A Sliding Window Scheme for Online Temporal Action Localization

Young Hwi Kim[1], Hyolim Kang[1], and Seon Joo Kim[1]

Yonsei University, 50 Yonsei-ro, Seodaemun-gu, Seoul, Republic of Korea
{younghwikim,hyolimkang,seonjookim}@yonsei.ac.kr

Abstract. Most online video understanding tasks aim to immediately process each streaming frame and output predictions frame-by-frame. For extension to instance-level predictions of existing online video tasks, Online Temporal Action Localization (On-TAL) has been recently proposed. However, simple On-TAL approaches of grouping per-frame predictions have limitations due to the lack of instance-level context. To this end, we propose Online Anchor Transformer (OAT) to extend the anchor-based action localization model to the online setting. We also introduce an online-applicable post-processing method that suppresses repetitive action proposals. Evaluations of On-TAL on THUMOS'14, MUSES, and BBDB show significant improvements in terms of mAP, and our model shows comparable performance to the state-of-the-art offline TAL methods with a minor change of the post-processing method. In addition to mAP evaluation, we additionally present a new online-oriented metric of early detection for On-TAL, and measure the responsiveness of each On-TAL approach.

Keywords: Online Video Understanding, Temporal Action Localization

1 Introduction

The rising amount of video production has increased the need for processing untrimmed videos. To automate the processing of untrimmed video, many researchers have attempted to solve the problems with deep neural networks (DNN). Temporal Action Localization (TAL) is one of the major untrimmed video understanding tasks, which detects the action instances with the class and the boundary information. Significant progress has been made in TAL recently by employing DNN [30,28,31,33,18,34].

As a challenging format of untrimmed videos, streaming video processing is receiving more attention in the surveillance system, live streaming services, and autonomous driving systems. Online Action Detection (OAD) is a task that takes one frame from the stream and immediately predicts the action class including the non-action class. In addition, Online Detection of Action Start (ODAS) finds the action starting point as early as possible even before the start of an action. These tasks mainly target the high responsiveness of action detection.

However, highly responsive systems are limited to specific applications, especially alarm systems, due to their frame-level output format. Although most video understanding tasks start from an action instance, aforementioned tasks do not consider instance-level contexts and are hard to apply to other applications.

To address these limitations on existing online video understanding tasks, an instance-level online video understanding task named Online Temporal Action Localization (On-TAL) [12] was recently introduced. On-TAL has two main differences with offline TAL: 1) by the online constraint, future frames cannot be accessed, and 2) modification of generated proposals in the past is not allowed, so the post-processing methods can only be applied on the currently generated proposals.

From these constraints, one of the obvious approaches to On-TAL is to get per-frame action predictions from the OAD backbone framework and to group them into action instances. However, grouping per-frame predictions fundamentally causes three issues of tick, fragmentation, and merging. The predicted action probabilities from OAD may fluctuate at non-action frames or in the middle of an action, and make ticks of an action instance or fragmented actions. In the case of consecutive actions, the OAD based methods cannot separate the action instances, making one merged action instance by the same action class.

To tackle these issues, we start with a TAL approach with the intuition that instance-level supervision would solve the limitations of the OAD based approaches. As we cannot access the future context by the first constraint of On-TAL, revisiting the sliding window scheme becomes a valid approach for On-TAL. The TAL methods with the sliding window approach [23,7] are not being actively utilized in recent works due to its restricted use of video contexts. Building rich video context has been receiving more attention lately [30,28,18,31,34] for TAL. However, we revisit the sliding window approach for online TAL and show its potential by reorganizing the framework and exploiting the state-of-the-art context encoders.

In this paper, we spotlight the sliding window scheme again, and propose an anchor-based model named Online Anchor Transformer (OAT) for On-TAL task. OAT employs the transformer encoder/decoder to encode a sliding window from each timeline and decode several anchor features from the encoded window features and the anchor query. The decoded anchor features are used to classify the action class and to regress the action end offset and the action length for the refinement of action boundaries.

Aside from accurate detection, another advantage of adopting the anchor-based approach for On-TAL is the ability to detect an action before the action ends. The anchor-based approach can make the action proposals earlier than the action ends, and those proposals are refined by the offset regression. On the contrary, OAD-based approaches must wait until the action ends as depicted in Fig. 1. Since the responsiveness is critical in online video understanding, we introduce a new online-oriented metric for On-TAL named Average Early Detected Time (AEDT), and compare it with other baseline On-TAL methods.

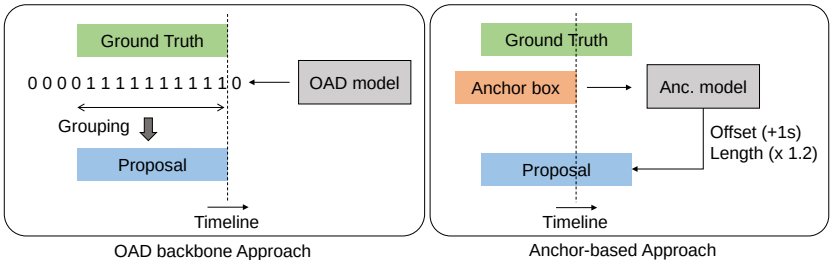

Fig. 1: Comparison of the OAD backbone approach and the anchor-based approach for Online Temporal Action Localization. When making an action instance proposal, the OAD backbone approach have to wait until the model predicts the end of an action. In contrast, Anchor-based approach can make proposals earlier than action ends, and refine them by the offset regression.

The anchor-based approach is not a master key, of course, since it produces a large amount of action proposals. Most TAL methods employ the post-processing of non-maximum suppression (NMS) to control excessive number of proposals, but this is not applicable to On-TAL by its second constraint. To deal with this, we propose Online Suppression Network (OSN) that approximates NMS without the use of the future context.

We evaluated our model on three online-applicable datasets, THUMOS'14 [11], MUSES [19], and BBDB [20]. Our method shows the state-of-the-art On-TAL performance by a significant margin. In addition, by simply changing the post-processing method to NMS, our model shows comparable performance compared to the state-of-the-art offline TAL methods, showing the potential of the sliding window scheme.

In summary, our contributions are described as follows:

- To avoid the fundamental limitations of OAD-based On-TAL approaches, we build the online-applicable anchor-based TAL framework using the sliding window scheme, which can significantly boost the performance and detect actions earlier than the action ends.
- We propose an online-oriented post-processing method named Online Suppression Network (OSN), which can approximate non-maximum suppression (NMS) without violating the online constraints.
- We introduce a new online-oriented metric for On-TAL named Average Early Detected Time (AEDT), which can indicate the responsiveness of the online video understanding task.
- Extensive experiments on THUMOS'14, MUSES, and BBDB show the effectiveness of our model for On-TAL, and comparison with offline TAL methods supports the potential of the sliding window scheme.

2 Related Work

Temporal action localization Temporal action localization (TAL) is the instance-level detection of actions that finds the action boundaries and its class. Early works [23,7] approach this task by using the sliding window scheme. As sliding window approaches only consider local parts of a video for the detection, most recent works have focused on widening the coverage of the video context by using graph networks [30,28,31], temporal multi-scale networks [33,18], relations among the action proposals [3] or concatenating the global context [34]. For another branch of TAL approach, end-to-end training methods [27,15] are recently getting attention. Forced by the constraint of the input format (On-TAL), we draw attention to the sliding window approach again and narrow the performance gap to the approaches that use coarse to fine video contexts.

Online video understanding As the start of the online video understanding, De Geest *et al.* [4] proposed the Online Action Detection (OAD) task which classifies per-frame action classes under the streaming video setting. Various approaches are studied for the OAD task, such as applying reinforcement learning [8], anticipating intermediate future [29,13], and building a new GRU-based neural network layer [5]. Recently, Wang *et al.* [26] designed a transformer-based model named OadTR which encodes past frames and decodes future frames to boost the performance of the OAD.

Aside from classifying action classes, Shou *et al.* [22] suggested Online Detection of Action Start (ODAS) that focuses on the action start. The goal of ODAS is detecting the action start frame as early as possible in streaming videos. The following work [9] improves the performance of ODAS by combining the conventional per-frame action classification and the class-agnostic start detection that is trained by reinforcement learning.

As another type of online video understanding, Online Temporal Action Localization (On-TAL) focuses on instance-level detection of actions. Kang *et al.* [12] firstly defined the On-TAL task, which takes a streaming video as an input and generates instance-level action proposals similar to TAL outputs. They grouped the predictions from a backbone OAD model with a method of context-aware actionness grouping (CAG). CAG is formalized into a Markov Decision Process (MDP), and trained by the imitation learning.

However, using OAD backbone causes inevitable limitations such as tick, fragmentation and merging. Our work excludes OAD backbones from the framework, and relieves such limitations by introducing the anchor-based framework. Although the anchor-based framework cannot be evaluated on frame-level detection (*e.g.* ODAS or OAD) as done in [12], our framework significantly boosts the On-TAL performance that are comparable to offline methods.

Post-processing of generated proposals The localization tasks, such as object detection and temporal action localization, use a proposal generation framework and typically employ the non-maximum suppression (NMS) method as its post-processing to reduce repetitive proposals. As a deep learning approach of NMS,

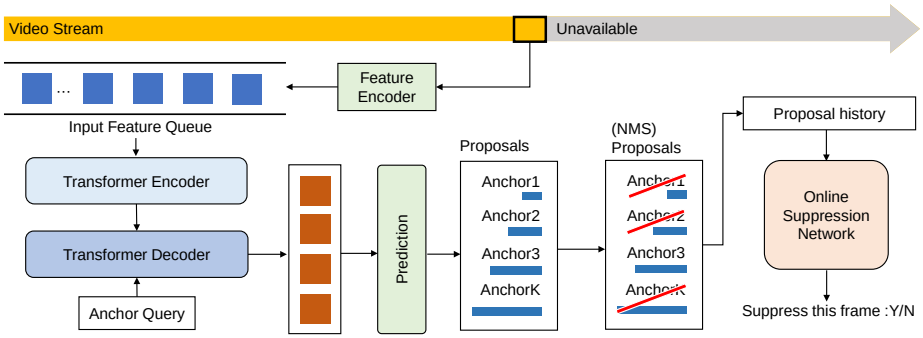

Fig. 2: Illustration of how OAT makes action proposals from a new input frame. The streaming video frames are encoded by the pre-trained feature encoder and the extracted features are gathered in the input feature queue. The transformer encoder generates queue-level features from the contents of the queue. With the anchor queries and the encoded features, the transformer decoder makes the anchor segment features of various anchor sizes. Each feature is fed into the prediction module that classifies the action and regresses the offset and the length for the action boundaries. To suppress repetitive proposals, we use the online post-processing method that consists of per-frame NMS and online suppression network.

Hosang *et al.* [10] suggested context-aware NMS methods for the purpose of removing threshold hyperparameters. In this paper, we propose Online Suppression Network (OSN) to enable online post-processing that does not use future context of videos.

3 Proposed Method

3.1 Problem Definition

Let $V = \{v_i\}_{i=1}^{T}$ and $\Psi = \{\psi_m\}_{m=1}^{M} = \{(s_m, e_m, c_m)\}_{m=1}^{M}$ denote an untrimmed video with T frames and its corresponding M label sets that have start time s_m, end time e_m and action category c_m. By the online constraint of On-TAL, we can observe the partial video $V_{1:t} = \{v_i\}_{i=1}^{t}$ at timestamp $t(1 \leq t \leq T)$. Given $V_{1:t}$, our goal is to generate action proposals as soon as action ends are detected, and recover Ψ in consecutive order. Once action proposals are generated, they are not allowed to be repaired or removed under the online constraint, so NMS-like post-processing cannot be applied to this task.

3.2 Architecture Overview

As depicted in Fig. 2, OAT consists of 4 main components: (1) transformer encoder, (2) transformer decoder, (3) prediction module, and (4) online post-processing. To have compact processing of incoming video frames, we convert

k consecutive raw video frames to a video feature vector by the pre-trained feature encoder. The feature vectors are serially gathered in the input feature queue of length L_q. If there exist empty spaces in the queue, they are filled with zeros. The transformer encoder encodes the feature sequence of the input queue and makes queue-level representations. The transformer decoder receives anchor queries as input along with queue-level representations, and produces decoded representations of length K that represent pre-defined anchor segments in the queue. Each representation is fed into the prediction module which consists of the classification head and the boundary regression head. The classification head classifies C categories plus one background class, and the boundary regression head refines the anchor segment boundaries only when the classification result of the anchor segment is not the background class. As the post-processing method, NMS is applied to K action proposals from OAT and its results are enqueued to the proposal history. By looking at the proposal history, the online suppression network (OSN) decides to use the current frame as the final output or suppress the frame.

3.3 Transformer Encoder

Each video feature in the input feature queue is a locally encoded representation that is extracted by the pre-trained feature encoder. We convert the local features into the temporally contextual features by using a standard transformer encoder as described in [24].

Given feature vectors from the input feature queue at timestamp t, the feature sequence is fed into a linear projection layer and projected into D-dimensional feature space. We denote this as $\mathbf{X} = \{\mathbf{x}_i\}_{i=t-L_q+1}^{t} \in \mathbb{R}^{L_q \times D}$. To provide positional information, we apply the sinusoidal positional encoding PE,

$$\mathbf{p}_i = \mathbf{x}_i + PE_i. \tag{1}$$

Then, $\mathbf{P} = \{\mathbf{p}_i\}_{i=1}^{L_q}$ is encoded by transformer encoder.

The transformer encoder is a sequence of N_e blocks that are consist of multi-head self-attention and feed forward network. One block of transformer encoder f operates as:

$$f(\mathbf{X}) = \mathrm{FFN}(\mathbf{X}') + \mathbf{X}', \tag{2}$$

$$\mathbf{X}' = \mathrm{MultiAtt}(\mathbf{X}) + \mathbf{X}, \tag{3}$$

where FFN is a feed forward network, and MultiAtt is a multi-head self-attention layer. As our model have N_e blocks of encoder, it outputs $\mathbf{E} = f_{N_e} \circ \cdots \circ f_1(\mathbf{P}) \in \mathbb{R}^{L_q \times D}$.

3.4 Transformer Decoder

For every new input video frame, our model generates K action proposals that are learned by pre-defined anchors of the ground truth. Previously, the anchor

segments of different length are physically cropped and reshaped into the same feature shape by various pooling methods. Our model integrates those complex steps, and generates anchor representations directly. The visualization of how the decoder is trained is reported in the supplementary material.

From the representation sequence $\mathbf{E}$ of length L_q from the encoder, we transform it to K-length representation sequence by using a standard transformer decoder. The transformer decoder has two types of inputs. One is a learnable anchor query $\mathbf{A} \in \mathbb{R}^{K \times D}$, and the other is the encoded queue-level context $\mathbf{E}$. Passing through the N_d blocks of multi-head self-attention (Eq. 3) and FFN (Eq. 2), the decoder outputs $\mathbf{D} \in \mathbb{R}^{K \times D}$.

3.5 Prediction Module and Loss Function

The decoded anchor representation $\mathbf{D}$ is fed into the prediction module that includes an action classification head and a boundary regression head. The classification head classifies the action category of the anchor segment including the background class. The boundary regression head predicts the distance between the target action end and the anchor segment's end (offset), and the ratio between the target action length and the anchor segment's length (length). Each head is a two-layered feed forward network and outputs $logit_c \in \mathbb{R}^{K \times (C+1)}$ and $\{logit_o, logit_l\} \in \mathbb{R}^{K \times 2}$, respectively.

During the training, we give the supervision of pre-defined anchors to the output of our model. We fix the end point of the anchors at the latest input frame, and define K anchors of various length (*e.g.* $\{L_q/8, L_q/4, L_q/2, L_q\}$). Then, we check the ground truth instances overlapping with each anchor and calculate Intersection-over-Union (IoU) between the ground truths and corresponding anchors. If IoU is higher than matching threshold θ_m, the anchor is matched to the ground truth and the loss $\mathcal{L}$ is calculated as follows:

$$\mathcal{L}_c = \sum_{i=1}^{K} \mathrm{CE}(\mathrm{softmax}(logit_{c,i}), g_{c,i}), \tag{4}$$

$$\mathcal{L}_o = \sum_{i=1}^{K} \mathrm{L1}(logit_{o,i} - \frac{g_{o,i} - a_{o,i}}{a_{l,i}}), \tag{5}$$

$$\mathcal{L}_l = \sum_{i=1}^{K} \mathrm{L1}(logit_{l,i} - \log\frac{g_{l,i}}{a_{l,i}}), \tag{6}$$

$$\mathcal{L} = \mathcal{L}_c + \mathcal{L}_o + \mathcal{L}_l, \tag{7}$$

where CE is the cross entropy loss, L1 is the L1 loss, g_c, g_o, g_l, a_o, a_l are the ground truth class, the ground truth action end, the ground truth action length, the anchor segment end, and the anchor segment length, respectively. Otherwise, the anchor is set to the background class and the boundary regression is omitted.

By fixing the end point of the anchors at the latest input frame, the proposals with the background class are generated at the early part of actions, and the

anchors are matched to their ground truth in the middle and the end of the actions, which have enough context to predict precise action boundaries.

During the inference, our model makes K action proposals $\{(s_i, e_i, c_i)\}_{i=1}^{K}$ for every timestamp as follows:

$$c_i = \text{softmax}(logit_{c,i}), \tag{8}$$

$$e_i = a_{o,i} + a_{l,i} logit_{o,i}, \tag{9}$$

$$s_i = e_i - a_{l,i} \exp(logit_{l,i}). \tag{10}$$

3.6 Online Post-processing

As OAT generates action proposals for every new input, suppressing the repetitive proposals is critical to the detection performance. Conventional NMS that are used in offline TAL methods cannot be applied to our framework due to the online constraint. Without the access to the future information, the On-TAL model has to pick the best proposal from the repetitive proposals in the video time axis. Therefore, we apply a neural network that determines the best time to register to the final instance set Ψ, which is approximately similar frames as the frames selected by NMS.

The proposed OAT model generates K action proposals for every new input frames. Then, we remove the proposals with the background class and apply NMS among the outputs of the single frame. The confidence scores of the action proposals are enqueued to the proposal history queue $q_h \in [0,1]^{L_q \times C}$, which has the same length as the window size L_q and can store the confidence scores by C classes separately. q_h is fed into the neural network (FC-ReLU-FC-Sigmoid), and the network outputs the NMS selection probability $s \in [0,1]^C$. If s is a value larger than the suppression threshold θ_s, the generated proposal of the current frame is registered to the final action instance set Ψ. Once registered to the final set, all following proposal candidates with high IoU are ignored. The step-by-step details are described in the supplementary material.

For the training, we made input tables ($[0,1]^{L_v \times L_q \times C}$ where L_v is video length) from the confidence scores of OAT outputs, and made ground truth labels ($\{0,1\}^{L_v \times C}$) from the NMS-applied results of OAT. If the proposal that generated on timestamp t is selected by NMS, the frames of timestamp $t-1, t, t+1$ are set to 1, otherwise the frames are set to 0. OSN is trained by the binary cross entropy loss.

By this method, our framework can approximate NMS only with the past context that does not violate the online constraint. Note that our model can also operate in the offline setting by switching to the post-processing method that stacks all output proposals of all timestamp, and adopting NMS.

4 Experiments

4.1 Datasets

Following Kang *et al.* [12], we use THUMOS'14 [11] to evaluate our model. They also cited ActivityNet v1.3 [6], but mentioned that it is not suitable for On-TAL

since the main objective of On-TAL is to detect multiple action instances in the streaming setting. Therefore, we find two more datasets with three conditions. The dataset should have boundary labels, multiple instances per video, and multiple classes per video. We use MUSES [19] and BBDB [20], which satisfy all conditions for evaluating our model.

THUMOS'14 [11] contains 200 training videos and 213 test videos with 20 action classes. The videos are sport-related untrimmed videos and have an average of 15 action instances per video.

MUSES [19] contains 3,697 videos, a total of 716 hours with 25 action classes. The videos are from TV and movie dramas, and have an average of 8.5 action instances and 3.3 action classes per video. The action instances are multi-shot event, which makes the dataset challenging.

BBDB [20] contains 1,172 full baseball games, adding up to 4,254 hours of videos with 30 classes. Since a single video is a full game, it has almost all classes in one video. Also, it has an average of 405 action instances per video. The lengths of videos are extremely long (over 2 hours), so the methods of processing whole video at once are difficult to adopt to this dataset.

4.2 Evaluation Metric

For an easy comparison to existing On-TAL and offline TAL methods, we aggregate all generated action proposals after processing all frames in videos and use mean Average Precision (mAP) with different IoU thresholds. We chose IoU threshold values in $\{0.3, 0.4, 0.5, 0.6, 0.7\}$ to report.

Since mAP metric is calculated once after all frames are processed, it cannot evaluate how reactive the model is. As the secondary evaluation metric, we propose a new online-oriented metric, named Average Early Detected Time (AEDT).

Assume that we have the true positive action proposal set $\{(p_{s,i}, p_{e,i}, p_{g,i}, c_i)\}_{i=1}^{N_{tp}}$, where p_s, p_e, p_g, c, N_{tp} are the action start time, the action end time, the proposal generation time, the action class, and the number of true positives. Also, we have its corresponding ground truth set $\{(g_{s,i}, g_{e,i}, c_i)\}_{i=1}^{N_{tp}}$, where g_s, g_e are the start time and the end time of ground truth actions. To calculate how early the action proposal is generated, we subtract the ground truth action end time from the generated time of the action proposal. Formally, AEDT is defined as follows:

$$AEDT = \frac{1}{N_{tp}} \sum_{i=1}^{N_{tp}} (p_{g,i} - g_{e,i}). \tag{11}$$

4.3 Implementation Detail

As the video feature encoder for THUMOS'14, we use the two-stream TSN [25] trained on Kinetics [1]. On MUSES, we use the I3D [1] feature extractor following

the conventions of previous MUSES work [19]. On BBDB, we use the RGB-stream of TSN (ResNet-50) after setting the videos into 6 fps.

For our network, we use $D = 1024$ feature dimensions for the transformer, 3 encoder blocks with 8 heads, and 5 decoder blocks with 4 heads. The threshold for matching the ground truth with the proposal (θ_m) is set to 0.5, the suppression network threshold (θ_s) is set to 0.1, and overlapping threshold (θ_o) is set to 0.3. For dataset specific parameters, we use the input queue size $L_q = 64$ and $K = 6$ anchors of size $\{4, 8, 16, 32, 48, 64\}$ for THUMOS'14, $L_q = 150$ and $K = 7$ anchors of $\{4, 9, 18, 37, 75, 112, 150\}$ on MUSES, and $L_q = 32$ and $K = 4$ anchors of $\{4, 8, 16, 32\}$ on BBDB. For training of the main network, we use Adam [14] optimizer with the learning rate of 0.0001, the weight decay of 0.0001, and the batch size of 128. The suppression network is separately trained with Adam optimizer with the learning rate of 0.0005, the weight decay of 0.0001, and the batch size of 128.

For the online execution, the per-frame process must be finished until the next frame comes. If a model executes faster than the frame sampling rate (*e.g.* 5 fps on THUMOS'14, 0.75 fps on MUSES, 1 fps on BBDB), it is applicable to online execution. The execution speed of our model is 70.5 fps on Nvidia RTX 2080 Ti GPU, which can be executed online.

4.4 Online Performance Comparison

To evaluate on the On-TAL task, we compare our model with the previous work (CAG-QIL) and the post-processing baseline (OAT-Naive). We set the post-processing method of OAT-Naive as registering the first proposal of the higher confidence score than the threshold to the final output set and ignore all following overlapping proposals.

As shown in Table 1, OAT performs significantly better than CAG-QIL, due to the advantages of the anchor-based approach. In Table 2, OAT can detect the action instance before the action ends which can be observed as minus values of AEDT. CAG-QIL, which is based on OAD grouping, shows near-zero values of AEDT, meaning that it detect action instances near the end of actions.

Comparing the post-processing methods, the naive approach shows the fastest detection of actions. However, the first detected proposal may not be accurate, so the mAP of the naive approach have inferior performance compared to OSN. On the other hand, OSN waits for the better proposal by approximating NMS, so it shows higher mAP and slower AEDT than the naive approach.

4.5 Offline Performance Comparison

As On-TAL is a newly proposed task, only a limited amount of works are available for the comparison. Therefore, we also compare our work with offline TAL methods in Table 3 and Table 4. Although our work succeeded to narrow the performance gap between online and offline methods, there is still a performance

Dataset	Method	0.3	0.4	0.5	0.6	0.7
THUMOS'14	CAG-QIL [12]	48.4	40.8	33.0	24.2	16.2
	OAT-Naive	57.6	50.6	43.0	30.0	15.7
	OAT-OSN	**63.0**	**56.7**	**47.1**	**36.3**	**20.0**
MUSES	CAG-QIL [12]	8.5	6.5	4.2	2.8	1.9
	OAT-Naive	20.3	16.6	12.9	7.7	3.6
	OAT-OSN	**22.1**	**18.5**	**14.2**	**8.9**	**4.7**
BBDB	CAG-QIL [12]	36.2	35.3	32.9	28.5	21.8
	OAT-Naive	52.4	52.1	49.5	44.2	37.8
	OAT-OSN	**64.4**	**64.2**	**63.4**	**60.4**	**53.4**

Table 1: Comparison of mAP (%) at different tIoU thresholds with On-TAL baseline (CAG-QIL) on 3 datasets.

Dataset	Method	0.3	0.4	0.5	0.6	0.7
THUMOS'14	CAG-QIL [12]	-0.11	-0.07	-0.08	-0.07	-0.04
	OAT-Naive	**-1.65**	**-1.66**	**-1.68**	**-1.73**	**-1.75**
	OAT-OSN	-1.28	-1.25	-1.25	-1.23	-1.21
MUSES	CAG-QIL [12]	-1.03	-0.69	-0.77	-0.79	-0.54
	OAT-Naive	**-18.57**	**-16.73**	-13.31	-11.05	**-9.80**
	OAT-OSN	-17.29	-16.08	**-13.99**	**-11.12**	-9.53
BBDB	CAG-QIL [12]	-0.02	0.03	0.10	0.17	0.21
	OAT-Naive	**-2.78**	**-2.78**	**-2.77**	**-2.75**	**-2.77**
	OAT-OSN	-0.15	-0.16	-0.16	-0.19	-0.25

Table 2: Comparison of AEDT (second) at different tIoU thresholds with On-TAL baseline (CAG-QIL) on 3 datasets.

gap to the state-of-the-art offline TAL methods. However, applying the video-level post-processing (OAT-NMS) boosts the performance and shows comparable performance to the state-of-the-art offline methods. For the post-processing method, we stack all generated action proposals in a video and apply NMS to make final action proposals. Note that other factors, including the network design and hyperparameters, are the same except for the post-processing. Our model is not applicable to global or multi-scale contexts due to the processing constraints of online inputs, but the comparable performance to offline methods with the minor post-processing change shows that our model produces high quality action proposals as offline methods. This indicates that the performance gap between the online and the offline method may be minimized depending on the improvement of the controlling algorithm for repetitive proposals.

4.6 Qualitative Evaluation

As the qualitative result, we compare the output proposal sets of CAG-QIL and OAT-OSN in Fig. 3. For the OAD-based models like CAG-QIL, the per-frame predictions of OAD raise fundamental difficulties to group them into accurate

Method	0.3	0.4	0.5	0.6	0.7
CDC [21]	40.1	29.4	23.3	13.1	7.9
BSN [17]	53.5	45.0	36.9	28.4	20.0
TAL-Net [2]	53.2	48.5	42.8	33.8	20.8
BMN [16]	56.0	47.4	38.8	29.7	20.5
G-TAD [28]	54.5	47.6	40.2	30.8	23.4
PBRNet [18]	58.5	54.6	51.3	41.8	29.5
VSGN [31]	66.7	60.4	52.4	41.0	30.4
ContextLoc [34]	68.3	63.8	54.3	41.8	26.2
MUSES [19]	68.9	**64.0**	**56.9**	**46.3**	**31.0**
OAT-OSN	63.0	56.7	47.1	36.3	20.0
OAT-NMS	**69.7**	**64.0**	53.9	42.9	27.0

Table 3: Comparison of mAP (%) at different tIoU thresholds with *offline* TAL methods on THUMOS'14.

Dataset	Method	0.3	0.4	0.5	0.6	0.7
MUSES	MR [32]	12.9	11.3	9.2	7.6	5.9
	G-TAD [28]	19.1	14.8	11.1	7.4	4.7
	P-GCN [30]	19.9	17.1	13.1	9.7	5.4
	MUSES [19]	25.9	22.6	18.9	**15.0**	**10.6**
	OAT-OSN	22.1	18.5	14.2	8.9	4.7
	OAT-NMS	**27.7**	**24.3**	**19.9**	14.9	9.2
BBDB	Single frame [20]	10.0	7.9	3.4	2.5	1.6
	CDC [21]	26.1	22.2	11.3	9.5	6.1
	OAT-OSN	64.4	64.2	63.4	60.4	53.4
	OAT-NMS	**66.6**	**66.5**	**65.8**	**63.5**	**56.7**

Table 4: Comparison of mAP (%) at different tIoU thresholds with *offline* TAL methods on MUSES and BBDB.

action instances. In the case of the high confidence score on the non-action frames, CAG-QIL makes short action proposals that is not related to the ground truth actions (Fig. 3 (a)). In the opposite case of the low confidence score on the action frames, it makes one ground truth action into several fragmented action proposals (Fig. 3 (b)). OAD-base models also shows limitations when the actions are overlapping or repeat with short intervals (Fig 3 (c)), as CAG-QIL shows the merged proposals on consecutive actions. On the other hand, OAT uses instance-level context from anchors and it shows the advantages of overcoming the problems of tick, fragmentation, and merge.

5 Model Analysis

Rethinking steps of sliding window scheme To share our observations of rethinking sliding window scheme, we show step-by-step results by changing the

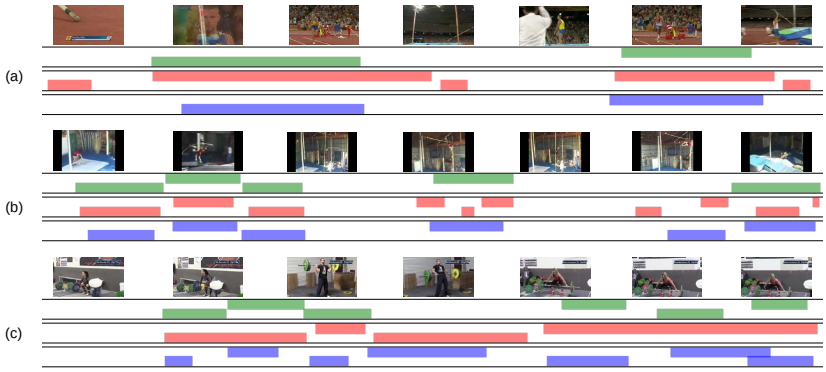

Fig. 3: Comparison on qualitative results of OAT-ONS with CAG-QIL [12]. Green is the ground truth, red is the result of CAG-QIL, and blue is the result of OAT-ONS. The rows are the examples of tick (a), fragmentation (b), and merge (c), respectively.

Method	0.3	0.4	0.5	0.6	0.7
Pooling-NMS	61.0	53.4	42.7	30.3	16.6
ConvNet-NMS	63.1	56.6	47.4	32.2	18.9
OAT-NMS	**69.7**	**64.0**	**53.9**	**42.9**	**27.0**

Table 5: Comparison of mAP (%) with step-by-step changes of the anchor encoding components on THUMOS'14.

component from the baseline method in Table 5. All methods in this experiment shares the sliding window framework and the same post-processing for the fair comparison.

Firstly, we set the baseline by changing the transformers into the pooling layers for the anchor encoding (Pooling-NMS). As a straightforward extension of TURN [7], Pooling-NMS divides each anchor segment into three parts, and the features of each part are average-pooled, and the three features are concatenated for the anchor encodings. It shows the performance of 42.7% @tIoU=0.5

For the next step, we try to use the convolutional networks for the anchor encoding (ConvNet-NMS). ConvNet-NMS has 2-layer convnets separately for each anchors, and the output shape of each convnets are the same. The performance gain of exploiting convnets is +4.7% @tIoU=0.5, mainly achieved by the richer context than the average pooling.

Our method, OAT, shows +6.5% @tIoU=0.5 gain from ConvNet-NMS. The sliding window includes background frames, so the transformer encoder makes rich representations including non-action contexts and its anchor decodings can make accurate action proposals. As the byproduct of finding the effective On-TAL framework, we found the sliding window, which is considered to be outdated

	0.3	0.4	0.5	0.6	0.7	Avg.	Props.
0.1	56.4	48.6	39.6	26.0	11.5	36.4	5247
0.2	57.2	49.5	40.7	28.2	13.6	37.8	4452
0.3	**57.8**	**50.7**	42.3	29.0	14.4	38.8	3991
0.4	57.6	50.6	**43.0**	30.0	15.7	**39.4**	3610
0.5	56.5	49.9	41.9	**30.4**	**16.6**	39.1	3269
0.6	53.2	47.0	39.8	29.6	15.8	37.1	2911

Table 6: mAP (%) with different thresholds (row), different tIoUs (column), the average of columns (Avg.), and the number of proposals (Props.) of OAT-Naive on THUMOS'14.

for the offline TAL, shows the comparable performance to state-of-the-art offline TAL methods.

Naive thresholding To strengthen the effectiveness of ONS, we show the detailed results of OAT-Naive by changing threshold values. Our final selection of the threshold is 0.4, since it shows the best performance on average, as seen in Table 6.

The smaller thresholds can catch the actions with the low confidence scores, but increase the number of false positives causing the performance decrease at high tIoU. As the thresholds get larger, the number of proposals decreases, and the proposals with high confidence remain. However, it overlooks the detected actions with low confidence, causing the performance drop at low tIoU.

The simple change of thresholds cannot reach to the performance of NMS and its approximation, ONS. Those methods select the maximum confidence, which is a relative operation, and are able to suppress repetitive proposals of both low and high confidence.

6 Conclusion

In this paper, we proposed the anchor-based action localization model, named Online Anchor Transformer (OAT), to deal with Online Temporal Action Localization. In addition to OAT, we also proposed the Online Suppression Network which is an online-applicable post-processing method. Our model shows significantly better performance than the baseline in terms of both mAP and online responsiveness. By changing the post-processing method, our model performs comparably to state-of-the-art offline TAL methods, making inspirations for rethinking the sliding window scheme.

Acknowledgements. This work has partly supported by the National Research Foundation of Korea(NRF) grant funded by the Korea government(MSIT)(NRF-2022R1A2C2004509) and by Institute of Information communications Technology Planning Evaluation (IITP) grant funded by the Korea government(MSIT), Artificial Intelligence Innovation Hub under Grant 2021-0-02068, Artificial Intelligence Graduate School Program under Grant 2020-0-01361.

References

1. Carreira, J., Zisserman, A.: Quo vadis, action recognition? a new model and the kinetics dataset. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 6299–6308 (2017)
2. Chao, Y.W., Vijayanarasimhan, S., Seybold, B., Ross, D.A., Deng, J., Sukthankar, R.: Rethinking the faster r-cnn architecture for temporal action localization. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 1130–1139 (2018)
3. Chen, P., Gan, C., Shen, G., Huang, W., Zeng, R., Tan, M.: Relation attention for temporal action localization. IEEE Transactions on Multimedia **22**(10), 2723–2733 (2019)
4. De Geest, R., Gavves, E., Ghodrati, A., Li, Z., Snoek, C., Tuytelaars, T.: Online action detection. In: Proceedings of the European Conference on Computer Vision (ECCV). pp. 269–284 (2016)
5. Eun, H., Moon, J., Park, J., Jung, C., Kim, C.: Learning to discriminate information for online action detection. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 809–818 (2020)
6. Fabian Caba Heilbron, Victor Escorcia, B.G., Niebles, J.C.: Activitynet: A large-scale video benchmark for human activity understanding. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 961–970 (2015)
7. Gao, J., Yang, Z., Chen, K., Sun, C., Nevatia, R.: Turn tap: Temporal unit regression network for temporal action proposals. In: Proceedings of the IEEE International Conference on Computer Vision (ICCV). pp. 3628–3636 (2017)
8. Gao, J., Yang, Z., Nevatia, R.: RED: Reinforced encoder-decoder networks for action anticipation. In: Proceedings of the British Machine Vision Conference (BMVC). pp. 92.1–92.11 (September 2017)
9. Gao, M., Xu, M., Davis, L.S., Socher, R., Xiong, C.: Startnet: Online detection of action start in untrimmed videos. In: Proceedings of the IEEE International Conference on Computer Vision (ICCV). pp. 5542–5551 (2019)
10. Hosang, J., Benenson, R., Schiele, B.: Learning non-maximum suppression. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 4507–4515 (2017)
11. Jiang, Y.G., Liu, J., Roshan Zamir, A., Toderici, G., Laptev, I., Shah, M., Sukthankar, R.: THUMOS challenge: Action recognition with a large number of classes. `http://crcv.ucf.edu/THUMOS14/` (2014)
12. Kang, H., Kim, K., Ko, Y., Kim, S.J.: Cag-qil: Context-aware actionness grouping via q imitation learning for online temporal action localization. In: Proceedings of the IEEE International Conference on Computer Vision (ICCV). pp. 13729–13738 (2021)
13. Kim, Y.H., Nam, S., Kim, S.J.: Temporally smooth online action detection using cycle-consistent future anticipation. Pattern Recognition **116**, 107954 (2021)
14. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
15. Lin, C., Xu, C., Luo, D., Wang, Y., Tai, Y., Wang, C., Li, J., Huang, F., Fu, Y.: Learning salient boundary feature for anchor-free temporal action localization. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 3320–3329 (2021)

16. Lin, T., Liu, X., Li, X., Ding, E., Wen, S.: Bmn: Boundary-matching network for temporal action proposal generation. In: Proceedings of the IEEE International Conference on Computer Vision (ICCV). pp. 3889–3898 (2019)
17. Lin, T., Zhao, X., Su, H., Wang, C., Yang, M.: Bsn: Boundary sensitive network for temporal action proposal generation. In: Proceedings of the European Conference on Computer Vision (ECCV). pp. 3–19 (2018)
18. Liu, Q., Wang, Z.: Progressive boundary refinement network for temporal action detection. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 34, pp. 11612–11619 (2020)
19. Liu, X., Hu, Y., Bai, S., Ding, F., Bai, X., Torr, P.H.: Multi-shot temporal event localization: a benchmark. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 12596–12606 (2021)
20. Shim, M., Kim, Y.H., Kim, K., Kim, S.J.: Teaching machines to understand baseball games: Large-scale baseball video database for multiple video understanding tasks. In: Proceedings of the European Conference on Computer Vision (ECCV). pp. 404–420 (2018)
21. Shou, Z., Chan, J., Zareian, A., Miyazawa, K., Chang, S.F.: Cdc: Convolutional-de-convolutional networks for precise temporal action localization in untrimmed videos. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 5734–5743 (2017)
22. Shou, Z., Pan, J., Chan, J., Miyazawa, K., Mansour, H., Vetro, A., Giro-i Nieto, X., Chang, S.F.: Online detection of action start in untrimmed, streaming videos. In: Proceedings of the European Conference on Computer Vision (ECCV). pp. 534–551 (2018)
23. Shou, Z., Wang, D., Chang, S.F.: Temporal action localization in untrimmed videos via multi-stage cnns. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 1049–1058 (2016)
24. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. Proceedings of the Advances in Neural Information Processing Systems (NeurIPS) **30** (2017)
25. Wang, L., Xiong, Y., Wang, Z., Qiao, Y., Lin, D., Tang, X., Van Gool, L.: Temporal segment networks: Towards good practices for deep action recognition. In: Proceedings of the European Conference on Computer Vision (ECCV). pp. 20–36 (2016)
26. Wang, X., Zhang, S., Qing, Z., Shao, Y., Zuo, Z., Gao, C., Sang, N.: Oadtr: Online action detection with transformers. In: Proceedings of the IEEE International Conference on Computer Vision (ICCV). pp. 7565–7575 (2021)
27. Xu, M., Perez Rua, J.M., Zhu, X., Ghanem, B., Martinez, B.: Low-fidelity video encoder optimization for temporal action localization. Proceedings of the Advances in Neural Information Processing Systems (NeurIPS) **34** (2021)
28. Xu, M., Zhao, C., Rojas, D.S., Thabet, A., Ghanem, B.: G-tad: Sub-graph localization for temporal action detection. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 10156–10165 (2020)
29. Xu, M., Gao, M., Chen, Y.T., Davis, L.S., Crandall, D.J.: Temporal recurrent networks for online action detection. In: Proceedings of the IEEE International Conference on Computer Vision (ICCV). pp. 5532–5541 (2019)
30. Zeng, R., Huang, W., Tan, M., Rong, Y., Zhao, P., Huang, J., Gan, C.: Graph convolutional networks for temporal action localization. In: Proceedings of the IEEE International Conference on Computer Vision (ICCV). pp. 7094–7103 (2019)

31. Zhao, C., Thabet, A.K., Ghanem, B.: Video self-stitching graph network for temporal action localization. In: Proceedings of the IEEE International Conference on Computer Vision (ICCV). pp. 13658–13667 (2021)
32. Zhao, P., Xie, L., Ju, C., Zhang, Y., Wang, Y., Tian, Q.: Bottom-up temporal action localization with mutual regularization. In: Proceedings of the European Conference on Computer Vision (ECCV). pp. 539–555 (2020)
33. Zhao, Y., Xiong, Y., Wang, L., Wu, Z., Tang, X., Lin, D.: Temporal action detection with structured segment networks. In: Proceedings of the IEEE International Conference on Computer Vision (ICCV). pp. 2914–2923 (2017)
34. Zhu, Z., Tang, W., Wang, L., Zheng, N., Hua, G.: Enriching local and global contexts for temporal action localization. In: Proceedings of the IEEE International Conference on Computer Vision (ICCV). pp. 13516–13525 (2021)