Active Temporal Action Detection in Untrimmed Videos via Deep Reinforcement Learning

NANNAN LI†1, HUIWEN GUO‡2, YANG ZHAO3, THOMAS LI4, AND GE LI†1

1School of Electronic and Computer Engineering, Peking University Shenzhen Graduate School, Shenzhen 518055 China (e-mail: linn@pkusz.edu.cn, geli@pku.edu.cn)
2Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences, Shenzhen 518055 China (e-mail: hw.guo89@gmail.com)
3School of Computer and Information, Hefei University of Technology, Hefei 230001 China (e-mail: yzhao@hfut.edu.cn)
4Gpower semiconductor Inc., Suzhou China (e-mail: thomas.li@gpower-semi.com)

* means that the authors have equal contributions.

Corresponding author: Ge Li (e-mail: geli@ece.pku.edu.cn).

The work was partially supported by National Natural Science Foundation of China (61602014), Shenzhen Fundamental Research Program (JCYJ20170818141120312), National Natural Science Foundation of China and Guangdong Province Scientific Research on Big Data (U1611461).

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 in the video should be naturally one of observation and refinement: observe the current temporal window and refine the span of attended window to cover true action regions. In this paper, we propose an active action detection model that learns to search actions through continuously adjusting the bounds of temporal attended window in a self-adaptive way. The whole process can be deemed as an exploring procedure, where an agent is firstly placed at the beginning of the video and then traverses the whole video by adopting a sequence of transformations on the current attended window 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. Actually, we construct an end-to-end trainable framework for the action detection task, which includes a proposal generation network based on Deep Q-learning, and the classification and regression networks responsible for the action category prediction and the action location adjustment respectively. In addition, we design an LSTM structure upon extracted CNN features of sparsely sampled frames to generate the effective feature representations for video sequences of various durations. We evaluate the action proposal performance of our approach on THUMOS’14, and assess the generalization ability for unseen action categories on ActivityNet. We also compare the action detection performance of ours with other state-of-the-art methods on both datasets. Experiment results validate the effectiveness of the proposed approach, which can achieve comparative or superior performance than other action detection methods via much fewer proposals.

INDEX TERMS

Video analysis, temporal action detection, deep reinforcement learning, deep learning, self adaptive

I. INTRODUCTION

Human action detection has long been an important research topic in the computer vision, and is gaining more attention in recent years, especially along with the explosive growth of user-generated videos. It is crucial to video understanding and analyzing, and can be applied for many practical tasks, such as video recommendation, video surveillance, video highlighting, or human-robot interaction, etc. Gener-
models to tackle with action recognition in the past few years, therefore the question of “what” has been solved satisfactorily. Nevertheless, another question of “when”, which we encounter more frequently while dealing with numerous untrimmed videos from real life, has not been settled reasonably. In this paper, we focus on the problem of action recognition as well as localization in long, untrimmed videos, where the span of the actual activity only occupies a fraction of the entire video duration, with efficiency and accuracy.

At present, most of modern approaches [4], [5], [8] for action detection adopt a sequential pipeline consisting of two steps, relying on hand-crafted features like IDT [6] encoded with Fisher Vector [7], or deep-learning feature representations such as C3D [2]. Firstly, a set of class independent action proposals are generated, which are obtained via operating an action/background classifier over a video at multiple temporal scales; then the proposals are classified by the pre-trained action detectors, and some post-processings such as duration prior or non-maximum suppression (NMS) are applied to refine the obtained temporal boundaries of action instances. Although such a pipeline has achieved decent performance in recent years, the extensive search for action instances is unsatisfying in terms of both accuracy and computational efficiency, especially for videos of long duration. We argue that the process of action detection should be one of sequential, iterative observation and result refinement, which consumes search steps as few as possible. Given a video for observation, a human can start from any time spot of the entire duration, and consecutively adjusts the current attended temporal window to approach true regions of action instances, depending on the assessment of the sequential observations in the recent history. By such way, the task of action detection can be accomplished with far fewer search steps compared to existing exhaustive-search based methods.

Based on this intuition, we propose an active action detection model that learns to continuously alter the current attended window to cover the matched action instance more precisely in a self-adapted way. This is achieved by applying a sequence of transformations, such as moving, expanding or shrinking, to the current temporal window that is initially placed at the beginning of the video and proceeds to draw near to true locations of action instances in a sequence of exploration. We formulate the problem as a Markov Decision Process (MDP), where the agent analyzes the content of the current attended window and selects the best action according to a learned policy. We obtain the policy through reinforcement learning, especially based on Deep Q-Learning [9], where the agent evaluates the values for each possible action at current observed states, and adopts the action that maximizes the sum of discount rewards in the long term. The proposed approach is fundamentally different from most sliding-window based methods, and is an active attention-action strategy that pays attention to the content of current region, and transforms the attended temporal window in such a way that the groundtruth region of an action instance is progressively more focused [10]. Therefore, different from existing sliding-window based approaches, our method generates various search trajectories for different action instances, depending on the video contents, the start search positions and the sequences of transformations 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 sequentially processes 15 temporal regions of an video to explore an unknown video segment. Thus, it is of great computational efficiency to deal with long untrimmed videos, compared with most of existing approaches.

The architecture of our proposed approach is illustrated in Fig. 1, which accepts video segments consisting of the current attended and augmented windows as input, and produces three outputs: the rewards for all possible actions (e.g. 7 actions), depending on which one optimal action will be selected at the current state; the action category prediction for a determined action proposal; the offsets of temporal boundaries for an action proposal. In general, our approach includes two main components: the Deep Q-Network generates action proposals through performing transformations on the current window according to a learned policy; the classification and regression networks devote to classify actions and refine locations of the action proposals. The entire model can be trained end-to-end, and generates all the action detection results for a given video sequence through processing it in one pass. To effectively represent video segments of various durations, we divide a video segment into several snippets, and devise a Long Short-Term Memory (LSTM) [11] structure upon extracted CNN feature representations of each snippet. Meanwhile, we concatenate the action histories with the representation features of both the attended and augmented windows as the state vector to be fed into the Deep Q-Network, avoiding the agent getting stuck into some repeated cycles.

Our model draws the inspiration from works [10], [12], [13] that have used reinforcement learning to build active models for object localization in image. However, we can not handle the video in a top-down way that is proved to be effective for image object localization, as the duration of a video is usually too long (from hundreds to thousands of frames). We initialize the search from the start of the video, which will terminate when an action instance has been found or the maximum transformation steps has been reached, and then a new search begins from the position away from the current attended region. We design a “jump” action to avoid the agent trapping itself at a region where no action occurs, which benefits to generate enough effective samples to accelerate the training process and to quickly locate action instances during the test phase. In summary, the key contributions of this paper are three folds: i.) we formulate the problem of temporal action detection as the process of active observation and refinement, where an agent performs a sequence of transformations on the current attended window to approach the true action region according to a learned policy that is realized via the reinforcement
learning; ii.) we construct an integrated framework based on deep neural network for action detection, which accepts an untrimmed video sequence as input, and consists of a Deep Q-Network for action proposals generation, and the classification and regression networks for the action category determination and action location refinement respectively. iii.) we conduct extensive experiments on two benchmarks for temporal action detection: THUMOS’14 and ActivityNet, and the results demonstrate that the proposed approach can achieve state-of-the-art detection performance in terms of precision via a small number of action proposals compared with other existing methods.

The rest of the paper is organized as follows: in section II, we review the related works to our approach, including those for action detection and object detection; in section III, we present our approach in great detail; in section IV, we report the experiment results on two datasets, including implementation details, ablation studies and comparisons with state-of-the-art methods; At last, we finish the paper with the conclusion.

II. RELATED WORK

We review relevant recent works in terms of video action recognition, temporal action detection, spatial-temporal action detection and object detection.

Action recognition This task has been tackled for a few years, and a large body of research works [1], [14]–[16], [18] have been presented. There exists a few works in literature, such as [23]–[26], describing in detail the existing research approaches and recent progresses they have achieved, to which interested readers can refer. In early years, researchers often handled the problem via constructing shallow classifiers like SVM based on low-level hand-crafted visual or motion features, such as HoG [27], SIFT [29], or MBH [17]. For instance, Wang et al. [14] encode dense trajectory features, which are obtained via tracking motion across the frames, with Bag-of-Word method and structure a SVM classifier for action categorization on real-life video data. Recently, impressed by the huge success of deep learning on image analysis related tasks, such as image classification [30], object detection [32], a number of approaches [1]–[3], [33] have introduced deep architecture models, especially CNN or Recurrent Neural Network (RNN), for better excavating the spatial-temporal information involved in the video clip. These approaches tackle the video via utilizing 2D convolution on a stack of frames and merging the results by early or late fusion [34], constructing 3D convolution model directly [2] on video volumes, applying RNN on CNN features extracted from every frame [33], or coping with image and optical-flow signals separately [1]. Karpathy et al. [34] firstly employ deep learning for action recognition in video, and design a variety of deep models processing single frame or sequence of frames. Wang et al. [3] propose Temporal Segment Network (TSN), which takes advantage of two-stream feature representation proposed in [1], and design a sparse sampling scheme to cope with video clips of long duration. Hou et al. [31] propose the FASNet that factorizes the action and scene, and encodes the most relevant and informative semantic cues for action recognition. Up to now, deep learning based methods have achieved state-of-the-art performances.

Temporal action detection Different from action recognition where the aim is to predict the action category for the whole video, temporal action localization needs to not only classify the action but also give out the temporal bounds. Most of existing methods [4], [5], [8], [35], [65] approach the problem via a two-step paradigm: first generating plenty of action proposals, and then running an action classifier on each of them. For instance, Shou et al. [4] utilize a multi-stage CNN detection network for action localization, where backgrounds are first filtered out by a binary action/background classifier based on C3D feature representation, then an action detection network incorporated both classification loss and temporal localization loss is trained for proposals refinement. By the limitation of the number of video frames allowed by the C3D model, they select 16 frames uniformly from the input video sequence. To better represent the motion patterns contained in video sequences of various durations, we utilize an LSTM structure built on extracted CNN features of evenly divided video snippets. They generate action proposals based on a sliding-window strategy, and operate a binary action classifier at uniformly distributed temporal positions with multiple scales. In order to reduce computational complexity introduced by such exhaustive search, Escorcia et al. [37] propose DAPs architecture that encodes the video sequence of fixed duration T with LSTM and retrieves action proposals of varied temporal lengths at one single process. However, their approach still needs to perform redundant computations on overlapping temporal windows while the attended window slides from the start to the end of a given video clip. Based on their work, Buch et al. [38] present a framework for action proposal generation in long video sequences, which only needs to process the entire video in a single pass without the use of overlapping temporal sliding windows, thus lifting the computational efficiency further. Likewise, our approach also processes the input video clip at one pass, where a search agent is first placed at the start of the clip, and moves towards the end via continuously adopting the optimal action according to a learned policy based upon current observation, thus resulting in an efficient search path to explore unknown video sequence. Gao et al. [35] divide the input video sequence into many short video units, and pool features extracted from a set of contiguous units as the representation of long video segment for action categorization, and meanwhile employ a coordinate regression network for temporal bound refinement. Our approach also incorporates a regression network, where the location offsets are calculated in the scale-invariant logarithmic space, which is based on the relative value rather than the absolute value used by [35], thus it will facilitate the model to converge more efficiently. For temporal action detection, it is often difficult for a visual detector to tell whether the action is just a portion of or
FIGURE 1: The framework of our temporal action detection model. It accepts video segments that consist of the current attended window and augmented temporal window as input, and outputs detected action results with regressed temporal boundaries and the inferred action category. The model constructs LSTM architecture upon extracted CNN features of randomly selected frames for video sequence encoding, and concatenates the action history vector with video sequence representations of both attended and augmented windows to constitute the state feature, which is then fed into Deep Q-Network to make the motion decision at each search step. Meanwhile, the feature representation of current attended window is sent to the classification and regression networks to generate final action detection results.

the entire action instance. To settle the difficulty, Xiong et al. [65] present a cascaded classification pipeline, where the features extracted from temporal pyramids of action proposal and surrounding segments are utilized to estimate the completeness of a candidate action instance. Unlike the works mentioned above, which process the video in chronological order, Yeung et al. [39] propose an attention based model that predicts the temporal action location through a few of glimpses over the whole video sequence, according to a learned policy obtained by reinforcement learning. Our approach is also based upon reinforcement learning, and the difference between their work and ours is that we locate the action through sequentially adjusting the span of current temporal window to narrow down the bias between true boundaries of matched action instances and predicted values, not inferring the temporal bounds directly as implemented by [39]. Except for the detection of completed actions, there exists few attempts [19]–[21] to tackle the task of early detection of the on-going action. Ma et al. [21] train an LSTM based model to learn activity progression for activities detection and early detection, where a novel ranking loss is designed to help the model better capture the progression of activities, and the classification loss is calculated at each time instant.

Spatio-temporal action detection A related problem to temporal action localization is that of detecting actions not only temporally but also spatially. Most of existing approaches [40]–[43] for spatio-temporal action localization are based upon the sliding-window scheme or the extension of object proposals from images to videos. They can be generally divided into three categories: supervised based [40]–[42], [50], semi-supervised [43], [44] and unsupervised based [22]. Zhou et al. [22] treat the spatial and temporal extents of the action as latent variables of a SVM framework, which is built upon background and separated foreground objects obtained via splitting and merging trajectories. Sul tani et al. [44] tackle with the problem of action detection in videos using web images without any prior annotations only action name, depending on semi-supervised method. Some other researchers address the task by first solving the spatial localization problem and then trimming the results on the temporal domain with certain post processing. Suman et al. [40] accomplish spatio-temporal action detection on account of frame-level detection results via dynamic programming across the whole video to solve spatio-temporal action association problem. On the whole, these approaches provide more detailed localization information, however, they tend to suffer from high computational cost. It is likely that temporal action proposals can be leveraged to reduce the temporal search space of such approaches for spatio-temporal localization with efficiency.
Object detection As presented in the task of temporal action detection, most of recent approaches for object detection are build upon the paradigm of “proposal + classification” [28], [32], [45] instead of sliding-window search strategy adopted by early methods. Object proposals are usually generated either by means relied on hand-crafted low-level visual cues, such as SelectiveSearch [46] and Edge-box [47], or produced by Region Proposal Network implemented on CNN feature maps with anchor boxes arranged uniformly on the image, such as Faster R-CNN [45]. However, generating too many proposals for a image with only a few objects is unnecessary and computational inefficient. Some researchers attempt to reduce the number of proposals with an active object detection strategy [10], [48]. Caideco et al. [10] learn an optimal policy to locate one single object in the image via Deep Q-Learning, where the search 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. [12] 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. Visual attention based models [49] are also related to our work. These approaches often leverage the information gathering from a sequence of history observations to facilitate the decision making at current state. For instance, Minh et al. [9] propose a visual attention model embedded in RNN to focus on different regions in a sequential decision-making process for digit character recognition. Inspired by [10], [12], we present an active temporal action detection model based upon Deep Q-Learning, where the agent takes a series of transformations on current attended temporal window to cover true action regions more accurately, relying on the sequence of observations and adopted actions in the history. The distinction between their work and ours lies in two folds: 1) they process the image in a top-down way, while we deal with the video sequence according to the time order; 2) we devise a multi-task learning network that incorporates the classification and regression networks to predict the action label and refine the action boundaries simultaneously.

III. SELF-ADAPTIVE ACTION DETECTION MODEL

In this section, we present our temporal action detection model, which is self-adapted and will gradually adjust the position of current temporal window based on the state representation encoded by the content of attended window and the history of executed actions, attending to cover true action regions as accurately as possible in a few steps. We cast the problem of search of temporal action proposals 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 consists of the input video sequence, and the agent has an observation of the current video segment at each time step, which is called temporal window and encoded with deep feature representation. The agent consecutively restructures the position or span of the attended window as to achieve the goal of locating the action precisely. The agent receives positive or negative rewards after each decision making during the train phase to learn an effective action policy. During the test phase, the agent performs according to the learned policy, which often provides an optimal action decision from the entire search process. Moreover, we construct the classification and regression networks based on search results of MDP, which gives inferences of action categories upon generated action proposals and promotes the accuracy of temporal action localization simultaneously. The framework of our detection model is illustrated in Fig. 1. The MDP comprises of three components, namely a set of actions A, a set of states S and the reward function R(s,a), which will be detailedly illustrated in the following paragraphs. The constructions of classification and regression networks will also be discussed. To avoid confusion, the action performed by the model during search is called motion in the remaining paragraphs.

A. MDP FORMULATION FOR ACTION PROPOSAL GENERATION

Actions. The set of actions A are selected to perform by the agent, which can be separated into two categories: one group of transformations on current attended window, e.g. “move left”, “move right”, “expand right”, and the remaining one for terminating the present search, “trigger”, as shown in Fig. 2. The transformation group can be further divided into a few regular motions realizing functionalities of shift and scale, and an irregular motion to propel the agent to be away from current position. The regular motions vary the current window in terms of position and temporal span around the attended segment, e.g. “move left”, “expand right” or “shrink”, which are adopted by the agent to increase the intersection with the groundtruth that has overlap with the current window. The irregular motion, namely “jump”, moves the window to a new position away from the current site to avoid that the agent traps itself around locations where either the action occurring nearby has been found or there exists no action. The change caused by any regular motion at each search step to the temporal window can be measured by a specific value, which is in proportion to the current window size. For instance, suppose that the current window is denoted as \([x_l, x_r]\), where \(x_l\) and \(x_r\) stand for the left and right boundaries respectively. The motion “move left” shifts the window to a new site of \([x_l', x_r']\) with \(x_l' = x_l - x_r' = x_r - x_r' = \alpha \times (x_r - x_l)\), while the motion “expand left” scales the window with the change of \(x_l - x_l' = \alpha \times (x_r - x_l)\) and \(x_r' = x_r\). Here, \(\alpha \in [0, 1]\) is a parameter that gives a trade-off between the search speed and localization accuracy. The motion “jump” will generate a new window with the average duration of actions within a dataset, and is placed at a position away from the current attended window. The regular motions drive the agent to gradually adjust its position to cover the true action region more accurately when the present window overlaps with the groundtruth; while the
irregular motion “jump” propels the agent to deviate from the discovered region, and to explore unknown video segment that may contain actions in a discontinuous and efficient way. The motion “trigger” is performed by the agent whenever it reckons that an action instance has been localized by the current window, terminating the process of the present search and restarting a new search for the next action at an initialized position being distance away from the current site.

**State.** The state of MDP is expressed as the concatenation of three components: \( o_c \) and \( o_a \) for the feature presentations of the current attended window and augmented window respectively, and \( h \) for the history vector of taken motions in the recent past. The search space of state set \( S \) is enormous, taking into consideration the large diversity of action postures among different actors, the variation of background scenes and camera viewpoints. Therefore, it is important to devise a feature representation that can depict the current observation with efficiency and generality. We construct an LSTM module upon extracted CNN features of single frames to generate a feature representation for a video segment. Specifically, given a video segment (the current attended window or augmented window), we firstly divide it into several snippets (e.g. 16) evenly. Then, we randomly select a single frame within each snippet, and utilize a CNN (e.g. VGG-16 [61]) to extract its feature representation, which is also deemed as the abstract description of a video snippet. At last, we sequentially feed the feature representation of each video snippet into an LSTM, and take the output hidden state of the last time instant as the feature presentation for the whole video segment. Compared with C3D model, which is designed to process only consecutive 16 frames, our LSTM structure upon CNN module can cope with video segments of arbitrary durations while utilizing a sparse sampling scheme. The augmented window is the extension of the current attended window, which extends the temporal duration with \( 1/3 \) at both of two boundaries. Compared with the attended window, the augmented window is larger and includes both the attended action and the context information that is necessary to guide the agent for making a proper motion, therefore we concatenate the feature representations of both attended and augmented windows for the integrated description of the current video segment.

The history of taken motions \( h \) is a binary vector that records which motions have been adopted by the agent during the recent past. Each motion in the history is represented by a 7-dimension binary vector where all the bit values are zero except the one corresponding to the taken motion. In our application, we totally reserve 5 past motions as the history, the history of taken motions informs the agent the search path that indicates the video segments recently attended. Although the dimension of \( h \) is low compared to the concatenation of representation feature vector \( o_c \) and \( o_a \), it is beneficial to stabilize search trajectories that might get stuck in repetitive cycles and improve the average localization precision, as pointed out by [10].

**Reward Fuction.** The reward function \( R(s, a) \) provides a feedback to the agent whenever it performs the motion \( a \) at the state \( s \), which awards the agent for executing motions that results in the improvement of action localization accuracy while gives the punishment for those motions that lead to the decline of the accuracy. The quality of action localization is evaluated via a simple yet indicative measurement, Intersection over Union (IoU), between current attended temporal window and the groundtruth of any action instance. Supposing that \( w \) stands for the current temporal window and \( g \) represents the groundtruth of an action, then the IoU between \( w \) and \( g \) is defined as \( \text{IoU}(w,g) = \frac{\text{area}(w \cap g)}{\text{area}(w \cup g)} \). The reward function value is estimated via the differential of IoUs between two successive states \( s \) and \( s' \), which is generated due to the state transfer of the agent from \( s \) to \( s' \) by performing the motion \( a \). Specifically, it is formulated as following:

\[
\begin{align*}
    r(s, a) &= \max_{1 \leq i \leq n} \left( \text{sign}(\text{IoU}(w'_i, g_i) - \text{IoU}(w, g_i)) \right), \\
    \text{where } w' \text{ and } w \text{ are attended windows corresponding to states } s' \text{ and } s \text{ respectively, } n \text{ is the number of action instances within the input video sequence. The reward function returns } +1 \text{ or } -1. \text{ Eq. 1 indicates that the agent receives the reward } +1 \text{ if the new window } w' \text{ has larger IoU with any of the groundtruth than the previous window } w, \text{ while the reward } -1 \text{ otherwise. Compared with the original differential in IoUs, which is often too small to confuse the agent about which motion benefits the action localization, such quantized binary reward makes the agent clearly realize that at current state, which motion drives the attended temporal window to intersect with the groundtruth more larger, and thus accelerates the convergence of the model during the training phase. In order to avoid the vibration introduced by the tiny distinction, we set a differential threshold } \xi_d \text{ (e.g. } \xi_d = 0.05) \text{ in the implementation. While } \text{IoU}(w'_i, g_i) - \text{IoU}(w, g_i) \geq \xi_d \text{, the reward function returns } +1, \text{ otherwise returns } -1. \text{ In addition, such reward function scheme facilitates better localization towards the temporal range of action especially for the video with multiple action instances, as there is no limitation on which action should be focused on at each state. The “trigger” motion has a different reward function scheme, as it leads to the termination of the current search and there is no next state. The reward of “trigger” is determined by a piecewise function about the IoU threshold, which can be presented as following:}
\end{align*}
\]

\[
    r_t(s) = \begin{cases} 
    +\eta & \text{if } \text{IoU}(w, g) \geq \tau \\
    -\eta & \text{otherwise .} 
    \end{cases}
\]

In Eq. 2, \( t \) represents the “trigger” motion, \( \eta \) is the reward value and chosen as 3 in our experiment, \( \tau \) is the IoU threshold, which controls the tradeoff between the localization accuracy and computational overhead. The large \( \tau \) will encourage the agent to locate the action more precisely, however, it consumes more search steps to complete the
localization. In actual implementation, during the training phase, we do not abort the current search process when the agent correctly performs the motion “trigger” for the first time, but encourage it continuously to go ahead for a few search steps until executing “trigger” motion again. Therefore, the agent recognizes many termination states that have IoUs with groundtruth more than \(\tau\). We utilize \(\tau = 0.65\) in our application, and find that larger \(\tau\), such as 0.7 or 0.8, gives rise to negligible promotion on recall value, which is validated by the experiments.

**Deep Q-learning**. The goal of the agent is to locate the action as accurate as possible, which is equivalent to maximize the sum of discounted rewards that are received through successively transforming the attended temporal window during a sequence of interactions with the environment (an episode). Specifically speaking, the agent needs to learn a policy \(\pi(s)\), which determines an optimal motion \(a\) at state \(s\) in the view of maximizing the long-term benefit that is to accurately locate the true action within limited search steps. Due to the lack of state transition probabilities and dealing with the model free environment, we utilize reinforcement learning method, specially Deep Q-learning model, to estimate the Q value for each state-action pair. In this paper, we follow the Deep Q-learning framework proposed by Mnih et al. [9] that estimates the action-value function via a deep neural network. The architecture of our Deep Q-Network (DQN) is illustrated in Fig. 1, which accepts the state vector encoded as the concatenation of video segment representation and history action vector as input and evaluates the Q values for all possible motions during the forward propagation in a single pass. During the training phase, the agent executes multiple episodes with randomly initialized temporal positions for each input video clip. Following [9], we also incorporate the replay-memory scheme to collect various transition experiences from the past episodes, from which each transition record may be repeatedly used for model updating, in favor of breaking short-term correlations between states and providing more data for training. The size of replay memory buffer is set as 2000, from which a minibatch (e.g., 200 transitions) is randomly selected as training samples at each time.

To progressively shift motions of the agent from exploration to exploitation, we train DQN with the \(\epsilon\)-greedy policy [53], which means that at each state, the agent randomly selects a motion from the action set \(A\) with probability \(\epsilon\) or greedily chooses the optimal motion generated by DQN with probability \(1-\epsilon\). We train DQN for 20 epochs, where during the first 10 epochs, \(\epsilon\) is annealed linearly from 1.0 to 0.1 to gradually encourage the agent to perform motions according to the learned model, while after 10th epoch, \(\epsilon\) is fixed at 0.1, henceforth the agent almost updates the model based on motion experiences. Exploration motivates the agent to collect a variety of transitions upon which the motion policy is built; while exploitation lets the agent adjust the learned policy on account of successes and failures resulted from selected motions.

**B. CLASSIFICATION AND REGRESSION NETWORK**

Given the action proposal generated by the Deep Q-Network, we utilize the classification and regression networks to predict the class category and to refine the action localization. The network accepts the feature representation of the current attended window as input, and has two fully-connected layers interleaved with dropout operation, and finally branches into two sibling output layers: one for class label estimation (\(m\) classes plus one background category) and the other for position offsets calculation (two outputs for the center location and temporal duration respectively). We utilize softmax loss for classification and smooth L1 loss for regression, the objective function of which is presented as following:

\[
L(u, t^u, t^v) = L_{cls}(u) + [u \geq 1]L_{reg}(t^u, t^v).
\]

\[
L_{cls}(u) = -\log p_u \quad \text{is the log loss for classification, where} \quad p_u \text{is the prediction score that the action belongs to class } u.
\]

The indicator function [\(u \geq 1\)] equals to 1 when \(u \geq 1\) and 0 otherwise. The regression loss \(L_{reg}(t^u, t^v)\) is defined over the target \(t^u\) and the predicted value \(t^v\), both of which can be presented as two-tuple values \(\delta c, \delta l\). For instance, given an action proposal \(p_i\), \(c_i\) and \(l_i\) denote the center location and the duration of the proposal while \(c^e_i\) and \(l^e_i\) stand for the same of the corresponding groundtruth action. The target \(\{\delta c^u_i, \delta l^u_i\}\) are computed as follows:

\[
\begin{align*}
\delta c^u_i &= (c^e_i - c_i)/l_i \\
\delta l^u_i &= \log(l^e_i/l_i).
\end{align*}
\]

Likewise, \(\{\delta c^v_i, \delta l^v_i\}\) represents the predicted relative offset outputted by the regression network. We consider an action proposal as the true positive detection if and only if
if its action class is correctly predicted and its IoU with the matched groundtruth is larger than a threshold $\xi_c = 0.6$. We only perform the regression on the true positive detection, and calculate the relative offset for each action category respectively.

### IV. EXPERIMENTAL RESULTS

We evaluate the performance of our temporal action detection model on the datasets of THUMOS’14 [58] and ActivityNet [54]. Firstly, we evaluate the recall performance of our DQN model, and then utilize these proposals for temporal action detection task. We follow the standard evaluation protocol [54]. Firstly, we evaluate the recall performance of our DQN model on the datasets of THUMOS’14 [58] and ActivityNet v1.1, v1.2 and v1.3. The version v1.2 is UCF-101 has no label on temporal action boundary, we train our model on validation set and report the results on test set.

**THUMOS’14**: It has annotated untrimmed videos from 20 sports action categories, and the whole duration lasts over 20 hours. The dataset comprises 413 videos with 200 for validation and 213 for test. As the designated training dataset is UCF-101 has no label on temporal action boundary, we train our model on validation set and report the results on test set.

**ActivityNet**: There are three released version of ActivityNet dataset: v1.1, v1.2 and v1.3. The version v1.2 comprises 76 hours of action annotated videos on about 100 action classes, on which we also evaluate the generalization ability of our proposal model.

### A. DATASET

We test our action detection model on two datasets:

**THUMOS’14**: It has annotated untrimmed videos from 20 sports action categories, and the whole duration lasts over 20 hours. The dataset comprises 413 videos with 200 for validation and 213 for test. As the designated training dataset is UCF-101 has no label on temporal action boundary, we train our model on validation set and report the results on test set.

**ActivityNet**: There are three released version of ActivityNet dataset: v1.1, v1.2 and v1.3. The version v1.2 comprises 76 hours of action annotated videos on about 100 action classes, on which we also evaluate the generalization ability of our proposal model.

### B. IMPLEMENTATION DETAILS

Our model is implemented on Torch 7 [55]. During the pre-processing stage, we downsample the videos to have the same frame rate (30 frames/second). The VGG-16 model is pre-trained on imagenet dataset and finetuned on UCF-101 for action detection. We fetch the feature vector from fc6 layer of VGG-16 model operated on a single frame, and fed it into the LSTM architecture. The vector dimension is condensed from 4096 to 1024 via PCA for overfitting prevention. The LSTM has one layer and 512 hidden units. There are 16 timesteps input for LSTM, which means that each video segment will be evenly divided into 16 snippets. The dimension of fully connected layers in both the DQN (3 layers) and CRN (2 layers) are all set as 1024. The whole training process can be divided into 3 stages: firstly, we train DQN as well as the LSTM module; then keep DQN steady, we train both LSTM and the CRN with the action proposals generated by DQN; at last, with the LSTM unchanged, we simultaneously adjust the DQN and CRN. The learning rate for DQN is $1e^{-3}$ with a decay rate of $1e^{-4}$, and the learning rate for CRN is $1e^{-4}$ with a decay rate of $1e^{-5}$. Dropout is applied with a ratio of 0.2 on both the DQN and CRN.

During the training phase, to avoid the agent getting itself trapped around temporal regions without any action instance, the agent is encouraged to randomly choose to take the motion “jump” towards left or right, whenever the IoU for current attended window is zero. During the test phase, the agent starts its search from the beginning of the input video and takes motions to adjust its position. The initial length of the search window is set as the mean duration of action instances within a given video dataset. The maximum motion steps taken by the agent is set as 15. The agent will restart a new search from the right side of current attended window, whenever it has taken a “trigger” motion or used up motion steps. Different from the training phase, the agent consistently takes a leap towards right, the distance of which is four times farther than “move left/right”, when it adopts a “jump” motion. Therefore, the agent traverses the video from the start to end, which guarantees that different action instances existed within given video sequence will be visited.

### C. TEMPORAL ACTION PROPOSAL EVALUATION

Given the test video sequence, temporal action proposals generated by DQN can be considered in two modes: 1) all the regions attended by the agent during a search episode; 2) terminal regions, which is explicitly indicated by the motion “trigger” to denote the presence of an action. Generally speaking, the DQN runs for about 450 steps with around 80 “trigger” averaged for each video. Without an action/background classifier, we score each attended region with the Q-value of motion “trigger” predicted by the model, and add a large constant only to terminal regions in order to give them higher priority when ranking the proposal candidates. To evaluate the quality of our temporal action proposals, we use the metrics as following [37]:

**Average Recall**: We compute average recall for IoU between 0.5 and 1, as a function of the number of proposals. The best proposal approach is expected to achieve a higher recall with less proposals.

**Recall vs. IoU**: We compute recall at a fixed number of proposals, as a function of IoU. This metric is used to evaluate the localization quality of the top ranked proposals that are of most important for further action detection task. We score the proposals with the output of DQN on “trigger” motion, and fix the number of proposals as 100.

1) **THUMOS’14**

We train our DQN on the validation set of THUMOS’14, and compare action proposal performance with recent temporal proposal methods on the test set. The recall performances for various comparison approaches are illustrated in Fig. 3. The compared methods include: DAPs [37], S-CNN [4], Sparseprop [56], BoFrag [57] and SST [38], the performances of which are cited from [37]. As shown in Fig. 3(a), compared
with the state-of-the-art methods, the performance of our approach is comparative on the whole, and is superior than others especially when the number of generated proposals is low (< 100). The reasons for such phenomenon can be explained from two aspects: 1) As mentioned before, the number of effective proposals for our approach, which are indicated by executing "trigger" motion, is 80 on average per video clip. Except for these valid proposals, others are intermediate exploring results generated by the agent when interacting with the video clip, which are unreliable on most occasions; 2) Restricted by the mechanism of action policy of the DQN, in most cases the agent tends to stop exploration when it considers that the IoU between current attended region and the matched action instance is more than 0.5, which results in that the number of proposals that have larger IoU is low. From Fig. 3(a), it can be observed that the total number of proposals for our approach is fewer than compared methods, the reason for which is that the agent often interacts with the given video clip at no more than 500 steps during a completed search. We also calculate the recalls at various IoUs with 100 top ranked proposals for various compared methods, which are shown in Fig. 3(b). Sliding window is the baseline that generates proposals including all temporal windows of lengths varying from 16 to 512 with 50% overlapping, and each window is scored with a random value. Fig. 3(b) shows that our approach gains higher recall values than others while the IoU is varying from 0 to 1, which confirms that the effective proposals indicated by “trigger” motion are credible and of high quality, and also explains the results in Fig. 3(a) that our approach behaves superiorly than other methods when the number of proposals is low. This is of significance for the overall performance of temporal action detection in practice, since it is unlikely to verify thousands of proposals on a video of few minutes for the sake of efficiency.

2) ActivityNet

An important characteristic of action proposal approaches is their generalization ability, which means that to generate proposals for action categories unseen during the training phase. We assess the generalization ability of our DQN on ActivityNet, which is a diverse action set and includes many categories unrelated with sports, such as Brushing teeth, Shoveling snow. Furthermore, more than 50% of activity annotations in ActivityNet span more than 512 frames, which is much longer than those short action instances in THUMOS’14. Taking both the variety of action categories and the diversity of action durations into consideration, we divide the validation set of ActivityNet into three groups: ActivityNet_untrain (all the action categories), ActivityNet \( \cap \) THUMOS’14 (9 sport categories shared by both datasets), ActivityNet \( \leq 512 \) (unseen action categories during the training phase and the action durations last up to 512 frames). The recall performances of the DQN on these three sub-datasets are illustrated in Fig. 4. We generate 450 proposals on average for each video sequence. The results indicate that on ActivityNet \( \cap \) THUMOS’14, where the same action categories are shared by both the test and training phases, the performance of the DQN is close to that on THUMOS’14 when the proposal number is low. With the proposal number becoming large, the performance gap between them is gradually widening, the reason of which is mainly due to the difference in data distribution of the same action category on two datasets. As for ActivityNet_untrain and ActivityNet \( \leq 512 \), the performances are inferior, the reasons for which are derived from two aspects: 1) The action duration distribution of ActivityNet is significantly different from that of THUMOS’14. A large number of action instances in ActivityNet are much longer than the mean length of action instances in THUMOS’14. The state representation model of the video segment, which are learned by the DQN on THUMOS’14 is not suitable for ActivityNet; 2) The video segments from unseen action categories in ActivityNet result...
in some unknown states in the DQN, which confuses the agent when making an optimal decision at each state. In other words, the state-action policies are not learned during the training phase, and therefore the agent lacks reliable policies for such occasions. In addition, we also train the DQN on the ActivityNet, the recall performance of which is shown in Fig. 4 as ActivityNet_train. It can be observed that the performances on the two datasets are similar, which demonstrates the effectiveness of the DQN.

D. ABLATION STUDY

During the experiments discussed above, the quality of action proposals is evaluated via the reward value of “trigger” motion gave out by the DQN at each forward propagation. However, the reward value of “trigger” motion only represents the likelihood of selecting “trigger” motion among the action set at the given state. Can it really reflect the relative merits of different proposals? In other words, is the proposal with larger “trigger” value more likely to be the true action instance than the one with smaller value? To answer the question, we devise the ablation study with a modified network model. The alterations of the DQN include the following two parts: 1) we delete the “trigger” motion from the action set of the DQN; 2) we add a binary classification network for action/non-action discrimination, which accepts the concatenation of the representations of current attended and augmented windows as input. Therefore, the “trigger” motion is replaced by the classification network, which is trained independently with the groundtruths, and can evaluate the quality for each action proposal under an uniform standard. The classification network has two fully-connected layers (1024-d) interleaved with dropout operation and outputs the probability of being action/non-action. For the sake of simplicity, we denote the model with the classification network as BC. During the training phase, we utilize the proposals that have the IoUs with the groundtruths more than 0.7 as positive samples, others as negatives. We calculate the Average Recall of the BC on THUMOS’14, and compare the result with the DQN, which is shown in Fig. 5.

We can see that the recall values of the BC and the DQN are almost the same when the number of proposal is less than 100; The BC model has a better recall performance than the DQN when the proposal number gets larger. This indicates that: 1) for the proposals pointed out by the “trigger” motion, which correspond to the first few dozens of proposals and are often crucial for effective action detection, the reward value indeed reflects the quality; 2) utilizing a classification network instead of the “trigger” motion can bring a small amount of improvement on recall.

E. TEMPORAL ACTION DETECTION ANALYSIS

In this section, we evaluate the performance of our proposed model on the task of temporal action detection, the goal of which is to not only locate the temporal regions where actions take place, but also designate the regions with right action categories. We implement the experiments on both ActivityNet and THUMOS’14 datasets, and utilize mean Average Precision (mAP) as the criterion. For THUMOS’14 dataset, we pick up the top-80 proposals (indicating by the “trigger” motion) and the top-200 proposals for assessing detection performance respectively, while for ActivityNet, we use the top-200 proposals for performance evaluation. Tab. 1 illustrates the comparison results of various action detection methods on THUMOS’14 dataset, which are evaluated via mAP under the IoU ranging from 0.1 to 0.7.

From Tab. 1, It can be observed that our approach achieves a competitive detection result compared with other state-of-the-art methods. For instance, at IoU of 0.5, the mAP of our approach with only top-80 proposals is 28.7, which is comparable to 28.9 of RC3D [62] and 29.2 of SS-TAD [63]. Whereas, both of them utilize at least 300 proposals for action detection. Such performance of our approach has great significance, as we can detect action instances with fewer proposals, which will bring about great computation reduction especially for tackling with long video sequences lasting for several seconds. Although the recall performance of our approach is modest on the whole, it has produced the competitive detection result. The reason we think, as pointed...
out by [37], is that the DQN has generated proposals with a small number of hard negatives, which allows the action detector to keep the number of false positives low. Actually, all the proposal methods tend to saturate after using sufficient proposals (e.g. 500), therefore, the mAP starts to decrease along with the decline of the ratio of true positives. Furthermore, to figure out whether the performance promotion of our approach with fewer proposals is produced by employing a different action classifier, we conduct contrast experiments of using proposals at a fixed number under an unified classifier for a few proposal methods. With the proposal number of 50, DAPs [37] and S-CNN [4] acquire mAPs at 6.99 and 16.9 under TSN [3] classifier; while with proposal number of 100, the mAPs for DAPs and S-CNN will present at 9.16 and 12.4, respectively. These scores are far below our performance of mAP at 27.1 with 80 proposals under TSN. The compared results demonstrate that our approach can surely generate fewer proposals of high quality, which are mostly composed of true positives and thus efficient for the task of temporal action detection.

When the top-200 proposals are adopted, the mAP of our approach reaches to 30.8, which is just inferior to CBR-TS [64] that utilizes a cascade regression structure to refine obtained action proposals, and SSN (Kinetics) [65] that is built on a strong feature extraction module refined on the enormous action-classification dataset Kinetics. The average precisions for each action category on THUMOS’14 of various approaches are shown in Fig. 6. Our approach behaves comparatively or superiorly than other methods for most of the categories. For instance, on classes Shotput, Pole Vault, Hammer Throw to name a few, our approach has achieved the best detection result. However, for a few classes, such as Basketball Dunk where multiple persons appear in one single scene, or Cliff Diving where the boundary between the starting of the action and the background is indistinct, our approach cannot generate a precise detection result.

The scale size \( \alpha \) of transformations on the current attended window is a vital hyperparameter, which balances the search speed and the action localization precision. With a smaller \( \alpha \), the DQN will consume more search steps to locate an action and the localization result is likely to be more accurate. We conduct experiments with various \( \alpha \) ranging from 0.1 to 0.4 on THUMOS’14 (IoU = 0.5), which is illustrated in Tab. 2. We can see that the mAP achieves the highest score when \( \alpha = 0.2 \). The reasons why the score of \( \alpha = 0.1 \) is lower than that of 0.2, we think, are as following: 1.) we use the same maximum search steps 15, while a larger value may be beneficial for a smaller \( \alpha \); 2.) the regression operation alters the localization results.

Since only the temporal action proposals on the training and validation sets of ActivityNet V1.2 are publicly available, we evaluate the performance of the proposed approach on the validation set, which is shown in Tab. 3. We can see that compared with state-of-the-art methods, our approach acquires the superior action detection result (0.1 promotion on average mAP). Considering the fact that ActivityNet is much larger than THUMOS’14, the experiment results further demonstrate the effectiveness of our approach, and negate the guess that the performance improvement is brought about by overfitting. The Fig. 7 illustrates how the DQN generates multiple action proposals for a given video sequence. Fig. 7 (a) presents the evolution of IoU as the agent sequentially

<table>
<thead>
<tr>
<th>( \alpha )</th>
<th>0.1</th>
<th>0.2</th>
<th>0.3</th>
<th>0.4</th>
</tr>
</thead>
<tbody>
<tr>
<td>mAP</td>
<td>26.7</td>
<td>28.7</td>
<td>27.8</td>
<td>26.9</td>
</tr>
</tbody>
</table>

TABLE 2: The mAPs of our approach under various \( \alpha \) values. The evaluations are performed with IoU = 0.5 on THUMOS’14.

<table>
<thead>
<tr>
<th>IoU</th>
<th>0.5</th>
<th>0.75</th>
<th>0.95</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td>Xiong et al. [8]</td>
<td>41.1</td>
<td>24.1</td>
<td>5.0</td>
<td>24.9</td>
</tr>
<tr>
<td>SSN (ImageNet) [65]</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>24.5</td>
</tr>
<tr>
<td>Ours</td>
<td>42.3</td>
<td>24.9</td>
<td>5.2</td>
<td>25.0</td>
</tr>
</tbody>
</table>

TABLE 3: Action detection results on ActivityNet v1.2 validation set. The performances are measured by mAPs calculated over multiple IoU thresholds (0.5, 0.75, 0.95), and the average mAPs are computed over the IoU range 0.5:0.05:0.95.
FIGURE 6: The average precision for each action category on THUMOS’14. The results are calculated with the official toolkit. The compared methods include: Yeung et al. [39], S-CNN [4], R-C3D [62], and SSN [65].

FIGURE 7: The instance of how the DQN takes motions to generate multiple action proposals. The examples are sampled from the action LongJump of THUMOS’14. (a) indicates the evolution of IoU when the agent transforms attended temporal windows. (b) shows the sequence of motions adopted by the agent to locate action instances. The last row is the time shaft, where the red lines denote the groundtruths. The top 3 rows are the exploring processes corresponding to action instances #1, #2 and #3. The green circle indicates that it is a right “trigger” decision.
transforms the attended temporal window when processing the input video sequence exhibited in Fig. 7 (b). The red or green circles indicate motion “trigger”. We can observe that there always exists a decline of IoU after the “trigger” motion is executed, which means that the agent is inclined to determine an attended temporal window as a completed action instance when its intersection with the matched action instance achieves a local maximum. Fig. 7 (b) displays that how the agent adopts a sequence of motions to locate various action instances, and the history of IoUs during the exploring procedure are also annotated nearby.

F. RUN-TIME PERFORMANCE

The run-time property of our approach is dependent on the DQN’s search efficiency for a input video sequence. For a well trained DQN agent, it will concentrate on the matched action instance in a couple of steps once it perceives that the current attended window contains action snippets. Meanwhile, it also can accelerate the exploring process over the video sequence with “jump” motion. Besides, the selection of scalar $\alpha$ is also an important factor that will influence the run-time efficiency. A large $\alpha$ will make the agent take a brief glance over the video sequence in most cases, but will result in coarse proposals. As a trade off, we set the $\alpha = 0.2$ during both the training and testing phase. On TITAN V platform, the average run-time of our approach over all testing videos in THUMOS’14 is 70 frames per second, including the online image feature extraction.

V. CONCLUSION

In this paper, we have introduced an active action detection model that learns to adaptively adjust the span of current attended temporal window to cover true regions of action instances in a few steps. We build our model based on deep reinforcement learning and lean an optimal policy to direct the agent to act. Specifically, given the input video sequence, an agent is first placed at the beginning of the video and transforms the size or position of attended temporal window based on current observation and the learned policy to search action instances existed in the whole video sequence. To effectively represent video segments of different lengths, we design an LSTM module based on extracted CNN features for temporal-dependency motion representation. Then, we train a Deep Q-Network to learn the reward function for a set of possible actions, based on which the action with the maximum reward is selected for execution at current state. In addition, we devise the classification and regression network to predict the action label and calculate the offsets between the predicted boundary results and groundtruths. The whole model can be trained end-to-end, and detects all the action instances within the given video sequence at one pass. We evaluate the action proposal performance of our approach on THUMOS’14 and test its generalization ability for unseen action classes on ActivityNet. Furthermore, we compare the action detection performance of our approach with other state-of-the-art methods on both of the two datasets. Experimental results validate the effectiveness of our approach, which can achieve comparative or superior performance than other detection methods while producing much fewer action proposals.

REFERENCES


2169-3536 (c) 2018 IEEE. Translations and content mining are permitted for academic research only. Personal use is also permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.


This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2018.2872759, IEEE Access