Online Temporal Action Localization with Memory-Augmented Transformer

Youngkil Song* Dongkeun Kim* Minsu Cho Suha Kwak

Pohang University of Science and Technology (POSTECH), South Korea
{songyk,kdk1563,mscho,suha.kwak}@postech.ac.kr
https://cvlab.postech.ac.kr/research/MATR/

Abstract. Online temporal action localization (On-TAL) is the task of identifying multiple action instances given a streaming video. Since existing methods take as input only a video segment of fixed size per iteration, they are limited in considering long-term context and require tuning the segment size carefully. To overcome these limitations, we propose memory-augmented transformer (MATR). MATR utilizes the memory queue that selectively preserves the past segment features, allowing to leverage long-term context for inference. We also propose a novel action localization method that observes the current input segment to predict the end time of the ongoing action and accesses the memory queue to estimate the start time of the action. Our method outperformed existing methods on two datasets, THUMOS14 and MUSES, surpassing not only TAL methods in the online setting but also some offline TAL methods.

Keywords: Temporal action localization · Online video understanding

1 Introduction

Today, video is one of the most popular types of media, with tons of videos being created and uploaded to online platforms like YouTube and TikTok every second. As a result, the demand for video understanding is growing rapidly. In particular, dealing with untrimmed videos is a key to practical video understanding since most videos in the real world are not trimmed for individual events in advance. Temporal action localization (TAL) has been extensively studied in this context [10, 17, 21, 30, 38, 39, 43–45, 47, 49], with the aim of detecting each action instance in the input untrimmed video by predicting its start time, end time, and action class.

Recently, online TAL (On-TAL), *i.e.*, TAL for streaming video [14], has gained increasing attention thanks to its potential applications in video surveillance, sports video analysis, and video summarization. Unlike TAL which demands the entire video as input, On-TAL aims to detect action instances using only input frames observed so far in an online manner. Moreover, once action instances are predicted, it is not allowed to modify the results afterward.

* Equal contribution.

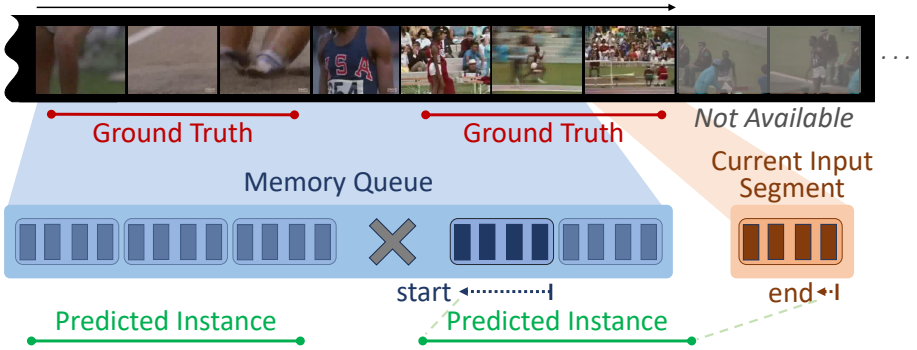

Fig. 1: A conceptual illustration of our model. Our model selectively stores information about previously occurred actions in the memory queue from streaming video. When the end of the action is detected in the current input segment, the model retrieves information from the memory queue to locate the start of the action.

A straightforward solution to this challenging task is to first conduct online action detection (OAD) [7, 35], *i.e.*, frame-by-frame action classification for streaming video, and then build action instances by aggregating the frame-wise action predictions. In this sense, earlier methods for On-TAL [14, 31] follow this approach by applying OAD and refining per-frame predictions using past observations. However, this OAD-based approach exploits only frame-level supervision for training, which is not optimal because On-TAL aims to predict action instances in the form of time intervals. To alleviate this issue, online anchor transformer (OAT) has been recently proposed [15], which adopts sliding window approach and anchor-based method to utilize instance-level supervision. Although OAT improved performance remarkably, however, it still has several drawbacks: Since it takes a fixed-size segment of input video at each iteration and does not consider long-term contexts, its capability is limited in detecting long action instances, and its performance is sensitive to hyperparameters like the size of the input segment.

To resolve the above issues, we propose a new end-to-end architecture for On-TAL, dubbed *memory-augmented* Transformer (MATR). At the heart of MATR lies in *memory queue*, which selectively stores past segment features so that the model exploits long-term context for inference. Thanks to the memory queue that contains long-term context and reduces dependence on the segment size, our model accurately localizes even long-term action instances without the need for careful hyperparameter tuning for each dataset.

Moreover, we propose a novel approach to action instance localization using the memory queue as illustrated in Fig. 1. First, it detects the end of an action using the current segment features and then scans past segment features in the memory queue to find the action start corresponding to the detected action end. To achieve this, we adopt two Transformer decoders: one for end detection and the other for start detection. Specifically, we adopt learnable queries as input to

the decoders so that each of them learns to localize action boundaries through the attention mechanism of Transformer [8,32]. In addition, motivated by recent advances on object detection [6,37], MATR decouples action classification and localization to further improve the performance. This is realized in MATR by using distinct queries for the two subtasks.

The proposed method is evaluated on two datasets, THUMOS14 [13] and MUSES [22]. Our model achieves the state of the art on both datasets in the online TAL setting. In addition, surprisingly, it is also comparable to existing offline TAL methods although it does not employ offline non-maximum suppression (NMS). In summary, the contribution of this paper is three-fold:

- We propose MATR, a new end-to-end architecture for On-TAL, leveraging memory queue to utilize long-term context and enabling precise localization of action instances with less dependence on dataset-specific hyperparameter.
- We introduce a new action instance localization method that first identifies action end using the input segments and scans past information in the memory queue to find the action start. Moreover, our method adopts distinct queries to separate information for action classification and localization.
- Our method outperformed the existing On-TAL methods on the two benchmarks. Also, extensive ablation studies demonstrate the contribution of each component of the proposed method.

2 Related Work

Temporal Action Localization. Temporal action localization (TAL) [5,10,17,21,23,30,38,39,43–45,47,49] which identifies action instances in time and classifying their actions, has given much attention on the video research community. Earlier attempts employ two-stage approaches that first generate action proposals, then classify action class and refine action boundaries [1,9,12,18,19,24,47]. Recent studies have primarily dealt with widening the range of video context through convolutional neural networks [28], graph neural networks [39,43], Gaussian kernel [25], and utilizing global context [49]. To extend temporal context and enable end-to-end TAL, Transformer [8,32] has recently been adopted [6,23,44]. These approaches incorporate temporal deformable attention module [23] and multi-scale features [44] to improve computational efficiency and performance.

Online Video Understanding. Online action detection (OAD), which classifies action classes in a frame-wise manner, is first proposed by Geest *et al.* [7]. To incorporate long-term context, recurrent neural network [40] and Transformer [35] are utilized. These methods predict actions for future frames and use this information to refine current frame predictions. Yang et al. [42] further improve the performance by using category exemplars and capturing long-term dependencies within a video segment. On the other hand, Shou *et al.* [29] proposes online detection of action start (ODAS), which aims to detect action start as early as possible. Gao *et al.* [11] further improves ODAS by adopting reinforcement learning.

Online Temporal Action Localization. Towards practical event understanding, online temporal action localization (On-TAL) has been proposed recently. Kang *et al*. [14] first propose On-TAL, which localizes action boundaries and classifies action classes without accessing future frames and offline post-processing. Earlier methods [14,31] adopt OAD framework to classify per-frame action class and group them into action instances to solve On-TAL. Specifically, Kang *et al*. [14] utilize Markov decision process to aware prior contexts to group per-frame predictions. SimOn [31] is proposed as an end-to-end On-TAL method that leverages past visual context and action probabilities for accurate action prediction at the current frame. Kim *et al*. [15] propose a sliding window scheme and an anchor-based method that adopts Transformer [32] for On-TAL. However, it only uses a fixed input segment which is highly dependent on the action length distribution of each dataset. Unlike the previous methods, we utilize a memory queue for long-term context, which also reduces the dependence on the input segment size.

Difference between OAD and On-TAL. OAD [2,33,35,40–42,46] focus on predicting action classes at the frame-level, which limits the ability to distinguish boundaries of overlapping actions. On the other hand, On-TAL [14,15,31] predicts entire action instances rather than individual frames, enabling it to precisely identify the start and end times of actions, making it effective for applications requiring instance-level action understanding, like sports video analysis.

Using Memory for Video Understanding. Prior work in video understanding employs memory to store contextual information, in the forms of long-term and short-term memory [2,33,36,41]: short-term memory captures fine details, and long-term memory stores compressed information of past contexts. Specifically, Wang *et al*. [33] divides long-term memory into several groups and compresses each group into a single vector. Stream Buffer [2] gradually reduces temporal dimensions of the memory using a compression module. However, unlike video classification tasks such as action recognition [36] and OAD [2,33,41,46], which requires to store representative information of past frames in the memory for classifying the action of the current frame, TAL needs to preserve the temporal positional information of past frames in the memory to accurately predict the exact time. In that sense, MATR preserves temporal information in memory to accurately localize actions and effectively stores information by using flag tokens. A comparison with the OAD memory modules is shown in Table 4.

3 Proposed Method

Consider an untrimmed video $\mathbf{V} = \{\mathbf{v}_i\}_{i=1}^{T}$ with T frames and M action instances $\Psi = \{(s_m, e_m, c_m)\}_{m=1}^{M}$, where each instance is represented by its start time s_m, end time e_m and action class c_m. Unlike temporal action localization (TAL), which allows a model to use whole video sequence as input, online temporal action localization (On-TAL) forces a model to predict action instances using only input frames seen until the current timestamp. Note that action instances predicted at previous iterations cannot be modified or deleted afterwards.

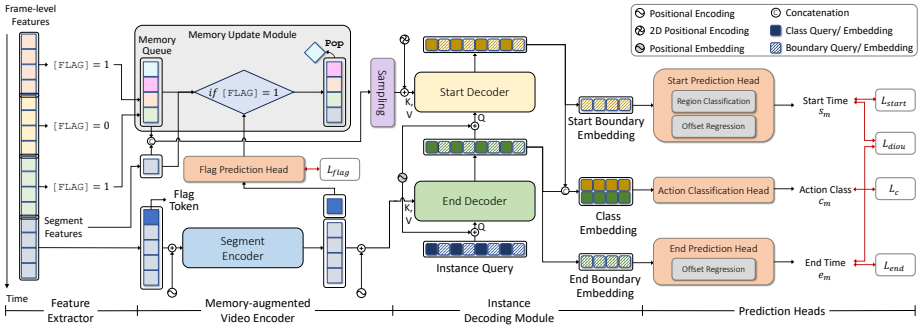

Fig. 2: Overall architecture of MATR. MATR consists of four parts: feature extractor, memory-augmented video encoder, instance decoding module, and prediction heads.

As shown in Fig. 2, MATR consists of four parts: feature extractor, memory-augmented video encoder, instance decoding module, and prediction heads. In line with previous research, the current segment of the input streaming video is given as a unit input, and its frame-level features, referred to as segment features, are extracted by a video backbone network and a linear projection layer (Sec. 3.1). The segment features are then fed to the memory-augmented video encoder, which encodes temporal context between frames in the current segment and stores the segment features into the memory (Sec. 3.2). The instance decoding module localizes action instances via two Transformer decoders: the end decoder and the start decoder. Specifically, the end decoder references the encoded segment features to locate the action end around the current time, and then the start decoder refers to the memory queue to find the action start based on the past information stored in the memory queue. Queries for each instance consist of a class query for action classification and a boundary query for action localization (Sec. 3.3). The outputs of the instance decoding module are used as inputs to the prediction heads, which consist of end prediction head, start prediction head, and action classification head (Sec. 3.4). The entire model is trained in an end-to-end manner (Sec. 3.5).

3.1 Feature Extractor

Given a segment comprising L_s consecutive video frames as input, a pretrained backbone [4, 34] followed by a linear projection layer extracts the segment features $\mathbf{X}_t = f\mathbf{x}_i g_{t-L_s+1}^{t} \; 2 \; \mathbf{R}^{L_s \times D}$ for each frame within the segment, following previous work [14, 15]. Inspired by Kim *et al.* [15], MATR employs a sliding window scheme, which predicts multiple action instances by moving frame by frame along the temporal axis. Note that the window size is the input segment size.

3.2 Memory-Augmented Video Encoder

At each time step, the input segment features are fed to two modules of the memory-augmented video encoder, the segment encoder and the memory update module. The segment encoder encodes the temporal context across input segment features, while the memory update module selectively stores the segment features and updates the memory queue.

Segment Encoder. The segment encoder is a standard Transformer encoder composed of self-attention layers and a feed forward network (FFN). The combination of segment features and a learnable flag token is fed into the encoder, and transformed to queries, keys, and values for self-attention. Sinusoidal positional encoding $\mathbf{S}_{\text{pos}} \in \mathbb{R}^{(L_s+1)\times D}$ is added to the queries and keys. Among the output embeddings, the encoded flag token is fed into the memory update module and the encoded segment features that integrate temporal context across input frames are used as input to the end decoder.

Memory Update Module. In an online setting, where an input is a streaming video, efficiently storing information of past frames and accessing it effectively is crucial for detecting action instances, in particular those of long-term actions whose temporal extents are beyond the size of input segments. To this end, we adopt the memory queue for storing information of past input segments in a first-in-first-out (FIFO) manner. Additionally, we propose an efficient memory update method employing the flag token. When the input segment features pass through the segment encoder, the flag token is concatenated into the input segment features and then employed in the flag prediction head. It is trained to predict [FLAG], which identifies whether an input segment is relevant to action instances, so that only relevant segments are stored in the memory. [FLAG] = 1 when there are overlapping frames between the input segment and action instances, and 0 otherwise. During training, the ground-truth [FLAG] is utilized. During inference, [FLAG] = 1 if sigmoid($\hat{y}$) $> \theta$ where $\hat{y}$ is the output logit of the flag prediction head and θ is a predefined threshold, and 0 otherwise. The input segment feature is added to the memory queue when [FLAG] = 1 and discards it when [FLAG] = 0. If the memory queue is full, the oldest segment feature is purged.

3.3 Instance Decoding Module

The instance decoding module localizes and classifies action instances by leveraging the encoded segment features and the memory queue through the attention mechanism of Transformer. Given the encoded segment features that have short-term temporal contexts across input segment and the memory queue that stores long-term contexts for actions that are potentially ongoing, a set of $2N$ instance queries $\mathbf{Q} = \{\mathbf{Q}_{\text{class}}; \mathbf{Q}_{\text{bound}}\} \in \mathbb{R}^{2N\times D}$ is trained to generate N action instances for each input segment. Half of these queries pertain to predicting the action class (class query; $\mathbf{Q}_{\text{class}} \in \mathbb{R}^{N\times D}$), while the other half is used for predicting the start and end timestamps (boundary query; $\mathbf{Q}_{\text{bound}} \in \mathbb{R}^{N\times D}$). The class and boundary query pairs for the same instance share the same positional embedding $\mathbf{E}_{\text{pos}} \in \mathbb{R}^{N\times D}$ to accurately identify and differentiate between

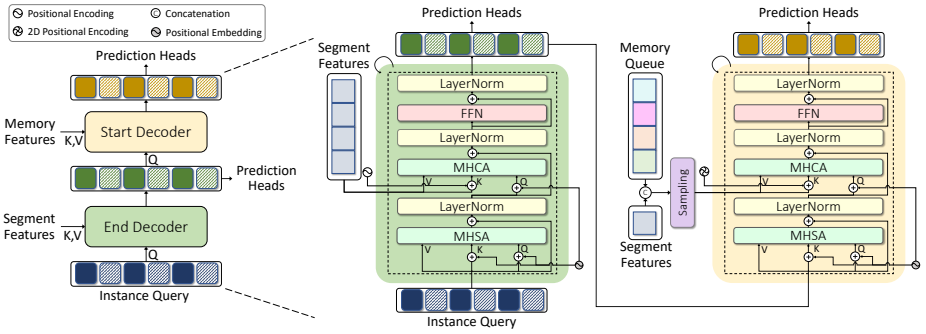

Fig. 3: Detailed architecture of the instance decoding module.

instances; $Q = f Q_{class} + E_{pos}; Q_{bound} + E_{pos} g$. Unlike the previous Transformer-based On-TAL method [15] that predicts start and end offset for each instance simultaneously, our method employs separate Transformer decoders for predicting start offset and end offset. As shown in Fig. 3, the start decoder and end decoders share the same architecture but utilizes different information to predict action start and action end, respectively.

End Decoder. Given the encoded segment features, instance queries Q is trained to generate the output embeddings used for detecting action end near the current timestamp. To this end, we adopt a Transformer [3] composed of multi-head self-attention layers, multi-head cross-attention layers, and FFNs. The detailed architecture is illustrated in Fig. 3. The outputs of the end decoder, which consist of end boundary embeddings and class embeddings, are then used as inputs to the start decoder and the prediction heads.

Start Decoder. The start decoder takes the output embeddings from the end decoder and locates the corresponding action start by utilizing the memory queue. First, the memory queue is concatenated with the current segment features. The concatenated memory features are used as long-term context in the start decoder. Then, 2D temporal positional encodings are added to the memory features, and the results are used as the *key* and *value* for the cross-attention layers of the start decoder. The overall architecture of the start decoder is the same as the end decoder (Fig. 3), but it uses the memory features rather than the encoded segment features. The output embeddings of the start decoder, start boundary embeddings and class embeddings, are used as inputs to the start prediction and the action classification head, respectively.

Moreover, we introduce two techniques to efficiently and effectively utilize the memory queue. First, we conduct a 50% uniform sampling to the memory features which enables efficient use of memory since adjacent frames contain similar information. Second, to expand the scope of positional encoding for streaming video, whose duration is unpredictable, we separate the temporal positional encoding into the two parts: relative segment position and relative frame position. A relative segment position is the position of a segment within the memory rela-

tive to the current segment. Meanwhile, a relative frame position is the position of a frame relative to the most recent frame within the same segment.

3.4 Prediction Heads

To generate N action instances $\{(\hat{s}_i, \hat{e}_i, \hat{c}_i)\}_{i=1}^{N}$ from the output of the instance decoding module, we use three prediction heads, which are end prediction head, start prediction head, and action classification head. Each of the prediction heads is composed of a 2-layer FFN.

End Prediction Head. The end prediction head estimates the offset between the end of the target action and the current timestamp. The boundary embeddings from the end decoder are fed to the 2-layer FFN, which in turn estimates the end offset $\{\hat{o}_i\}_{i=1}^{N} \in \mathbb{R}^N$.

Start Prediction Head. The start prediction head estimates the offset between the start of the target action and the current timestamp. Unlike the end prediction, which finds the end offset around the input segment, the start prediction needs to perform offset regression over a relatively wider range. To narrow down the scope of the start offset regression, we divide the time horizon into $L_m + 2$ regions: the region preceding the coverage of memory, L_m regions covered by the memory, and the region corresponding to the current input segment. In that sense, the start prediction head consists of the region classification and the offset regression, each composed of a 2-layer FFN. They take the boundary embeddings from the start decoder as input to predict the start time. The region classification identifies the regions to which the start time is assigned, represented as $\{\hat{b}_i | \hat{b}_i = \text{argmax}(\hat{r}_i)\}_{i=1}^{N} \in \mathbb{R}^N$. Here, $\{\hat{r}_i\}_{i=1}^{N} \in \mathbb{R}^{N \times (L_m+2)}$ are the output logits from the region classification head. Following this, the offset regression head is utilized to estimate the offset within the regions, denoted as $\{\hat{o}_i\}_{i=1}^{N} \in \mathbb{R}^{N \times (L_m+2)}$. Note that the start offset $\hat{o}_i$ is calculated for all $L_m + 2$ regions, and the offset from the identified region $\{\hat{o}_i(\hat{b}_i)\}_{i=1}^{N} \in \mathbb{R}^N$ is used in inference. By combining start classification result $\hat{b}_i$ and start regression result $\hat{o}_i(\hat{b}_i)$, the start time is predicted. The effectiveness of the proposed start prediction head is validated in Table 5.

Action Classification Head. For action class prediction, we utilize the class embeddings from both the end decoder and the start decoder. These embeddings are concatenated and fed into the action classification head to derive class probabilities $\{\hat{p}_i\}_{i=1}^{N} \in \mathbb{R}^{N \times (C+1)}$, where C is the number of action class.

Action Instance Prediction. Finally, our model makes N action proposals $\{(\hat{s}_i, \hat{e}_i, \hat{c}_i)\}_{i=1}^{N}$ at time t by

$$\hat{s}_i = t \quad (\hat{b}_i + \hat{o}_i(\hat{b}_i)) \quad L_s, \tag{1}$$

$$\hat{e}_i = t + \hat{a}_i \quad L_s, \tag{2}$$

$$\hat{c}_i = \text{argmax}(\hat{p}_i), \tag{3}$$

where L_s is the segment length. Since the sliding window scheme generates N action proposals at each timestamp, post-processing is crucial for performance improvement by removing redundant or overlapped action instances.

Non-maximum suppression (NMS) is applied to the action proposals at each timestamp, subsequently removing those highly overlapped with the proposals generated from the past. To prevent more reliable prediction made in the future from being removed, instances where the predicted end time is beyond the current timestamp, $t < \hat{e}_i$, are also removed.

3.5 Training Objective

At timestamp t, the model is trained to detect action instances whose end times are in the range $[t \quad T_d + 1, t + T_a]$, where T_d and T_a are hyperparameters. Then, hungarian algorithm matches the action proposals and the ground-truth action instances with the lowest matching cost. The matching cost of the ground-truth group i and the proposal $\sigma(i)$ is given by:

$$C_{i,\sigma(i)} = \hat{p}_{\sigma(i)}(y_i) + \mathrm{IoU}(b_i, \hat{b}_{\sigma(i)}), \tag{4}$$

where σ is the permutation of N action proposals, $\hat{p}_i$ is the class probabilities of the i-th proposal, $b_i = [s_i, e_i]$ is the ground-truth action boundary, and $\hat{b}_i = [\hat{s}_i, \hat{e}_i]$ is the predicted action boundary. We adopt the focal loss [20] for the action classification, the cross entropy loss for the start region classification, the ℓ_1 loss for both the start offset regression and the end offset regression. The action classification loss L_{class}, the start prediction loss L_{start}, and the end prediction loss L_{end} are defined as follow:

$$L_{\mathsf{class}} = \sum_{i=1}^{N} \mathrm{FL}(\hat{p}_i, y_i), \tag{5}$$

$$L_{\mathsf{start}} = \sum_{i=1}^{N_{\mathrm{match}}} [\mathrm{CE}(\mathrm{Softmax}(\hat{r}_i), r_i) + |\hat{v}_i - v_i|] \quad v_i > 0], \tag{6}$$

$$L_{\mathsf{end}} = \sum_{i=1}^{N_{\mathrm{match}}} |\hat{u}_i - u_i|, \tag{7}$$

where y_i, r_i, v_i, u_i is the ground-truth action class, start region, start offset, and end offset, respectively.

To provide instance-level supervision and facilitate the connection between start and end time prediction, we employ the DIoU loss [48] following ActionFormer [44]:

$$L_{\mathsf{diou}} = \sum_{i=1}^{N_{\mathrm{match}}} 1 - \mathrm{IoU}(\hat{b}_i, b_i) + \frac{\rho^2(\hat{c}_i, c_i)}{d_i^2}, \tag{8}$$

where $\rho(\ ,\ )$ is the Euclidean distance between two points, $\hat{c}_i$ and c_i are the center of the proposal and ground-truth instance, and d_i is the smallest enclosing 1-dim box length.

To train the memory update module in an end-to-end manner, we employ the flag loss for training the flag prediction head:

$$L_{\mathsf{flag}} = \mathrm{BCE}(\mathrm{Sigmoid}(\hat{g}), g), \tag{9}$$

where $\hat{g}$ and g is the predicted logit of flag token and ground-truth [FLAG] respectively.

Our model is trained with six losses simultaneously in an end-to-end manner. The total loss is calculated as follows:

$$L = L_{\text{class}} + L_{\text{start}} + L_{\text{end}} + L_{\text{diou}} + L_{\text{flag}}. \tag{10}$$

Note that all loss coefficients are set to 1 and loss balancing is not required.

4 Experiments

4.1 Experimental Setting

Datasets. We evaluate our method and previous methods on two On-TAL benchmarks: THUMOS14 [13] and MUSES [22]. THUMOS14 contains 200 videos for training and 213 videos for testing with 20 action classes, while MUSES has 2,587 videos for training and 1,110 videos for testing with 25 action classes. MUSES consists of multi-shot action instances, which makes action localization more challenging.

Hyperparameters. We use a frozen two-stream TSN [34] pretrained on Kinetics [4] to extract RGB and flow features for THUMOS14, and an I3D [4] trained on Kinetics [4] utilizing only RGB features for MUSES, following previous On-TAL work [14, 15, 31]. The segment size L_s is set to 64 in THUMOS14 and 75 in MUSES. The dimension of the segment feature is set to $D = 1024$ for both datasets. For the memory-augmented video encoder, we stack 3 Transformer layers with 8 attention heads for both datasets. The flag threshold θ is set to 0.5. The memory size L_m is set to 7 for THUMOS14, while 15 for MUSES. For the instance decoding module, we stack 5 Transformer layers with 4 attention heads for both datasets. The number of the class query and the boundary query $N = 10$ for THUMOS14 and $N = 6$ for MUSES.

Training. During training, we use Adam optimizer [16], with the initial learning rate of 1e 8 with CosineAnnealing scheduler [26]. The batch size is set to 64 for THUMOS14, while 75 for MUSES. The focal loss coefficient is $\alpha = 0.25$ and $\gamma = 2$ for THUMOS14 and $\alpha = 1$ and $\gamma = 5$ for MUSES. All loss coefficients are set to 1. More details are listed in the supplementary material (Sec. D).

4.2 Comparison with the State of the Art

We compare our method with previous online and offline TAL methods on THUMOS14 and MUSES dataset. As presented in Table 1, our method outperforms all previous On-TAL methods in both benchmarks by a substantial margin: 4.9%p of average mAP in THUMOS14 and 0.7%p of average mAP in MUSES. The results show that MATR utilizing the memory queue is more effective than previous OAD-based methods [14, 31, 33, 46] and instance-level method OAT-OSN [15]. We also evaluate two OAD methods utilizing memory [33, 46] in On-TAL setting by grouping their frame-level predictions into action instances.

Table 1: Comparison with On-TAL methods on THUMOS14 and MUSES dataset. The results are reported in mAP measure (%). '*' indicates that the result is validated using the On-TAL ground truth from classification annotation.

Method	THUMOS14							MUSES						
	backbone	0.3	0.4	0.5	0.6	0.7	Average	backbone	0.3	0.4	0.5	0.6	0.7	Average
Offline TAL														
G-TAD [39]	TSN [34]	54.5	47.6	40.2	30.8	23.4	39.9	I3D [4]	19.1	14.8	11.1	7.4	4.7	11.4
ContextLoc [49]	I3D	68.3	63.8	54.3	41.8	26.2	50.9	–	–	–	–	–	–	–
P-GCN [43]	I3D	63.6	57.8	49.1	–	–	–	I3D	19.9	17.1	13.1	9.7	5.4	13.0
MUSES [22]	I3D	68.9	64.0	56.9	46.3	31.0	53.4	I3D	25.9	22.6	18.9	15.0	10.6	18.6
ActionFormer [44]	I3D	82.1	77.8	71.0	59.4	43.9	66.8	–	–	–	–	–	–	–
TriDet [27]	I3D	83.6	80.1	72.9	62.4	47.4	69.3	–	–	–	–	–	–	–
OAD-based Online TAL														
TeSTra [46]	TSN	35.6	29.2	21.4	13.4	7.6	21.4	–	–	–	–	–	–	–
MAT [33]	I3D	36.0	27.5	19.2	12.3	6.4	20.3	–	–	–	–	–	–	–
CAG-QIL [14]	TSN	44.7	37.6	29.8	21.9	14.5	29.7	I3D	8.5	6.5	4.2	2.8	1.9	4.8
SimOn [31]	TSN	54.3	45.0	35.0	23.3	14.6	34.4	–	–	–	–	–	–	–
SimOn* [31]	TSN	57.0	47.5	37.3	26.6	16.0	36.9	–	–	–	–	–	–	–
Instance-level Online TAL														
OAT-Naive [15]	TSN	57.6	50.6	43.0	30.0	15.7	39.4	I3D	20.3	16.6	12.9	7.7	3.6	12.2
OAT-OSN [15]	TSN	63.0	56.7	47.1	36.3	20.0	44.6	I3D	22.1	18.5	14.2	8.9	4.7	13.7
MATR	TSN	**70.3**	**62.7**	**52.1**	**38.6**	**23.7**	**49.5**	I3D	**23.5**	**19.3**	**14.3**	**9.4**	**5.7**	**14.4**

However, their performance is far lower than MATR, which verifies the effectiveness of the proposed memory queue of MATR. We also compare our model with several offline TAL methods. Although there is still a performance gap between online and offline methods, our model achieves comparable performance to offline methods, and even outperforms previous work [39,43]. The performance on the MUSES dataset is lower compared to THUMOS14 and the reason is twofold: (1) Action instances in MUSES are captured in multi-shots, making it challenging to detect action boundaries, and (2) there are confusing action classes such as "quarrel" and "fight," making action classification hard. These make MUSES challenging and limit the upper-bound of performance on it.

4.3 Ablation Studies

We perform ablation studies on THUMOS14 and MUSES dataset to show the effectiveness of the proposed modules.

Ablation on the Proposed Modules. In Table 2, we conduct an ablation study to verify the effectiveness of each component of our model. In Table 2)(a), focuses on the components of the memory-augmented video, both removing the flag token and the segment encoder led to a performance drop. The results show the efficacy of selectively storing the past information and the importance of temporal context across the input segment. In Table 2(b), we use a single decoder and predict both start and end at once, which results significant performance drop, from 49.5 mAP to 42.7 mAP. The result demonstrates the contribution of our new action localization method, using two separate decoders for the start and the end prediction. We also ablate new query design, sampling the memory for the start decoder, and positional embedding for instance queries. The results

Table 2: Ablation of the proposed modules on THUMOS14 dataset. The results are reported in mAP measure (%).

Method	0.3	0.4	0.5	0.6	0.7	Average
Ours	**70.3**	**62.7**	**52.1**	38.6	**23.7**	**49.5**
(a) Memory-augmented video encoder						
w/o flag token	67.3	61.0	50.1	36.5	22.3	47.4
w/o segment encoder	65.8	59.3	49.9	36.3	21.8	46.6
(b) Instance decoding module						
single decoder	65.1	56.4	45.0	30.5	16.7	42.7
w/o splitting query	66.2	61.4	50.9	38.0	23.2	47.9
w/o sampling	66.1	59.6	50.6	**39.9**	22.5	47.2
w/o pos embedding	67.2	60.5	49.8	36.3	22.7	47.3
(c) Training objective						
w/o DIoU loss	62.2	53.4	43.2	31.0	17.4	41.4

Table 3: Analysis of the maximum length of the memory queue.

Memory size	0.3	0.4	0.5	0.6	0.7	Average
(a) THUMOS14 dataset						
w/o mem	65.8	58.9	47.6	36.9	20.8	46.0
1	67.2	59.9	50.7	37.9	22.7	47.7
3	67.3	61.9	51.9	37.9	22.6	48.3
7	**70.3**	**62.7**	52.1	38.6	23.7	**49.5**
11	67.9	60.6	49.8	37.8	23.2	47.9
15	66.9	60.1	50.7	38.4	24.1	48.0
19	67.0	60.1	**52.5**	**40.2**	**24.6**	49.1
(b) MUSES dataset						
w/o mem	22.3	17.6	13.1	9.1	4.6	13.3
1	23.0	18.0	13.5	8.6	4.9	13.6
3	23.1	18.3	13.9	8.9	5.1	13.9
7	**23.5**	18.1	13.5	8.9	4.9	13.8
11	22.9	18.7	14.0	9.4	5.4	14.1
15	**23.5**	**19.3**	**14.3**	9.4	**5.7**	**14.4**
19	22.7	18.9	13.8	**9.6**	5.5	14.1

Table 4: Comparison between different memory modules on the THUMOS14.

Memory module	Backbone	Segment size	Inference time	fps	Memory parameters	Average mAP
MAT [33]	TSN [34]	64	191.9ms	5.2	40.1M	46.9
E2E-LOAD [2]	TSN	64	196.1ms	5.1	53.1M	47.9
MATR	TSN	64	**167.1ms**	**6.0**	**24.0M**	**49.5**

show that all design choices of the proposed components are important for action instance localization. Lastly, training our model without DIoU loss shows a drastic performance drop, which demonstrates the importance of instance-level supervision for On-TAL (Table 2(c)).

Impact of the Memory Queue Size. We conduct an ablation study on the different memory queue sizes. As shown in Table 3, employing the memory queue enhances the performance compared to the without memory case, and generally the performance improves as the memory queue size increases. With an adequately large memory queue size covering the top 99% of the instances in the training split, such as 7 for the THUMOS14 and 5 for the MUSES, the model shows robust performance and the performance is higher than the previous state-of-the-art model [15]. The memory queue size of 7 and 15 shows the best on THUMOS14 and MUSES, respectively.

Comparison of the proposed memory queue with OAD methods using memory. We compare our memory module with previous memory modules from OAD tasks [2, 33] by replacing the memory-augmented video encoder (Sec. 3.2) of our model. For a fair comparison, hidden dimension of the attention blocks for all methods is set to 1024. As shown in Table 4, MATR outperforms the existing memory modules in terms of both mAP and space-time complexity. While existing memory modules use multiple attention blocks, our memory module consists of a segment encoder and a flag prediction head, making it superior in terms of parameters and inference speed.

Analysis of the Segment Size. As shown in Fig. 4, our model is less sensitive to the input segment size, which should be carefully tuned across various

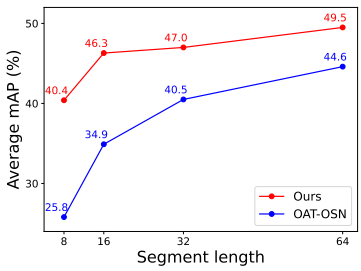

Fig. 4: Average mAP (%) versus the segment size in THUMOS14. Red line represents the result of our model and blue line represents the result of OAT-OSN.

Table 5: Ablation on the start and the end prediction heads. Reg and Cls refer to the offset regression and the region classification respectively.

Start	End	0.3	0.4	0.5	0.6	0.7	Average
Reg	Reg	64.2	59.0	50.0	37.8	22.7	46.7
Reg + Cls	**Reg**	**70.3**	**62.7**	**52.1**	**38.6**	**23.7**	**49.5**
Reg + Cls	Reg + Cls	64.0	58.9	50.2	38.3	23.2	46.9

Table 6: Ablation on the memory compression factor on the THUMOS14 dataset.

Compress factor	0.3	0.4	0.5	0.6	0.7	Average
None (Ours)	**70.3**	**62.7**	52.1	38.6	23.7	**49.5**
2	67.3	59.9	49.4	36.9	22.3	47.7
4	68.1	61.5	**52.7**	**39.7**	**24.5**	49.3
8	67.5	61.3	52.0	38.8	23.8	48.7

Table 7: Comparison of the inference time and # parameters on the THUMOS14.

Method	Backbone	Segment size	Inference time	fps	# Parameters	Average mAP
OAT-OSN [15]	TSN [34]	64	163.7ms	6.1	128.7M	44.6
MATR	TSN	64	167.1ms	6.0	192.8M	49.5
MATR	TSN	16	53.8ms	18.6	192.8M	46.3

datasets. While the performance of OAT-OSN drops significantly from 44.6 mAP to 25.8 mAP when the segment size is reduced from 64 to 8, the performance drop of ours is only 9.1%p.

Analysis of Start and End Prediction Heads. We investigate various offset prediction methods for the start prediction and the end prediction (Table 5). For the first method, start offset regression without region classification, the offset is normalized using the offset statistics from the training dataset. The second approach, which adopts region classification and offset regression for the start prediction, and only offset regression for the end prediction, exhibits the best performance. The reason for performance drop when using region classification for the end prediction is that frame-wise end classification is a complex problem, adversely affects the performance.

Analysis of Memory Compression Factor. We also conduct an experiment on efficiently storing input segment features in the memory by compressing them. We utilize a 1D convolution layer to compress the number of segment features. As shown in Table 6, 4 compression yields competitive results with efficient memory usage. However, we decide not to use this method due to the efficient utilization of memory already achieved through the flag token and avoid to use additional hyperparameters that require optimization for each dataset.

Inference Time. As shown in Table 7, when using the same backbone and the segment size, the inference time of MATR is almost equal to that of OAT-OSN, yet its average mAP is significantly better. Moreover, by reducing the segment length to 16, MATR outperforms OAT-OSN in terms of both mAP and

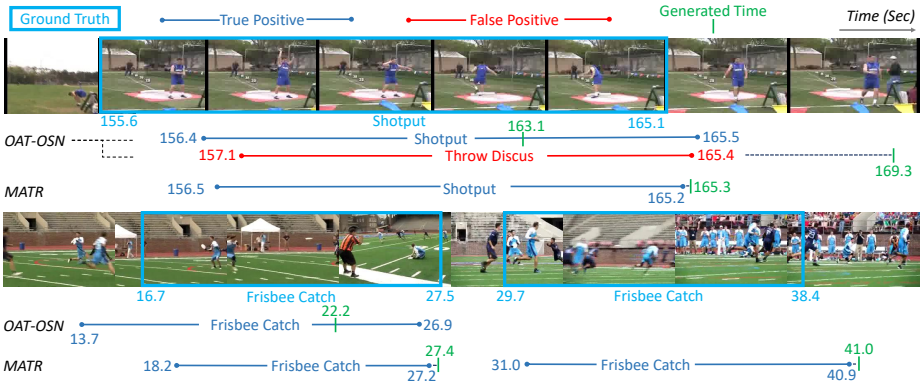

Fig. 5: Qualitative results of MATR and OAT-OSN [15] on the THUMOS14. Generated time is the timestamp when the predicted instance is generated.

the inference time. MATR uses more parameters due to the separation of the decoder into the start decoder and the end decoders.

4.4 Qualitative Results

Fig. 5 illustrates the predictions of our model and OAT-OSN [15]. The results show that MATR has the capability to efficiently detect action instances by identifying intervals immediately after the current action ends. Additionally, the results show that our model is able to classify and localize action instances more precisely than OAT-OSN.

5 Conclusion

In this paper, we address online temporal action localization by introducing a new model, MATR, which leverages the memory queue to exploit long-term context. We also propose a new action instance localization method based on Transformer, which separately identifies action starts and action ends. As a result, MATR not only surpasses existing online TAL methods but also demonstrates comparable performance to offline TAL methods. The code base will be open to the public to promote future research on On-TAL.

Limitations: When the memory queue contains multiple action instances, there is a risk of action instances being matched with incorrect start points. Additionally, when storing the input segment into the memory queue, MATR only considers whether the input segment is relevant to action instances but does not leverage the relationship with the past context stored in the memory. To address these issues, investigating a method to utilize the information stored in the memory queue when storing the input segment features, could lead to more effective memory storage and utilization.

Acknowledgement. This work was supported by the NRF grant and the IITP grant funded by Ministry of Science and ICT, Korea (RS-2019-II191906, RS-2021-II212068, RS-2022-II220290, NRF-2021R1A2C3012728).

References

1. Buch, S., Escorcia, V., Shen, C., Ghanem, B., Carlos Niebles, J.: Sst: Single-stream temporal action proposals. In: Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 2911–2920 (2017)
2. Cao, S., Luo, W., Wang, B., Zhang, W., Ma, L.: E2e-load: End-to-end long-form online action detection. Proc. IEEE International Conference on Computer Vision (ICCV) (2023)
3. Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., Zagoruyko, S.: End-to-end object detection with transformers. In: Proc. European Conference on Computer Vision (ECCV). pp. 213–229. Springer (2020)
4. Carreira, J., Zisserman, A.: Quo vadis, action recognition? a new model and the kinetics dataset. In: Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2017)
5. 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)
6. Cheng, F., Bertasius, G.: Tallformer: Temporal action localization with a long-memory transformer. In: Proc. European Conference on Computer Vision (ECCV). pp. 503–521. Springer (2022)
7. De Geest, R., Gavves, E., Ghodrati, A., Li, Z., Snoek, C., Tuytelaars, T.: Online action detection. In: Proc. European Conference on Computer Vision (ECCV). pp. 269–284. Springer (2016)
8. Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., Houlsby, N.: An image is worth 16x16 words: Transformers for image recognition at scale. In: Proc. International Conference on Learning Representations (ICLR) (2021), https://openreview.net/forum?id=YicbFdNTTy
9. Escorcia, V., Caba Heilbron, F., Niebles, J.C., Ghanem, B.: Daps: Deep action proposals for action understanding. In: Proc. European Conference on Computer Vision (ECCV). pp. 768–784. Springer (2016)
10. Gao, J., Yang, Z., Chen, K., Sun, C., Nevatia, R.: Turn tap: Temporal unit regression network for temporal action proposals. In: Proc. IEEE International Conference on Computer Vision (ICCV). pp. 3628–3636 (2017)
11. Gao, M., Xu, M., Davis, L.S., Socher, R., Xiong, C.: Startnet: Online detection of action start in untrimmed videos. In: Proc. IEEE International Conference on Computer Vision (ICCV). pp. 5542–5551 (2019)
12. Heilbron, F.C., Niebles, J.C., Ghanem, B.: Fast temporal activity proposals for efficient detection of human actions in untrimmed videos. In: Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 1914–1923 (2016)
13. 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)
14. 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: Proc. IEEE International Conference on Computer Vision (ICCV). pp. 13729–13738 (2021)

15. Kim, Y.H., Kang, H., Kim, S.J.: A sliding window scheme for online temporal action localization. In: Proc. European Conference on Computer Vision (ECCV). pp. 653–669. Springer (2022)
16. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. In: Proc. International Conference on Learning Representations (ICLR) (2015)
17. 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: Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 3320–3329 (2021)
18. Lin, T., Liu, X., Li, X., Ding, E., Wen, S.: Bmn: Boundary-matching network for temporal action proposal generation. In: Proc. IEEE International Conference on Computer Vision (ICCV). pp. 3889–3898 (2019)
19. Lin, T., Zhao, X., Su, H., Wang, C., Yang, M.: Bsn: Boundary sensitive network for temporal action proposal generation. In: Proc. European Conference on Computer Vision (ECCV). pp. 3–19 (2018)
20. Lin, T.Y., Goyal, P., Girshick, R., He, K., Dollár, P.: Focal loss for dense object detection. In: Proc. IEEE International Conference on Computer Vision (ICCV). pp. 2980–2988 (2017)
21. Liu, Q., Wang, Z.: Progressive boundary refinement network for temporal action detection. In: Proc. AAAI Conference on Artificial Intelligence (AAAI). vol. 34, pp. 11612–11619 (2020)
22. Liu, X., Hu, Y., Bai, S., Ding, F., Bai, X., Torr, P.H.: Multi-shot temporal event localization: a benchmark. In: Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 12596–12606 (2021)
23. Liu, X., Wang, Q., Hu, Y., Tang, X., Zhang, S., Bai, S., Bai, X.: End-to-end temporal action detection with transformer. IEEE Transactions on Image Processing **31**, 5427–5441 (2022)
24. Liu, Y., Ma, L., Zhang, Y., Liu, W., Chang, S.F.: Multi-granularity generator for temporal action proposal. In: Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 3604–3613 (2019)
25. Long, F., Yao, T., Qiu, Z., Tian, X., Luo, J., Mei, T.: Gaussian temporal awareness networks for action localization. In: Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 344–353 (2019)
26. Loshchilov, I., Hutter, F.: Sgdr: Stochastic gradient descent with warm restarts. arXiv preprint arXiv:1608.03983 (2016)
27. Shi, D., Zhong, Y., Cao, Q., Ma, L., Li, J., Tao, D.: Tridet: Temporal action detection with relative boundary modeling. In: Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 18857–18866 (2023)
28. 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: Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 5734–5743 (2017)
29. 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: Proc. European Conference on Computer Vision (ECCV). pp. 534–551 (2018)
30. Shou, Z., Wang, D., Chang, S.F.: Temporal action localization in untrimmed videos via multi-stage cnns. In: Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 1049–1058 (2016)
31. Tang, T.N., Park, J., Kim, K., Sohn, K.: Simon: A simple framework for online temporal action localization. arXiv preprint arXiv:2211.04905 (2022)

32. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. In: Proc. Advances in Neural Information Processing Systems (NeurIPS). pp. 5998–6008 (2017)
33. Wang, J., Chen, G., Huang, Y., Wang, L., Lu, T.: Memory-and-anticipation transformer for online action understanding. In: Proc. IEEE International Conference on Computer Vision (ICCV). pp. 13824–13835 (2023)
34. 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: Proc. European Conference on Computer Vision (ECCV). pp. 20–36. Springer (2016)
35. Wang, X., Zhang, S., Qing, Z., Shao, Y., Zuo, Z., Gao, C., Sang, N.: Oadtr: Online action detection with transformers. In: Proc. IEEE International Conference on Computer Vision (ICCV). pp. 7565–7575 (2021)
36. Wu, C.Y., Li, Y., Mangalam, K., Fan, H., Xiong, B., Malik, J., Feichtenhofer, C.: Memvit: Memory-augmented multiscale vision transformer for efficient long-term video recognition. In: Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 13587–13597 (2022)
37. Wu, Y., Chen, Y., Yuan, L., Liu, Z., Wang, L., Li, H., Fu, Y.: Rethinking classification and localization for object detection. In: Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 10186–10195 (2020)
38. Xu, M., Perez Rua, J.M., Zhu, X., Ghanem, B., Martinez, B.: Low-fidelity video encoder optimization for temporal action localization. Proc. Advances in Neural Information Processing Systems (NeurIPS) **34**, 9923–9935 (2021)
39. Xu, M., Zhao, C., Rojas, D.S., Thabet, A., Ghanem, B.: G-tad: Sub-graph localization for temporal action detection. In: Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 10156–10165 (2020)
40. Xu, M., Gao, M., Chen, Y.T., Davis, L.S., Crandall, D.J.: Temporal recurrent networks for online action detection. In: Proc. IEEE International Conference on Computer Vision (ICCV). pp. 5532–5541 (2019)
41. Xu, M., Xiong, Y., Chen, H., Li, X., Xia, W., Tu, Z., Soatto, S.: Long short-term transformer for online action detection. Proc. Advances in Neural Information Processing Systems (NeurIPS) **34**, 1086–1099 (2021)
42. Yang, L., Han, J., Zhang, D.: Colar: Effective and efficient online action detection by consulting exemplars. In: Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 3160–3169 (2022)
43. Zeng, R., Huang, W., Tan, M., Rong, Y., Zhao, P., Huang, J., Gan, C.: Graph convolutional networks for temporal action localization. In: Proc. IEEE International Conference on Computer Vision (ICCV). pp. 7094–7103 (2019)
44. Zhang, C.L., Wu, J., Li, Y.: Actionformer: Localizing moments of actions with transformers. In: Proc. European Conference on Computer Vision (ECCV). pp. 492–510. Springer (2022)
45. Zhao, C., Thabet, A.K., Ghanem, B.: Video self-stitching graph network for temporal action localization. In: Proc. IEEE International Conference on Computer Vision (ICCV). pp. 13658–13667 (2021)
46. Zhao, Y., Krähenbühl, P.: Real-time online video detection with temporal smoothing transformers. In: Proc. European Conference on Computer Vision (ECCV). pp. 485–502. Springer (2022)
47. Zhao, Y., Xiong, Y., Wang, L., Wu, Z., Tang, X., Lin, D.: Temporal action detection with structured segment networks. In: Proc. IEEE International Conference on Computer Vision (ICCV). pp. 2914–2923 (2017)

48. Zheng, Z., Wang, P., Liu, W., Li, J., Ye, R., Ren, D.: Distance-iou loss: Faster and better learning for bounding box regression. In: Proc. AAAI Conference on Artificial Intelligence (AAAI). vol. 34, pp. 12993–13000 (2020)
49. Zhu, Z., Tang, W., Wang, L., Zheng, N., Hua, G.: Enriching local and global contexts for temporal action localization. In: Proc. IEEE International Conference on Computer Vision (ICCV). pp. 13516–13525 (2021)