CAG-QIL: Context-Aware Actionness Grouping via Q Imitation Learning for Online Temporal Action Localization

Hyolim Kang, Kyungmin Kim, Yumin Ko, Seon Joo Kim

Yonsei University

{hyolimkang,kyungminkim,yuminko,seonjookim}@yonsei.ac.kr

Abstract

Temporal action localization has been one of the most popular tasks in video understanding, due to the importance of detecting action instances in videos. However, not much progress has been made on extending it to work in an online fashion, although many video related tasks can benefit by going online with the growing video streaming services. To this end, we introduce a new task called Online Temporal Action Localization (On-TAL), in which the goal is to immediately detect action instances from an untrimmed streaming video. The online setting makes the new task very challenging as the actionness decision for every frame has to be made without access to future frames and also because post-processing methods cannot be used to modify past action proposals. We propose a novel framework, Context-Aware Actionness Grouping (CAG) as a solution for On-TAL and train it with the imitation learning algorithm, which allows us to avoid sophisticated reward engineering. Evaluation of our work on THUMOS14 and Activitynet1.3 shows significant improvement over non-naive baselines, demonstrating the effectiveness of our approach. As a by-product, our method can also be used for the Online Detection of Action Start (ODAS), in which our method also outperforms previous state-of-the-art models.

1. Introduction

Fueled by flourishing video platforms, video understanding tasks are drawing substantial attention in the computer vision research community. Among many video understanding tasks, Temporal Action Localization (TAL), the task of extracting action instances from an untrimmed video, has been one of the most popular topics. A plethora of works have been done in TAL [32, 30, 5, 8, 44, 22, 43, 39, 41, 19, 1], implying the importance of action instances in video understanding

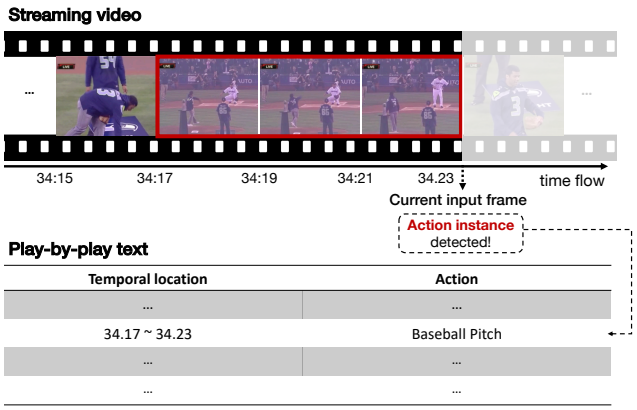

Figure 1. Example of play-by-play system applied on the video from THUMOS14.

However, detecting action instances from streaming videos has received no attention, even though more video streaming services are being provided which require real-time and online approaches. In contrast, many online algorithms are being introduced in other video understanding tasks such as object detection and tracking [2, 36, 28], video object segmentation [25], and video instance segmentation [40]. We argue that temporal action localization can also benefit by going online, as it can provide valuable information for many practical real-world applications that involve online video understanding.

Popular sports websites provide a feature called live play-by-play system that shows the progress of a sports game in real-time. To be able to develop an AI-based play-by-play system, the algorithm needs to detect both the start, the end time and the class information of the occurring action in an online manner. Previous temporal action localization methods cannot be used as they operate in an offline fashion, requiring the whole video sequence to be seen. Figure 1 shows an application of the online temporal

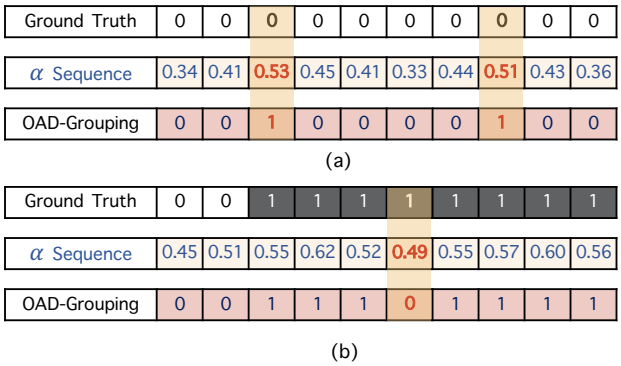

Figure 2. Limitations of simple extension of OAD to On-TAL: (a) action tick and (b) fragmentation. α sequence represents a series of actionness scores from a binary OAD model. In the case of action tick, the model emits very short action instances because α slightly exceeds the threshold (0.5) although there is no action. Action fragmentation refers to the opposite situation. To avoid these problems, the model should be aware of its decision context and be *conservative* in changing its decision.

action localization in the play-by-play system for the live sports broadcasting. Another important use of the online version of the temporal action localization can be found in the robotics domain. For a robot to interact with humans in real-time, it needs the information about the whole action instance before deciding how to react.

To this end, we suggest a new challenging task, Online Temporal Action Localization, or On-TAL, which aims to produce action instances from an untrimmed streaming video on the fly. As the name suggests, the final output of On-TAL is the same as offline TAL – action instances with the start and the end timestamps. But in the On-TAL setting, an action instance needs to be produced as soon as the action ends, which poses several challenges differentiating it from offline TAL as follows:

- Without accessing future frames, a model has to decide whether the current frame contains an action or not because it needs to return an action instance immediately when the action ends. Note that this decision making occurs for every frame.
- As the action instance is produced promptly and one cannot go back in time, modification of past proposals is strictly prohibited, making it impossible to use prevailing post-processing methods such as (Soft-) Non-Maximum Suppression (NMS) [3].

With these constraints, we cannot simply extend previous TAL approaches for the online setting since most previous TAL methods require the whole video to be seen and use the NMS technique to eliminate duplicate proposals.

Online Action Detection (OAD) is a popular online video processing task whose objective is to extract per-frame labels from the streaming video. As it provides per-frame labeling, On-TAL can be solved by taking OAD as an intermediate procedure; training a binary OAD model that distinguishes action frames and grouping them. However, this approach has limitations that are not negligible: action fragmentation and action tick (Figure 2). These problems occur due to the model's unawareness of its decision context, not considering its past decision sequence although it is essential in making a correct decision for the current frame.

Therefore, we propose augmenting context information into the actionness grouping process, forming Context-Aware Actionness Grouping (CAG). Underline the fact that in the CAG setting, the model's decision at the current frame affects what the model will decide in the future. Incorporating this recurrency to training is the main objective of our paper because standard supervised learning methods cannot tackle it properly. Hence, we formulated CAG as a Markov Decision Process (MDP), and tried to apply reinforcement learning [33]. With this, the transition dynamics can be naturally integrated into the training process.

Nevertheless, what would be the best reward scheme for our MDP still remains unclear. This is critical because misdesigned hand-crafted reward function would lead to sub-optimal policy. In order to resolve the issue, we adopt the imitation learning (IL) [27], which tries to follow not only an immediate expert action at a state but also the whole trajectory produced by the expert policy. By this, we can avoid an exhaustive search for the optimal reward function for our MDP.

We evaluate our models on two popular video datasets, THUMOS14 [18] and Activitynet1.3 [6]. We compare our model with various baselines and show that our model, CAG with imitation learning, significantly outperforms the other models. Moreover, as our model also can be used as online detection of action start (ODAS), we evaluated our model on the ODAS task. Surprisingly, our model outperforms state-of-the-art ODAS models.

The main contribution of our paper is summarized as follows:

- We introduce a new challenging task, Online Temporal Action Localization (On-TAL). Due to its ability to process streaming videos and promptly respond with instance-level information of actions, it opens the door for various real-time video understanding applications, for which previous video-related tasks cannot be used.
- For On-TAL, we devise an original framework, CAG, which takes into account not only the frame context but also the model's own decision context. To train CAG, we formalize the CAG framework into a Markov Decision Process (MDP) and propose a novel training method using imitation learning.

- Its effectiveness and robustness are validated by extensive experiments on popular video datasets: THUMOS14 and Activitynet1.3. Besides, our model can also be used for ODAS, outperforming previous ODAS algorithms.

2. Related Work

2.1. Temporal Action Localization

Temporal Action Localization (TAL) is analogous to object detection in the image domain and there are numerous works on this topic [32, 30, 5, 8, 44, 22, 43, 39, 4, 21, 42, 23, 19, 1] including [41] that exploited reinforcement learning to solve the problem. More extensive reviews on existing TAL works can be found in [37].

Our method is influenced by the Temporal Actionness Grouping (TAG), proposed in [44]. However in our setting, actionness grouping must be done without accessing future frames, while TAG exploits a full video and utilizes the post-processing of generated proposals such as grouping of primitive proposals to avoid action fragmentation and NMS to eliminate duplicates.

2.2. Online Video Understanding

There are online video understanding tasks that share the similar goal to the proposing On-TAL task, namely Online Action Detection (OAD) and Online Detection of Action Start (ODAS).

Online Action Detection (OAD), which was first proposed in [9], is a task to extract per-frame class label. Various methods have been suggested to improve the performance of OAD [38, 11], including [13] which utilized reinforcement learning to enable earlier action detection. As it provides *dense* class score for each frame, it can be regarded as a good intermediate representation for further processing, but direct application to real-world problem is limited.

On the other hand, Online Detection of Action Start (ODAS) [31], whose goal is to detect the start of an action as soon as possible, generates *sparse* start point predictions, making the task more practical. Especially, [14] designed LocNet which utilizes reinforcement learning to pose implicit *sparsity constraint* over the decision context. Note that one of our baselines, the CAG agent trained with handcrafted reward scheme (CAG-RL in Table 1, 2, 4), can be seen as a direct extension of [14] to On-TAL.

Overall, previously proposed online video processing tasks mainly focus on frame-level information. On the contrary, On-TAL provides action instances that have richer semantics, which enables its direct deployment to real-world computer vision problems. Moreover, as an action instance contains an action start point as its element, a model that can deal with On-TAL automatically solves ODAS problem, indicating that On-TAL can be seen as higher-level task than ODAS.

3. Online Temporal Action Localization

Assume an untrimmed video $V = \{x_\tau\}_{\tau=1}^{T}$ with M action instances $\Psi = \{\psi_m\}_{m=1}^{M} = \{(s_m, e_m, c_m)\}_{m=1}^{M}$, where x_τ denotes τth frame and s_m, e_m, and c_m represent the start frame index, the end frame index, and the class label of the mth action instance ψ_m respectively. As we are in the online setting, x_τ is provided in a serial order.

Following recent temporal proposal generation methods [5, 12, 22, 10], consecutive k frames are converted to a visual feature f and all the following steps, including online generation of action proposals, run on this feature sequence. A model for this proposal generation should emit appropriate action proposal as soon as it detects the action end. That is, the model should decide whether or not to generate an action proposal for every feature f it encounters. The granularity of the feature sequence may matter for On-TAL; smaller k would result in finer decision making, leading to possibly more accurate but noisy proposals while bigger k would work in the opposite way.

The final goal of On-TAL is to recover Ψ by aggregating each online-generated action instance ψ without any NMS-like post-processing.

On-TAL can be solved by extending the OAD framework if we assume that there are no overlaps among action instances. In this setting, consecutive k frames are converted to a feature f by an encoder E, resulting in feature sequence $\{f_t\}_{t=1}^{\lceil T/k \rceil}$, instead of raw frame sequence $\{x_\tau\}_{\tau=1}^{T}$. At each timestep t, the OAD model M takes a feature f_t as its input and outputs an actionness score $\alpha_t (0 < \alpha_t < 1)$, where α_t is close to 1 if the feature contains an action and 0 otherwise. Simply grouping $\{\alpha_t | \alpha_t > threshold\}$ in the online manner would result in action instances.

4. Proposed Method

4.1. Context-Aware Actionness Grouping (CAG)

Even though the naive OAD extension can deal with the frame context by using a stateful RNN architecture for the OAD model M, it still cannot take the decision context into account. To this end, we propose a new framework to solve On-TAL, Context-Aware Actionness Grouping (CAG), which is described in Algorithm 1.

In CAG (Figure 3), we adopted a new component, context-aware agent Υ, which takes two historic queues (actionness queue q_a and decision queue q_d with both lengths n) and the current frame's class probability p_t as its input and returns a discrete output $d \in \{0, 1\}$. To generate action instances, the model aggregates $\{d_t | d_t = 1\}$, instead of grouping $\{\alpha_t | \alpha_t > threshold\}$ as it did in straightforward OAD extension.

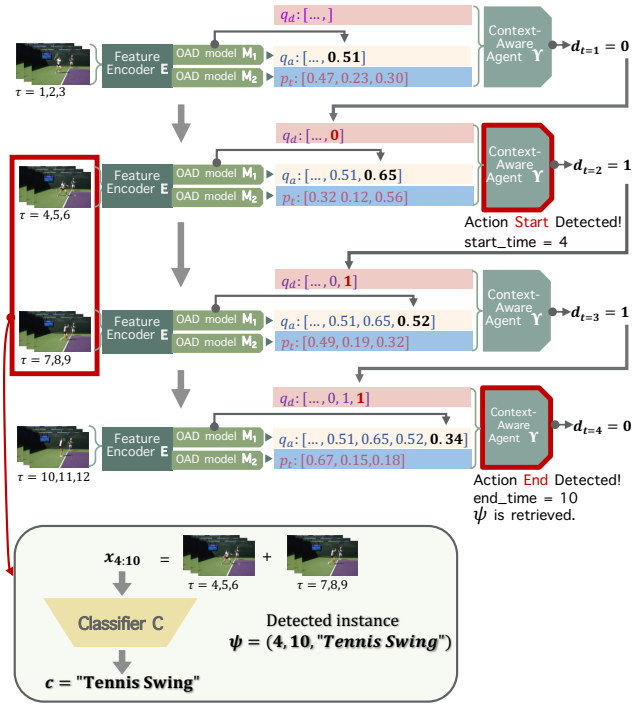

Figure 3. Overview of our approach (CAG) where $k = 3$ and the number of classes is 3 including background class. In this case, an action instance $\psi = (4, 10, \text{“}TennisSwing\text{”})$ is retrieved at timestep $t = 4$. In On-TAL setting, the model should return **nothing** at timestep $t = 1, 2, 3$, which makes the task more challenging.

4.2. CAG as a Markov Decision Process

In CAG, the decision of prior timestamps is taken into account to make a decision in the current frame, which makes the training of Υ complicated. Ignoring this recurrency, one can directly map current states (q_a, q_d, p) to Υ's output d using the standard supervised learning method, which is also called as Behavioral Cloning (BC) [26]. However, BC-trained agent cannot "plan" to lead itself to good states since BC completely neglects the transition dynamics. Furthermore, due to the compounding error, the agent cannot decide which action would be appropriate if the agent encounters out-of-distribution states [29]. These drawbacks prevent the BC-trained agent from achieving good performance, which can be seen in our experiments.

In order to model the recurrency, we formulate the problem into a Markov Decision Process (MDP). At timestep t, a state s_t is represented as $\{q_a^t; q_d^t; p_t\}$, where $q_a^t = [\alpha_{t-n}, ..., \alpha_t]$ and $q_d^t = [d_{t-n-1}, ..., d_{t-1}]$[1]. The decision d_t [2] lies on a discrete decision space $\{0, 1\}$, where $d_t = 1$ denotes that $x_{k(t-1):kt}$ are action frames and background otherwise. Unlike [34], the transition dynamic of the MDP is stochastic because the agent would face unknown α_{t+1} and p_{t+1} at timestep $t + 1$. In other words, the model has only partial control for the states that it will encounter in the future.

[1] Initialization technique for q_a and q_d at the first timestep can be found in the supplementary material.

[2] d_t should be called as an *action* if we use normal MDP terms. But to avoid confusion with actionness α, we denote it as a *decision*.

Algorithm 1: Context-Aware Actionness Grouping (CAG) at Inference Stage

Component:
Feature Encoder E which takes k consecutive frames and outputs a frame feature,
OAD model M_1 for actionness output,
OAD model M_2 for class probability output,
Context-Aware Agent Υ,
Classifier C,
Input: Video Stream $\{x_\tau\}_{\tau=1}^{T}$
Output: Action instance set Ψ

$\Psi \leftarrow \phi$
$d_{prev} \leftarrow 0$
$q_d.initialize()$
$q_a.initialize()$
for $t \leftarrow 1$ **to** $\lceil T/k \rceil$ **do**
  $f_t \leftarrow E(x_{k(t-1):kt})$
  $\alpha_t \leftarrow M_1(f_t)$
  $p_t \leftarrow M_2(f_t)$
  $q_a.dequeue()$ // Remove the front-most element.
  $q_a.enqueue(\alpha_t)$
  $d \leftarrow \Upsilon(q_a, q_d, p_t)$
  if $d_{prev} = 0$ *and* $d = 1$ **then**
    $s \leftarrow k(t-1) + 1$
  else if $d_{prev} = 1$ *and* $d = 0$ **then**
    $e \leftarrow k(t-1) + 1$
    $c \leftarrow C(x_{s:e})$
    $\psi \leftarrow (s, e, c)$
    $\Psi \leftarrow \Psi \cup \psi$
  $q_d.dequeue()$
  $q_d.enqueue(d)$
  $d_{prev} \leftarrow d$
end

4.3. Training method for CAG

4.3.1 CAG with Reinforcement Learning

Reinforcement Learning (RL) makes a model to be capable of planning in the MDP setting because its objective is to maximize the *cumulative* reward of the trajectory, not the *immediate* one. In other words, RL enables the model to be conscious of the transition dynamics of the MDP and to be foresighted, solving the BC's main limitation.

Nonetheless, what would be a fair reward scheme for our MDP still remains problematic. Note that better frame-wise

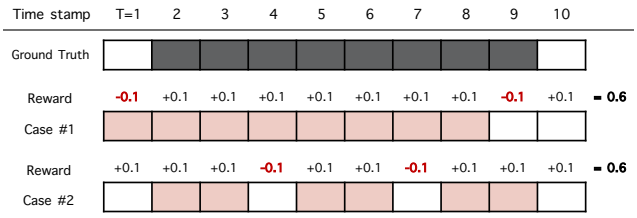

Figure 4. In the case of frame-wise OAD, the reward scheme would be straightforward; $+0.1$ for the right decision and -0.1 for the wrong decision. However, this reward scheme assigns the same cumulative rewards in case #1 and case #2. Apparently, in the perspective of CAG, case #1 is better than case #2. But the RL-trained model does not prefer case #1 to case #2 because the cumulative rewards of the both cases are the same. The *best* frame-wise OAD model would be the best CAG model, but a *good* frame-wise OAD model is **NOT** necessarily a good CAG model.

OAD performance does not guarantee better action proposal performance as shown in Figure 4.

One way to resolve this issue is to apply sophisticated reward functions. For example, complimenting successively correct decisions would prevent the action fragmentation. However, this reward function will assign excessive rewards to long action segments and do nothing about the action ticks. Therefore, when we apply RL to train CAG, an exhaustive search for the adequate reward function is inevitable.

4.3.2 CAG with Imitation Learning

In Imitation Learning (IL), the goal of a task is defined by given expert trajectories. That is, no reward signal is available and the model should figure out what is a good policy by only using the given expert transitions $(state, action, next\_state)$. Since a model's training procedure should incorporate MDP's transition dynamics, many IL methods restore the reward function $R(state, action)$ from given expert transitions.

SQIL [27] suggests a completely different approach. The basic intuition is very simple; as our ultimate goal is to *imitate* the expert, an expert transition should be considered as a *good* transition, meaning that the reward of the expert transition should be set to 1 while a reward of an agent's transition set to 0. These transitions are stored in the replay buffer and used when the soft-Q learning [15] runs. SQIL tends to be more stable than other method that utilizes adversarial training [16]. Moreover, since SQIL directly approximates the Q function without using a distinct network to estimate the reward function $R(state, action)$, SQIL is more parameter-efficient than other methods that involve reward function approximation.

As the imitation learning fully exploits the MDP structure without requiring a pre-defined reward scheme, it can alleviate previously mentioned problems of BC and RL. Therefore, our final training scheme for CAG adopts the imitation learning, specifically SQIL.

Nevertheless, we empirically found that a direct application of SQIL to CAG yields unsatisfactory results. We attribute it to the mismatch between our task and the maximum entropy assumption [45] of the soft-Q setting, since CAG will cause action ticks or fragmentation with a single wrong decision.

Thus, we adopt hard-Q variant of SQIL, which is similar to the popular DQN method [24]. The only difference from the original DQN is that we used given expert transitions with constant reward $+0.1$ and agent transitions with constant reward -0.1 to approximate the Q-function. To be specific, Q network $Q_\theta(s, d)$ is updated as follows:

$$\theta \leftarrow \theta - \eta\nabla_\theta(\delta^2(D_{expert}, +0.1) + \delta^2(D_{agent}, -0.1)),$$
$$\text{where } \delta^2(D, r) \triangleq$$
$$\frac{1}{|D|}\sum_{(s_t,d_t,s_{t+1})\in D}(Q_\theta(s_t, d_t) - (r + \gamma\max_{d_{t+1}} Q_\theta(s_{t+1}, d_{t+1}))^2. \quad (1)$$

In the above equation, D is a minibatch which is comprised of transitions that have come from its subscript, while γ and η denote the discount factor and the learning rate respectively. We will call this method as a Q-Imitation Learning (QIL) for the rest of this paper, just subtracting the "soft" prefix from SQIL.

The whole training procedure is *two-staged*, meaning that the OAD model M_1, M_2 and the Context-Aware Agent Υ are trained separately. For the first stage of training, the binary OAD model M_1 and the multi-class OAD model M_2, which are both composed of a simple one layer LSTM [17] network and two additional FC layers, are trained with the cross entropy loss. After that, α (actionness) and p_t (class probability) sequences for all training videos are calculated using the trained OAD models. From the calculated sequences, the expert database for the imitation learning is constructed as Figure 5.

In the second stage of training, QIL (Equation (1)) is conducted in the MDP *environment* (Figure 6) to train the Context-Aware agent Υ, which consists of two FC layers with LeakyReLU activation function. For proposal classification, independent TSM [20] classifier C is used and a class probability from the classifier is regarded as a confident score when we compute mAP.

5. Experiments

5.1. Datasets and Features

We validate our method on two standard datasets: THUMOS14 [18] and Activitynet1.3 [6]. THUMOS14 contains 413 untrimmed videos with 20 action classes and it is split

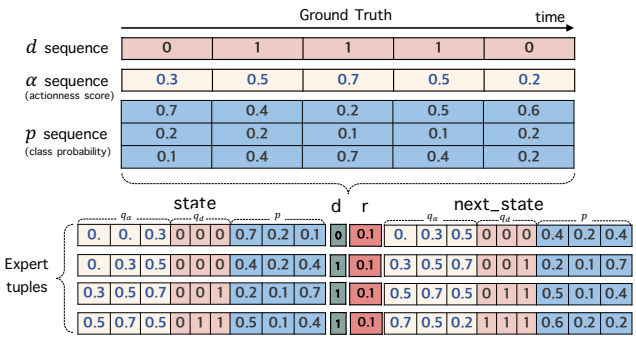

Figure 5. Expert database construction. Here, we assume that the number of classes is 3, as can be seen by a p sequence having 3 rows. From the given ground truth d, α, and p sequences, we can extract four expert transitions, consisting of state, decision (d), reward (r), and next_state. Aggregating these tuples, we can construct an expert database.

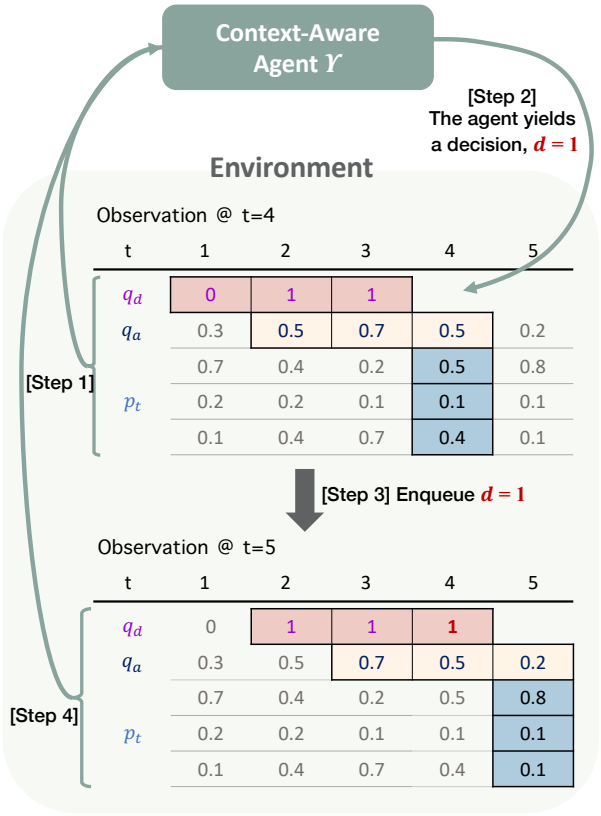

Figure 6. Snapshot of the MDP at $t = 4$. α and p sequences are pre-calculated in the first stage of training and rewards are set to -0.1 for all the agent transitions if we are in CAG-QIL setting.

into 200 training videos and 213 test videos. There are more than 15 action instances per video in the dataset. Activitynet1.3 consists of 19,994 untrimmed videos and the videos are divided into training, validation, and test sets by the ratio of 2:1:1. Unlike THUMOS14, videos in Activitynet1.3 only have 1.5 action instances per video, indicating that the dataset is not most suitable for our task because our main objective is to detect multiple action instances in the streaming setting.

For a frame feature f, 6 consecutive frames ($k = 6$) are put into the two-stream TSN [35] trained on Kinetics [7] and its outputs are used. On Activitynet1.3, we rescaled the feature sequence of each video to length 100 by linear interpolation, following the convention in TAL literatures [39, 22, 19, 1].

However when it comes to evaluating ODAS performance in THUMOS14, we used Activitynet pretrained features to ensure a fair comparison with other ODAS works. [31, 14]

5.2. Evaluation Metric

For TAL metric, we used mean Average Precision (mAP) to enable fair comparison between our model and the other *offline* TAL models. We report mAPs with multiple tIOUs in a set $\{0.3, 0.4, 0.5, 0.6, 0.7\}$ for THUMOS14 and $\{0.5, 0.75, 0.95\}$ for Activitynet1.3.

To evaluate ODAS performance, we measure the point-level average precision (p-AP) and calculate p-mAP by averaging p-AP over all the action classes, following previous works [31, 14]. For each action class, detected start points are sorted in descending order according to their confidence scores and then AP is measured accordingly. Each action start point is counted as a true positive only when its action class is correct and its temporal distance from a ground truth point is smaller than the offset. As done in previous works, duplicate predictions are not allowed for the same ground truth point.

For the reason that our model uses OAD output as its intermediate representation and does not provide frame-wise labeling, comparison among the other OAD models is inappropriate. Note that the context-aware agent's objective is to post-process the OAD output and produces valid action instances.

5.3. Baselines

We compare our method to multiple baselines to demonstrate the effectiveness of our approach.

OAD-Grouping model generates an action instance by grouping the result of the binary OAD model. It is the same model described in the Section 3 with a constant $threshold$.

OAD-Grouping w/ Hindsight Threshold model assigns different $threshold$ value for each class which results in the highest class AP. We denote the model as "*hindsight* threshold model" because we exploit the *test set* and conduct grid search in ([0.3:0.05:0.7]) with *hindsight*, to find the best-performing threshold per each class. In this setting, mAP is calculated by averaging the highest AP of each class. This can be considered as *upper bound* of sophisticated $threshold$ tuning because it uses the different

Method	0.3	0.4	0.5	0.6	0.7
OAD-Grouping	33.3	28.0	22.0	16.8	10.4
w/ Hindsight Threshold	35.9	30.3	24.5	18.6	12.1
w/ Temporal Smoothing	38.3	32.4	24.9	18.2	10.7
CAG-BC	6.4	4.8	3.3	2.2	1.5
CAG-RL	32.8	27.0	22.2	16.8	10.8
CAG-QIL w/o p_t	43.0	34.9	27.2	19.6	12.4
CAG-QIL	**44.7**	**37.6**	**29.8**	**21.9**	**14.5**

Table 1. On-TAL results on test set of THUMOS14. Note that CAG-QIL outperforms all the baselines with large margin. Moreover, the ablation study of augmenting p to the state representation proves that including class probability as a part of the state can improve CAG-QIL's performance.

Method	0.5	0.75	0.9
OAD-Grouping	28.1	15.7	3.3
CAG-BC	9.5	7.4	3.4
CAG-RL	21.4	11.3	2.2
CAG-QIL	**30.5**	**18.5**	**4.1**

Table 2. Comparison among QIL, RL, and BC result in validation set of ActivityNet1.3 in terms of mAP@tIoU. Only CAG-QIL surpassed the baseline model's performance.

best-performing $threshold$ for each class when calculating mAP.

OAD-Grouping w/ Temporal Smoothing additionally applies temporal smoothing filter to the output of the binary OAD model. Specifically, we adopted an average filter with the size of k. We tried various size of $k \in \{3, 5, 7, 9, 11, 13\}$ and found that $k = 5$ is the best performing filter size. In this way, action tick and fragmentation can be alleviated in some extent.

CAG-BC learns a direct mapping from the state $(q_a; q_d; p)$ to the decision d using the standard supervised learning method (also known as Behavior Cloning). Note that CAG-BC uses the same input and output form with CAG-QIL but they have different algorithms for learning a mapping.

CAG-RL denotes an agent trained with hand-crafted reward scheme (frame-wise OAD reward), which is depicted in the Figure 4. Like CAG-QIL, the DQN algorithm [24] is deployed to conduct reinforcement learning.

5.4. Result on On-TAL

Table 1 clearly shows the effectiveness of proposed method, CAG-QIL, in THUMOS14 dataset. It would be worth noting that OAD grouping models with additional tricks are not directly applicable to real-world On-TAL problem, as the model with hindsight threshold uses test set to select the best per-class threshold value and the model with temporal smoothing cannot run online. Beating the OAD grouping models with the non-naive tricks with large margin indicates that the CAG-QIL has exceptional ability to deal with On-TAL. Furthermore, the result demonstrated in the table 2 tells us that CAG-QIL also performs well in the other dataset; Activitynet1.3.

		Method	0.3	0.4	0.5	0.6	0.7
Offline	Two-Stage	S-CNN [32]	36.3	28.7	19.0	10.3	5.3
		CDC [30]	40.1	29.4	23.3	13.1	7.9
		SST [5]	41.2	31.5	20.0	10.9	4.7
		SSN [44]	51.9	41.0	29.8	-	-
		BSN [22]	53.5	45.0	36.9	28.4	20.0
		TAL-Net [8]	53.2	48.5	42.8	33.8	20.8
		G-TAD [39]	54.5	47.6	40.2	30.8	23.4
		G-TAD+P-GCN [39]	66.4	60.4	51.6	37.6	22.9
	One-Stage	End-to-End learning [41]	36.0	26.4	17.1	-	-
		SMS [42]	36.5	27.8	17.8	-	-
		SSAD [21]	43.0	35.0	24.6	-	-
		SS-TAD [4]	45.7	-	29.2	-	9.6
		GTAN [23]	57.8	47.2	38.8	-	-
Online		**CAG-QIL**	44.7	37.6	29.8	21.9	14.5

Table 3. Comparison of mAP@tIoU with various *offline* TAL methods in THUMOS14 testing set.

Method	0.3		0.4		0.5		0.6		0.7	
	Avg	SD	Avg	SD	Avg	SD	Avg	SD	Avg	SD
CAG-RL	0.4	1.5	-0.1	1.3	-0.1	0.8	0.0	0.8	0.0	0.8
CAG-BC	-25.2	5.4	-21.6	3.8	-17.2	2.7	-13.1	1.9	-8.6	1.0
CAG-QIL	12.6	1.1	9.7	0.6	7.5	0.7	5.2	1.2	3.5	1.5

Table 4. Comparison among QIL, RL, and BC result in THUMOS'14 testing set in terms of mAP@tIoU. A positive value means the relative improvement by an algorithm to the OAD baseline, while a negative value denotes degradation. Average and standard deviation are calculated with four experimental results to show the robustness of our method.

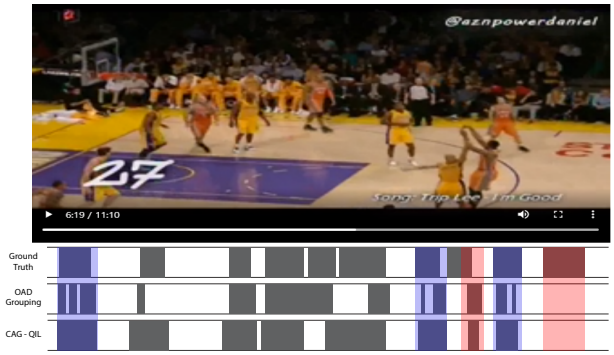

Figure 7. Screenshot of our demo program to demonstrate CAG-QIL's qualitative result. As we can see in the blue region, our model tends to emit *merged* proposals, resulting in significant increment of the mAP score (Table 1). However, since context-aware agent Υ takes pre-calculated α sequence as its input, it would lead to wrong action proposal if α sequence is miscalculated. This failure cases can be found in the red region.

Offsets (second)	1	2	3	4	5	6	7	8	9	10
Shou et al. [31]	3.1	4.3	4.7	5.4	5.8	6.1	6.5	7.2	7.6	8.2
ClsNet-only [14]	13.9	21.6	25.8	28.9	31.1	32.5	33.5	34.3	34.8	35.2
StartNet-CE [14]	17.4	25.4	29.8	33.0	34.6	36.3	37.2	37.7	38.6	38.8
StartNet-PG [14]	19.5	27.2	30.8	33.9	36.5	37.5	38.3	38.8	39.5	39.8
CAG-QIL (Ours)	**20.3**	**31.2**	**37.2**	**41.4**	**44.2**	**46.0**	**47.3**	**48.1**	**48.9**	**49.8**

Table 5. Performance on online detection of action start (ODAS) on THUMOS14. Note that CAG-QIL is NOT specifically trained for ODAS and yet outperform the current state-of-the-art method (StartNet) in the ODAS task.

As the first work on On-TAL, there are no available methods to compare within the same condition. Therefore, we compare our results with recent *offline* TAL methods in Table 3, as the outputs of both tasks are the same. While there is still performance gap between our method and state-of-the-art *offline* temporal action localization methods, our method's performance is comparable to recent one-stage *offline* approaches despite the strong constraints described in the earlier section. Note that only [41] and ours do not have NMS-like post processing in Table 3.

To validate our model's robustness, we conducted 3 more experiments for each model in THUMOS14. For each trial, we used different weights for the OAD models M_1, M_2. Remember that if we use different weight for the OAD models, the pre-calculated α and p sequences of videos vary, drawing different data for the CAG training. Since every trial has different baseline (OAD-Grouping model) performance, we report *differences* between the algorithms' and the baseline's mAP score, showing the algorithm's improvement over the baseline's (Table 4). The result proves that CAG-QIL steadily outperforms not only the baseline but also CAG with reinforcement learning, while CAG-BC only shows constant deterioration.

Figure 7 qualitatively illustrates the effect of CAG-QIL. Note that OAD-Grouping and CAG-QIL have the same α sequence as their input since they share the identical OAD model M_1. CAG-QIL successfully resolves the action fragmentation issue by exploiting its decision context if the α sequence is reasonably calculated.

Throughout the experimental results, it is interesting to see that the CAG-RL hardly improves TAL performance. This is because the frame-wise OAD reward cannot penalize action fragmentation as we anticipated in the Figure 4. It can be proven by the fact that the CAG-RL does not reduce the proposal number (5641) than the OAD-Grouping baseline (4841) in THUMOS14 test set.

Owing to the light-weighted OAD models and context-aware agent, our algorithm is computationally cheap, which implies that there is no problem in running it online. Instead, we found that the main bottleneck is in the optical flow computation (used for OAD) that takes 126ms for 6 stacked frames. In our experiment, our full pipeline runs in 29.4 fps.

5.5. Result on ODAS

As our model can promptly yield an action start point as a by-product of the action instance generation, we can evaluate the start point generation performance with a metric for online detection of action start (ODAS) task [31, 14] whose goal is to detect the occurrence and class of action start as soon as the action happens. But since our classification procedure is deferred to the action end and uses a full action instance, direct comparison would result in unfair advantage to our model. To avoid this and ensure fair comparison, we used multi-class OAD model M_2's output p_t, where t denotes detected action start timestep, for the class probability of the action start point instead of using independent classifier C's output.

Table 5 shows the comparison results between the current state-of-the-art model in ODAS and our model. Surprisingly, our model outperforms StartNet [14] at every time offsets even though our model was not specifically trained to solve the ODAS task. This demonstrates that our model is good enough to detect accurate start point.

6. Conclusion

In this paper, we first formalized a new challenging task, Online Temporal Action Localization (On-TAL) and proposed Context-Aware Actionness Grouping via Q Imitation Learning (CAG-QIL) framework as a solution. Our model showed comparable mAP score to recent one-stage offline TAL works, and outperformed the state-of-the-art ODAS performance. Future work will include improving the performance and also devising an end-to-end framework for the task.

Acknowledgement

This work was conducted by Center for Applied Research in Artificial Intelligence(CARAI) grant funded by Defense Acquisition Program Administration(DAPA) and Agency for Defense Development(ADD) (UD190031RD), and also by Institute of Information & communications Technology Planning & evaluation (IITP) grant funded by the Korea government(MSIT), Artifitial Intelligence Graduate School Program, Yonsei University, under Grant 2020-0-01361

References

[1] Yueran Bai, Yingying Wang, Yunhai Tong, Yang Yang, Qiyue Liu, and Junhui Liu. Boundary content graph neural network for temporal action proposal generation. In *European Conference on Computer Vision*, 2020. 1, 3, 6

[2] Philipp Bergmann, Tim Meinhardt, and Laura Leal-Taixe. Tracking without bells and whistles. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019. 1

[3] Navaneeth Bodla, Bharat Singh, Rama Chellappa, and Larry S. Davis. Soft-nms – improving object detection with one line of code. In *Proceedings of the IEEE International Conference on Computer Vision*, 2017. 2

[4] Shyamal Buch, Victor Escorcia, Bernard Ghanem, Li Fei-Fei, and Juan Carlos Niebles. End-to-end, single-stream temporal action detection in untrimmed videos. *British Machine Vision Conference*, 2019. 3, 7

[5] Shyamal Buch, Victor Escorcia, Chuanqi Shen, Bernard Ghanem, and Juan Carlos Niebles. Sst: Single-stream temporal action proposals. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017. 1, 3, 7

[6] Fabian Caba Heilbron, Victor Escorcia, Bernard Ghanem, and Juan Carlos Niebles. Activitynet: A large-scale video benchmark for human activity understanding. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015. 2, 5

[7] Joao Carreira and Andrew Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017. 6

[8] Yu-Wei Chao, Sudheendra Vijayanarasimhan, Bryan Seybold, David A Ross, Jia Deng, and Rahul Sukthankar. Rethinking the faster r-cnn architecture for temporal action localization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018. 1, 3, 7

[9] Roeland De Geest, Efstratios Gavves, Amir Ghodrati, Zhenyang Li, Cees Snoek, and Tinne Tuytelaars. Online action detection. In *Proceedings of the European Conference on Computer Vision*, 2016. 3

[10] Victor Escorcia, Fabian Caba Heilbron, Juan Carlos Niebles, and Bernard Ghanem. Daps: Deep action proposals for action understanding. In *Proceedings of the European Conference on Computer Vision*, 2016. 3

[11] Hyunjun Eun, Jinyoung Moon, Jongyoul Park, Chanho Jung, and Changick Kim. Learning to discriminate information for online action detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2020. 3

[12] Jiyang Gao, Zhenheng Yang, Kan Chen, Chen Sun, and Ram Nevatia. Turn tap: Temporal unit regression network for temporal action proposals. In *Proceedings of the IEEE International Conference on Computer Vision*, 2017. 3

[13] Jiyang Gao, Zhenheng Yang, and Ram Nevatia. Red: Reinforced encoder-decoder networks for action anticipation. *British Machine Vision Conference*, 2017. 3

[14] Mingfei Gao, Mingze Xu, Larry S Davis, Richard Socher, and Caiming Xiong. Startnet: Online detection of action start in untrimmed videos. In *Proceedings of the IEEE International Conference on Computer Vision*, 2019. 3, 6, 8

[15] Tuomas Haarnoja, Haoran Tang, Pieter Abbeel, and Sergey Levine. Reinforcement learning with deep energy-based policies. In *Proceedings of International Conference on Machine Learning*, 2017. 5

[16] Jonathan Ho and Stefano Ermon. Generative adversarial imitation learning. In *Advances in Neural Information Processing Systems*. 2016. 5

[17] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 1997. 5

[18] Y.-G. Jiang, J. Liu, A. Roshan Zamir, G. Toderici, I. Laptev, M. Shah, and R. Sukthankar. THUMOS challenge: Action recognition with a large number of classes. http://crcv.ucf.edu/THUMOS14/, 2014. 2, 5

[19] Chuming Lin, Jian Li, Yabiao Wang, Ying Tai, Donghao Luo, Zhipeng Cui, Chengjie Wang, Jilin Li, Feiyue Huang, and Rongrong Ji. Fast learning of temporal action proposal via dense boundary generator. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2020. 1, 3, 6

[20] Ji Lin, Chuang Gan, and Song Han. Tsm: Temporal shift module for efficient video understanding. In *Proceedings of the IEEE International Conference on Computer Vision*, 2019. 5

[21] Tianwei Lin, Xu Zhao, and Zheng Shou. Single shot temporal action detection. In *Proceedings of the ACM international conference on Multimedia*, 2017. 3, 7

[22] Tianwei Lin, Xu Zhao, Haisheng Su, Chongjing Wang, and Ming Yang. Bsn: Boundary sensitive network for temporal action proposal generation. In *Proceedings of the European Conference on Computer Vision*, 2018. 1, 3, 6, 7

[23] Fuchen Long, Ting Yao, Zhaofan Qiu, Xinmei Tian, Jiebo Luo, and Tao Mei. Gaussian temporal awareness networks for action localization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019. 3, 7

[24] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 2015. 5, 7

[25] Seoung Wug Oh, Joon-Young Lee, Ning Xu, and Seon Joo Kim. Video object segmentation using space-time memory networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019. 1

[26] Dean A Pomerleau. Efficient training of artificial neural networks for autonomous navigation. *Neural Computation*, 1991. 4

[27] Siddharth Reddy, Anca D Dragan, and Sergey Levine. Sqil: Imitation learning via regularized behavioral cloning. *arXiv preprint arXiv:1905.11108*, 2019. 2, 5

[28] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016. 1

[29] Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-

regret online learning. In *Proceedings of the international Conference on Artificial Intelligence and Statistics*, 2011. 4

[30] Zheng Shou, Jonathan Chan, Alireza Zareian, Kazuyuki Miyazawa, and Shih-Fu Chang. Cdc: Convolutional-deconvolutional networks for precise temporal action localization in untrimmed videos. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017. 1, 3, 7

[31] Zheng Shou, Junting Pan, Jonathan Chan, Kazuyuki Miyazawa, Hassan Mansour, Anthony Vetro, Xavier Giro-i Nieto, and Shih-Fu Chang. Online detection of action start in untrimmed, streaming videos. In *Proceedings of the European Conference on Computer Vision*, 2018. 3, 6, 8

[32] Zheng Shou, Dongang Wang, and Shih-Fu Chang. Temporal action localization in untrimmed videos via multi-stage cnns. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016. 1, 3, 7

[33] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. 2018. 2

[34] Borui Wang, Ehsan Adeli, Hsu-kuang Chiu, De-An Huang, and Juan Carlos Niebles. Imitation learning for human pose prediction. In *Proceedings of the IEEE International Conference on Computer Vision*, 2019. 4

[35] Limin Wang, Yuanjun Xiong, Zhe Wang, Yu Qiao, Dahua Lin, Xiaoou Tang, and Luc Van Gool. Temporal segment networks: Towards good practices for deep action recognition. In *Proceedings of the European Conference on Computer Vision*, 2016. 6

[36] Zhongdao Wang, Liang Zheng, Yixuan Liu, and Shengjin Wang. Towards real-time multi-object tracking. *arXiv preprint arXiv:1909.12605*, 2019. 1

[37] Huifen Xia and Yongzhao Zhan. A survey on temporal action localization. *IEEE Access*, 2020. 3

[38] Mingze Xu, Mingfei Gao, Yi-Ting Chen, Larry S Davis, and David J Crandall. Temporal recurrent networks for online action detection. In *Proceedings of the IEEE International Conference on Computer Vision*, 2019. 3

[39] Mengmeng Xu, Chen Zhao, David S Rojas, Ali Thabet, and Bernard Ghanem. G-tad: Sub-graph localization for temporal action detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2020. 1, 3, 6, 7

[40] Linjie Yang, Yuchen Fan, and Ning Xu. Video instance segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019. 1

[41] Serena Yeung, Olga Russakovsky, Greg Mori, and Li Fei-Fei. End-to-end learning of action detection from frame glimpses in videos. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016. 1, 3, 7, 8

[42] Zehuan Yuan, Jonathan C Stroud, Tong Lu, and Jia Deng. Temporal action localization by structured maximal sums. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017. 3, 7

[43] Runhao Zeng, Wenbing Huang, Mingkui Tan, Yu Rong, Peilin Zhao, Junzhou Huang, and Chuang Gan. Graph convolutional networks for temporal action localization. In *Proceedings of the IEEE International Conference on Computer Vision*, 2019. 1, 3

[44] Yue Zhao, Yuanjun Xiong, Limin Wang, Zhirong Wu, Xiaoou Tang, and Dahua Lin. Temporal action detection with structured segment networks. In *Proceedings of the IEEE International Conference on Computer Vision*, 2017. 1, 3, 7

[45] Brian D Ziebart, Andrew L Maas, J Andrew Bagnell, and Anind K Dey. Maximum entropy inverse reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2008. 5