions make the agent gradually adjust its position to cover the motion more accurately when it has found one; while the action "*jump*" let the agent explore unknown region that may contain the motion in a discontinuous and efficient way. The action "*trigger*" is employed by the agent whenever it considers that a motion has been localized by the current window, and stops the sequence of current search, and restarts a new search for the next motion with an initial window position away from current site.

State. The state of MDP is the concatenation of two components: the presentation of current window and the history of taken actions. To describe the motion within current window generally, the feature extracted from the C3D CNN model (Tran et al. 2015), which is pretrained on Sports-1M and finetuned on UCF-101, is utilized as the presentation. Here, we choose the feature vector from *fc6* layer (4096 dimension) in our problem, consideing its good abstract representation for the semantic information about the motion. The original C3D model can only accept 16 frames as input, however, the duration of temporal window is always far more than that number. To tackle with the problem, we design two different solutions: i.) uniformly select 16 frames from the whole duration ; ii.) fed all the frames into the C3D model and add a additional pooling layer (average pooling for our problem) between the "*pool5*" layer and the "*fc6*" layer, which condenses the dimension of extracted feature vector from "*pool5*" to the value specified by the C3D model. The history of the taken actions is a binary vector that tells which action has been adopted by the agent in the past. Each action in the history is represented by a 7-dimension binary vector where all the values are zero except the one corresponding to the taken action. In the experiment, we totally record 4 past actions as the history. The history of taken actions informs the agent the search path that has been passed through and the regions already attended, so as to stabilize search trajectories that might get stuck in repetitive cycles.

Reward Fuction. The reward function $R(s, a)$ provides a feedback to the agent when it performs the action a at the current state s, which awards the agent for actions that will bring about the improvement of motion localization accuracy while gives the punishment for actions that leads to the decline of the accuracy. The quality of motion localization is evaluated via the simple yet indicative measurement, Intersection over Union (IoU) between current attended temporal window and the groundtruth. Supposing that w stands for the current window and g represents the groundtruth region of motion, then the IoU between w and g is defined as $\text{IoU}(w, g) = \text{span}(w \cap g) \,/\, \text{span}(w \cup g)$. The reward function is proportional to the difference between IoUs of two successive states s and s', where the agent moves to state s' from s by executing the action a. Specially, it is formulated as following:

$$r(s, a) = \max_{1 \le i \le n} \text{sign}(\text{IoU}(w', g_i) - \text{IoU}(w, g_i)), \quad (1)$$

where w' and w are attended windows corresponding to state s' and s respectively, n is the number of groundtruths within the input video. The reward function returns +1 or -1. Equation 1 indicates that the agent receives the reward +1 if the new window w' has more overlap with any of the groundtruth than the previous window w, while the

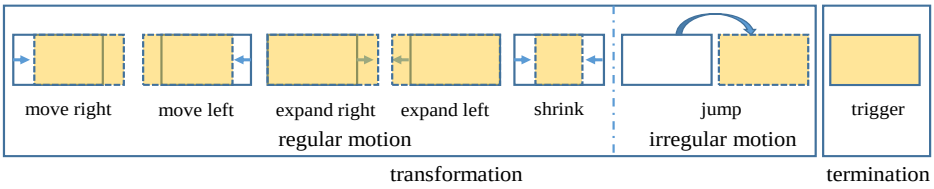


Figure 2: Illustrations of the actions adopted by the agent for motion search in our experiment. White boxes with solid line stand for current attended temporal window; while yellow boxes with dash line represent the resulting temporal window after taking the corresponding motion.

reward -1 otherwise. Such binary reward value makes the agent clearly realize that at present state, which action drives the attended window towards the groundtruth, and thus accelerates the convergence rate of the model during training phase. In addition, such reward-function scheme facilitates better localization towards motion regions especially for the video with multiple motion instances, as there is no limitation on which motion should be focused on at each state. The "*trigger*" action has a different reward function scheme, as it leads to the termination of search and there is no next state. The reward of "*trigger*" is determined by a piecewise function of IoU threshold, which can be presented as following:

$$r_e(s) = \begin{cases} +\eta & \text{if IoU}(w, g) \geq \tau \\ -\eta & \text{otherwise.} \end{cases} \quad (2)$$

In equation 2, e represents the "*trigger*" action, η is the reward value and chosen as 3 in our experiment, τ is the IoU threshold, which controls the tradeoff between the localization accuracy and computational overhead. The large τ will encourage the agent to locate the motion more precisely, however it consumes more action steps to complete the search. In training phase, we do not stop the search process when the agent correctly performs the action "*trigger*" for the first time, and let it continuously explore uncover regions. Therefore, our model recognizes many termination states that have IoU with groundtruth more than τ. We utilize $\tau = 0.5$ for our problem, and find that larger τ, such as 0.6 or 0.7, gives rise to negligible promotion on recall value, which is validated by the experiments.

Deep Q-learning. The goal of the agent is to maximize the sum of discounted rewards that are received through continuously transforming the current attended window during a sequence of interactions with the environment (an episode). In other words, the agent needs to learn a policy $\pi(s)$ that specifies an optimal action at current state s in the view of maximizing the long-term benefit. Due to the lack of state transition probability and the model free environment, we utilize reinforcement learning, specially Deep Q-learning, to estimate the optimal value for each state-action pair. In this paper, we follow the deep Q-learning framework proposed by Mnih *et al.* (Mnih et al. 2015) that estimates the action-value function via a neural network. The architecture of our Deep Q-Network (DQN) is illustrated as the up branch of Fig. 1. Similar to (Caicedo and Lazebnik 2015; Jie et al. 2016), the C3D CNN model is just used for feature extraction, and we do not train the whole pipeline for the full feature hierarchy learning, due to the good generalization of CNN model pretrained on large dataset and short of sufficient motion detection data for jointly training both two networks. During training phase, the agent operates multiple episodes with randomly initialized positions for each video clip. We train separate DQN for each motion category and follow the ϵ-greedy policy. Specially, the agent randomly selects an action from the whole action set with probability ϵ at current state, while greedily chooses the optimal action according to the learned policy with probability 1-ϵ. During the whole training epochs, ϵ is annealed linearly from 1.0 to 0.1, which gradually shifts from exploration to exploitation. Following (Mnih et al. 2015), we also incorporate the replay-memory scheme to collect various transition experiences from the past episodes, from which each record may be repeatedly used for model updates, in favor of breaking short-term correlations between states. A minibatch (*e.g.* 200 records) is randomly sampled from replay-memory as training samples to update the model at each time.

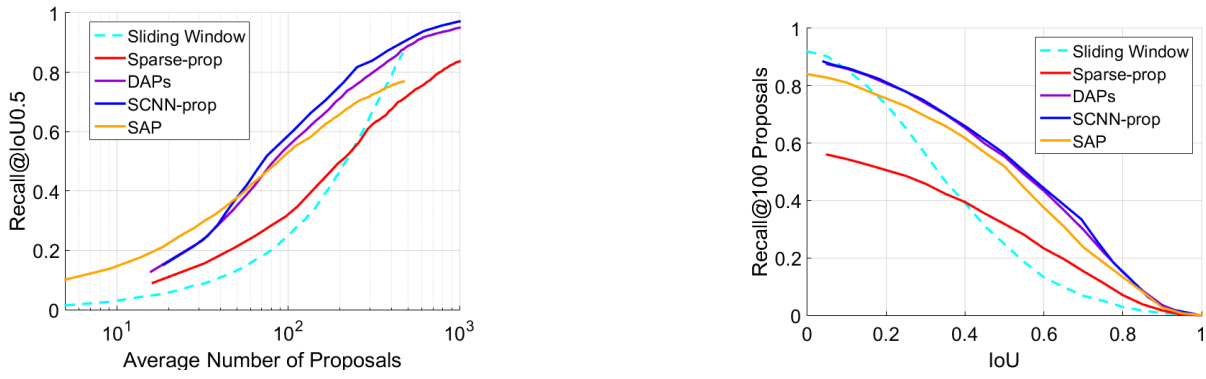
Regression Network

Inspired by Fast R-CNN (Girshick et al. 2014) where a regression network is incorporated to revise the position deviation between the predicted result and the groundtruth, we also introduce a regression model to refine the motion proposals. As shown in the down branch of Fig. 1, the regression channel accepts 4096-dimension feature vector as input and gives out two coordinate offsets on both starting and end moment. Unlike spatial bounding box regression, in which coordinate scaling is needed due to various camera-projection perspectives, we directly utilize original temporal coordinate (*i.e.* frame number) for offsets calculation leveraging the advantage of unified frame rate among video clips in our experiment. The regression biases are represented as the ratio of position deviation relative to the predicted span, which are defined as following:

$$o_s = (s_p - s_g)/(e_p - s_p), \quad o_e = (e_p - e_g)/(e_p - s_p), \quad (3)$$

where s_p and e_p are frame indexes for predicted starting and end moment, while s_g and e_g are frame indexes for the matched groundtruth. The loss for temporal coordinate regression, L_{reg}, is defined as following:

$$L_{reg} = \frac{1}{N_{end}} \sum_{i=1}^{N_{end}} (|o_{s,i}| + |o_{e,i}|), \quad (4)$$



(a) (b)

Figure 3: Evaluation results of recall performance on THUMOS'14. SCNN-prop and DAPs are state-of-the-art methods while *Sliding Window* in dash line is the baseline. We use the codes provided by (Escorcia et al. 2016) to calculate recalls

where N_{end} is the number of actions that correctly perform "$trigger$" in a minibatch. In other words, we only regress the position of temporal window whose IoU with groundtruth is more than 0.5. We utilize L_1 norm to make the loss be insensitive to outliers.

Experimental Results

We evaluate the performance of our SAP model[1] on the dataset THUMOS'14. Followed the standard evaluation protocol, our method achieves a competitive recall compared to the state-of-the-art results and outperforms the existing methods by a large margin for action detection task.

Implementation Details

Datasets. We validate the quality of SAP on labeled untrimmed videos from the challenging THUMOS'14, which contains over 20 hours of video from 20 sport action categories. The dataset comprises 413 videos with 200 for validation and 213 for test. We train SAP on validation set and report results on test set.

Training Details. SAP is implemented on Torch 7 (Collobert, Kavukcuoglu, and Farabet 2011). We train category specific model for each action and keep the same parameters settings. In pre-processing stage, we downsample the videos to have the same frame rate (30 frames/second), hence, the C3D model can extract more compact feature descriptors for input video sequence and the temporal regression network can operate more effectively on unified time basis. The replay memory buffer size is set as 2000, while the minibatch size is 200. The learning rate for DQN is 1e-3 with a decay rate of 5e-5, while the learning rate for regression network is 1e-4 with a decay rate of 9e-5. Dropout is applied with a ratio of 0.2. To accelerate training and avoid the agent getting itself trapped around temporal regions without any action instance, we force the agent to take a "*jump*" action towards left or right if the IoU for current window is zero, which will drive the window to the region around a groundtruth.

Testing Details. During test phase, the agent starts its search from the beginning of the video, and take actions to adjust the position of current attended temporal window. We set a maximum action steps for the agent as 15. The agent will restart its search from the right side of current window, if it takes a "*trigger*" action or finishes maximum action steps. Note that different from training, the agent consistently takes a leap towards right (two times farther than move "*left/right*") when it adopts a "*jump*" action. Therefore, the agent sequentially traverses the video from the start to end, which guarantees that different action instances existed within given video sequence will be visited. We choose the windows, where the agent takes "*trigger*" action, as proposals and utilize the pre-trained TSN (Wang et al. 2016) action detector as our classifier.

Temporal Proposals Evaluation

All the regions attended by the agent can be understood as temporal proposal candidates. Our methods run for about 400 steps with around 50 triggers averaged for each video. Fig.6 is an instance of the detection process of DQN. For each attended region, we score them with the Q-value predicted by the model, and add a large bonus only to "*trigger*" regions in order to give them higher priority when ranking the proposals. To assess the recall performance of our method, we use the metrics from (Escorcia et al. 2016):

Recall vs. Average Number of Proposal: average recall over all categories at IoU 0.5 is calculated as a function of average number of proposal. The best proposal approach is expected to achieve a higher recall with less proposals.

Recall vs. IoU: for a fixed number of proposals, recall is calculated at IoU between 0.05 to 1. To measure the localization quality of the top ranked proposals that are of most important for further recognition task, we fix the number of proposals to 100.

We compare SAP with DAPs (Escorcia et al. 2016), SCNN-prop (Shou, Wang, and Chang 2016), Sparse-prop (Heilbron, Niebles, and Ghanem 2016) and *sliding window*.

[1]The model implementations can be found on: https://github.com/hjjpku/Action_Detection_DQN

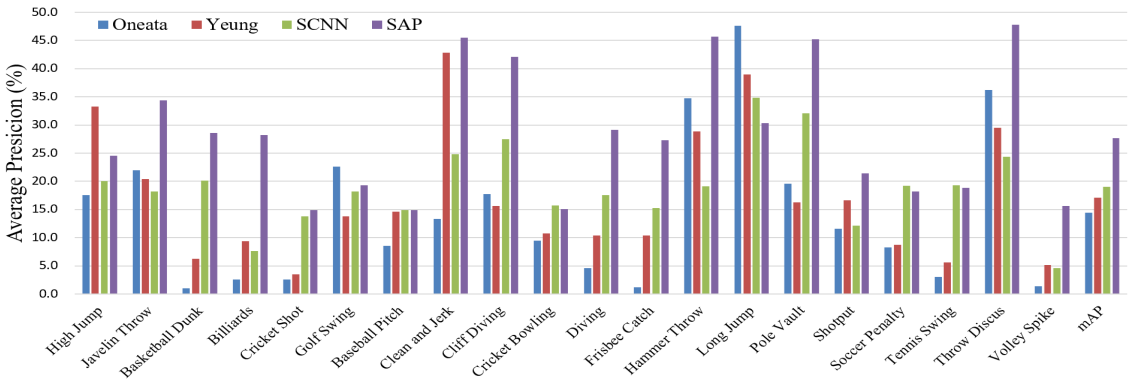


Figure 4: Histograms of average precision for each categories on THUMOS'14. The results are calculated with the official toolkit. The mAP(%) for Oneata *et al.*, Yeung *et al.*, SCNN and our SAP are 14.4, 17.1, 19.0 and 27.7 respectively.

SCNN-prop and DAPs are the state-of-the-art methods while *sliding window* is the baseline. For DAPs, SCNN-prop and Sparse-prop, we plot the curves using the proposal results provided by the authors. *sliding window* generates the proposals including all siding windows of lengths from 16 to 512 with 50% overlapping, and each window is scored with a random value. As shown in Fig.3.(a), our method achieves a better performance than the state-of-the-art methods in the early state of recall, and we have a competitive recall performance for the top 100 proposals according to Fig.3.(b). Notice that, the recall growth of our method slows down after about 70 proposals. It is because that our DQN agent tries to figure out the ground truth as fast as possible, and tends to stop exploration when it considers that an action region with IoU more than 0.5 has been found. Therefore, except the "*trigger*" segments, other proposals are intermediate results during the exploring process, which are unreliable on most occasions.

Temporal Action Detection Analysis

Following the convention (Jiang et al. 2014), we evaluate the performance of our SAP on the temporal localization task with mean Average Precision(mAP) score at 50% IoU. In the experiments, we take "*trigger*" windows as proposals and classify them with a pre-trained TSN. SAP is compared with other state-of-the-art methods in the literature, including CDC (Shou et al. 2017), SCNN (Shou, Wang, and Chang 2016), Oneata *et al.* (Oneata, Verbeek, and Schmid 2013) and Yeung *et al.* (Yeung et al. 2016). As is shown in Table 1, SAP outperforms the state-of-the-art approaches on THUMOS'14 by a large margin of 4.4%. The average precisions for each action category of various approaches are shown in Fig. 4. CDC is not included in Fig. 4 because the author did not publish the class specific results yet. Our approach behaves superiorly than other methods for most of categories, and especially for classes *Billiards, Basketball Dunk, Cliff Diving*, to name a few, the performance promotions are significant. The detection examples of action *PoleVault* are illustrated in Fig. 6.

Furthermore, to figure out the doubt whether the performance promotion of our approach is produced by employing a strong classifier, we conduct **contrast experiments** of using proposals at fixed number under a pre-trained TSN classifier for different proposal methods. As is shown in Table 1, with proposal number of 50, DAPs and SCNN-prop acquire mAPs at 6.99 and 16.9; while with proposal number of 100, the mAPs for DAPs and SCNN-prop will present at 9.16 and 12.4, respectively. These scores are far below our performance of mAP at 26.4 with 50 proposals. The compared results demonstrate that our approach can surely generate fewer proposals of high quality, which are mostly composed of true positives and are essential for the task of temporal action localization. For more experiment details, please consult our supplementary material.

To further analyze the contributions of different model components, namely temporal pooling and coordinate regression, for action detection task, we implement **ablation study**. We construct three models, which are described as follows:

• **SAP**: The integrated model with architecture shown in Fig. 1, where DQN agent generates proposals with the features processed through temporal pooling layer and finetunes the proposals with regression network.

• **SAP w/o POOLING**: The model without temporal pooling layer, uniformly samples video frames from the input video segment to extract C3D features and finetunes the proposals with regression network.

• **SAP w/o POOLING w/o RGN**: The basic model without both temporal pooling layer and regression network.

For each model, we evaluate the proposals and overall action localization performance. First of all, we use **Recall vs. Average Number of Proposal** at IoU=0.5 to evaluate the proposal performance that is shown in Fig. 5. Then we present the quantitative detection results of the models in Table 1 that are reported by mAP scores at 50% IoU. The mAPs are calculated with a fixed number of average proposals ($i.e.$ 50). Interestingly, it seems that **SAP w/o POOLING** has the superior recall performance than **SAP**, as shown in Fig. 5, however, **SAP** outperforms the other models by a large margin on overall detection performance. This is because **SAP** produces proposals with a small number of hard negatives, which allows the activity classifier to keep the num-

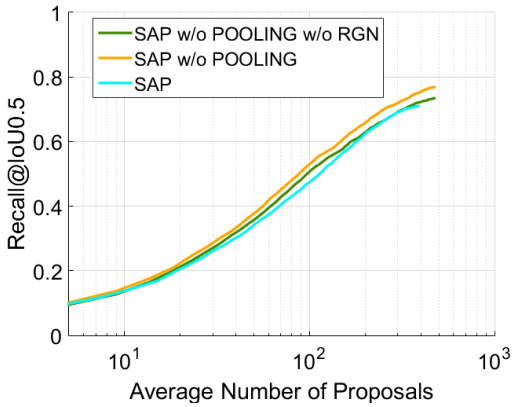


Figure 5: Recall Evaluation for ablation study: Recall vs. Average Number of Proposal at IoU=0.5

Model@Proposal Number	mAP(%)
DAPs+TSN @50	6.99
DAPs+TSN @100	9.16
SCNN-prop+TSN @100	12.4
SCNN-prop+TSN @50	16.9
SAP w/o POOLING w/o RGN@50	22.3
SAP w/o POOLING@50	24.6
SAP@50	**26.4**
Oneata *et al.*@NA	14.4
Yeung *et al.*@NA	17.1
SCNN @NA	19.0
CDC @NA	23.3
SAP@NA	**27.7**

Table 1: Temporal-action detection results evaluation: mAP calculated with a fixed number of average proposals on THUMOS'14. @NA means the proposal number is not specified for the methods.

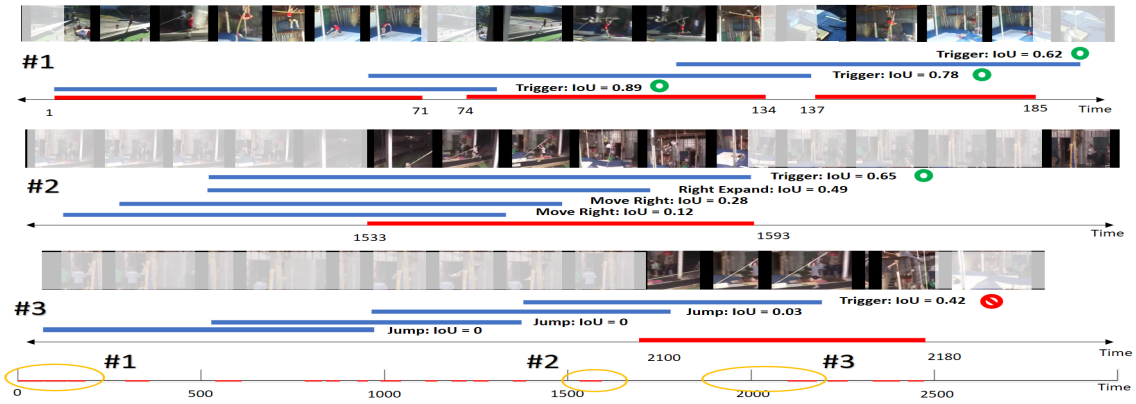


Figure 6: An instance of how DQN agent takes actions to generate proposals. The examples are sampled from the action *PoleVault* of THUMOS'14. The last row is the time shaft, where the red lines correspond to the ground truth. The top 3 rows are the running details corresponding to action instances #1, #2 and #3. The blue lines are the agent's search histories. A green circle indicates that it is a right "*trigger*" decision while the red one indicates a wrong one.

ber of false positive low. Besides, the results also illustrate that localization regression is of benefit to the detection task without exception.

Run-time Performance

The run-time property of our method is dependent on the DQN's performance. For a well trained DQN agent, it will concentrate on the ground truth in a couple of steps once it perceives the action segment. Meanwhile, it can also accelerate the exploring process over the video with "jump" action. Besides, the selection of scalar α is also an important factor that will influence the run-time performance. A large α will make the agent take a brief glance over the video in most of the case, but will also result in coarse proposals. As a trade off,we set the $\alpha = 0.2$ during the training and testing phase. During the test, we load the video first and then record the run-time of our model used to scan the video. On Tesla K80, the average run-time over all testing videos in THUMOS'14 is 266.6 FPS, including the online feature extraction.

Conclusion

In this paper, we have introduced an active action proposal model called SAP that learns to adaptively adjust the span of attended current window to cover the true action regions in a few steps. We build our model based on deep reinforcement learning and lean an optimal policy to direct the agent to act. In order to precisely locate the action, we design a regression network to revise the offsets between predicted bound results and the groundtruth. Experiment results on THUMOS'14 dataset validate that the proposed approach can achieve comparable performance with most of modern action-detection methods with much fewer action proposals.

Acknowledgments

This work was supported by National Science Foundation of China (No.U1611461), Shenzhen Peacock Plan (20130408-183003656) and National Science Foundation of China (61602014).

References

Bellver, M.; Giró-i Nieto, X.; Marqués, F.; and Torres, J. 2016. Hierarchical object detection with deep reinforcement learning. *arXiv preprint arXiv:1611.03718*.

Caicedo, J. C., and Lazebnik, S. 2015. Active object localization with deep reinforcement learning. In *Proceedings of the IEEE International Conference on Computer Vision*, 2488–2496.

Collobert, R.; Kavukcuoglu, K.; and Farabet, C. 2011. Torch7: A matlab-like environment for machine learning. In *BigLearn, NIPS Workshop*, number EPFL-CONF-192376.

Donahue, J.; Anne Hendricks, L.; Guadarrama, S.; Rohrbach, M.; Venugopalan, S.; Saenko, K.; and Darrell, T. 2015. Long-term recurrent convolutional networks for visual recognition and description. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2625–2634.

Escorcia, V.; Heilbron, F.; Niebles, J.; and Ghanem, B. 2016. Daps: Deep action proposals for action understanding. In *In European Conference on Computer Vision*, 768–784.

Gaidon, A.; Harchaoui, Z.; and Schmid, C. 2013. Temporal localization of actions with actoms. *IEEE transactions on pattern analysis and machine intelligence* 35(11):2782–2795.

Gao, J.; Yang, Z.; Sun, C.; Chen, K.; and Nevatia, R. 2017. Turn tap: Temporal unit regression network for temporal action proposals. *arXiv preprint arXiv:1703.06189*.

Girshick, R.; Donahue, J.; Darrell, T.; and Malik, J. 2014. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 580–587.

Heilbron, F. C.; Niebles, J. C.; and Ghanem, B. 2016. Fast temporal activity proposals for efficient detection of human actions in untrimmed videos. In *Computer Vision and Pattern Recognition*, 1914–1923.

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

Jie, Z.; Liang, X.; Feng, J.; Jin, X.; Lu, W.; and Yan, S. 2016. Tree-structured reinforcement learning for sequential object localization. In *Advances in Neural Information Processing Systems*, 127–135.

Laptev, I., and Lindeberg, T. 2003. Space-time interest points. In *9th International Conference on Computer Vision, Nice, France*, 432–439. IEEE conference proceedings.

Mathe, S.; Pirinen, A.; and Sminchisescu, C. 2016. Reinforcement learning for visual object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2894–2902.

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A. A.; Veness, J.; Bellemare, M. G.; Graves, A.; Riedmiller, M.; Fidjeland, A. K.; Ostrovski, G.; et al. 2015. Human-level control through deep reinforcement learning. *Nature* 518(7540):529–533.

Oneata, D.; Verbeek, J.; and Schmid, C. 2013. Action and event recognition with fisher vectors on a compact feature set. In *Proceedings of the IEEE International Conference on Computer Vision*, 1817–1824.

Ren, S.; He, K.; Girshick, R.; and Sun, J. 2015. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, 91–99.

Shou, Z.; Chan, J.; Zareian, A.; Miyazawa, K.; and Chang, S. F. 2017. Cdc: Convolutional-de-convolutional networks for precise temporal action localization in untrimmed videos.

Shou, Z.; Wang, D.; and Chang, S.-F. 2016. Temporal action localization in untrimmed videos via multi-stage cnns. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 1049–1058.

Simonyan, K., and Zisserman, A. 2014. Two-stream convolutional networks for action recognition in videos. In *Advances in neural information processing systems*, 568–576.

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

Uijlings, J. R.; Van De Sande, K. E.; Gevers, T.; and Smeulders, A. W. 2013. Selective search for object recognition. *International journal of computer vision* 104(2):154–171.

Wang, H., and Schmid, C. 2013. Action recognition with improved trajectories. In *Proceedings of the IEEE International Conference on Computer Vision*, 3551–3558.

Wang, L.; Xiong, Y.; Wang, Z.; Qiao, Y.; Lin, D.; Tang, X.; and Gool, L. V. 2016. Temporal segment networks: Towards good practices for deep action recognition. In *European Conference on Computer Vision*, 20–36.

Xiong, Y.; Zhao, Y.; Wang, L.; Lin, D.; and Tang, X. 2017. A pursuit of temporal accuracy in general activity detection. *arXiv preprint arXiv:1703.02716*.

Yeung, S.; Russakovsky, O.; Mori, G.; and Fei-Fei, L. 2016. End-to-end learning of action detection from frame glimpses in videos. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2678–2687.

Yuan, J.; Ni, B.; Yang, X.; and Kassim, A. A. 2016. Temporal action localization with pyramid of score distribution features. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 3093–3102.

Yue-Hei Ng, J.; Hausknecht, M.; Vijayanarasimhan, S.; Vinyals, O.; Monga, R.; and Toderici, G. 2015. Beyond short snippets: Deep networks for video classification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 4694–4702.

Zhu, Y., and Newsam, S. 2016. Efficient action detection in untrimmed videos via multi-task learning. *arXiv preprint arXiv:1612.07403*.

Zitnick, C. L., and Dollár, P. 2014. Edge boxes: Locating object proposals from edges. In *European Conference on Computer Vision*, 391–405. Springer.