A Detailed Explanation of Prior Works

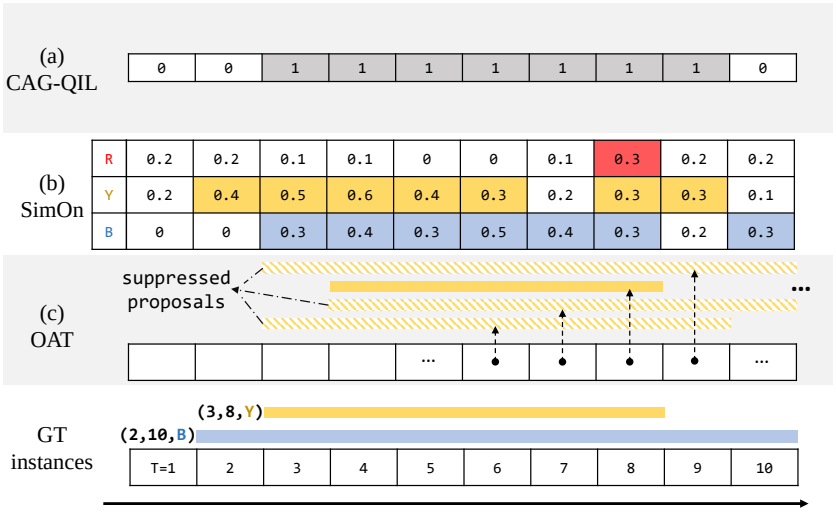

Fig. 1. Overview of previous methods. In this figure, we assume 3 different classes, denoted as "R, Y, B", and two actual ground truth action instances (2,10,B) and (3,8,Y). For CAG-QIL [5], a generated action instance is colored gray since it produces class-agnostic proposals. The threshold value is set to 0.25 in SimOn [17]. Unlike methods using softmax, SimOn employs sigmoid functions for each class logit in its multilabel classification approach, leading to class probabilities that may not sum to 1. In OAT [6], we have shown only the class Y proposals in $t = 6, 7, 8, 9$ for clarity, but note that the same process is applied to both R and B for each timestep.

To provide a better understanding of prior works, we depict their proposal generation processes in Fig. 1. CAG-QIL [5] employs a discrete binary decision-making module per timestep, forming class-agnostic action proposals by simply grouping '1's (see Fig 1 (a)). However, this method fails to recognize overlapping action instances.

Conversely, SimOn [17] assigns continuous per-class scores each timestep. A specific threshold, set to 0.25 in Fig 1, transforms these scores into binary decisions, leading to class-aware action proposals (Fig 1 (b)). While this allows detecting overlapping action instances with *different* classes, it tends to generate an excessive number of proposals as the class number increases, a phenomenon confirmed in our main paper's experimental section.

OAT [6], an anchor-based TAL extension, continuously generates action proposals at each timestep. However, registering all of them as final predictions leads to a high false positive rate. To mitigate this, OAT incorporates a suppression strategy (Fig. 1). At $T = 6$ and $T = 7$, the Online Suppression Net-

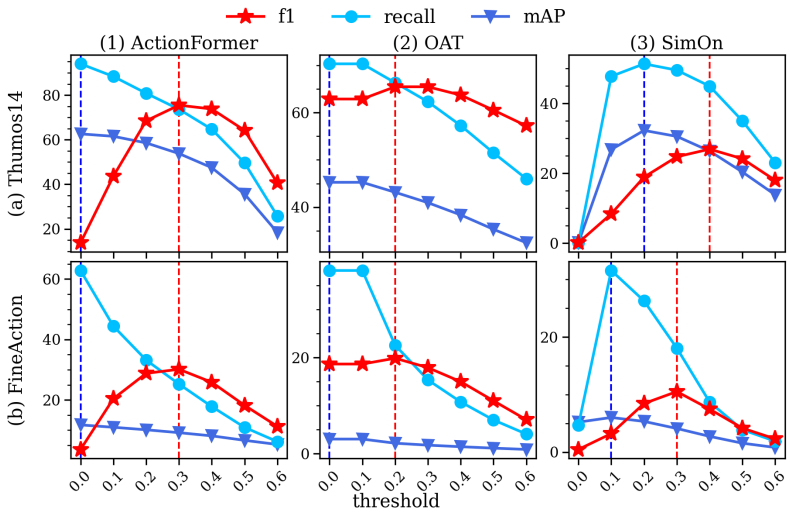

Fig. 2. Analysis of the F1 score, recall, and standard mAP against different settings. The x-axis displays the thresholds used to filter out action instances with low confidence scores in ActionFormer and OAT, and the thresholds applied to the per-frame class logit to determine frame-wise action decisions in SimOn. The red dotted vertical lines indicate the optimal threshold for F1 score, while blue dotted lines denote the optimal threshold for recall and mAP score.

work (OSN) suppresses proposals by emitting a continuous value; if this exceeds the $threshold_{OSN}$, a proposal is accepted (e.g., at $T = 8$). However, subsequent proposals are not OSN-suppressed but evaluated against a predefined $threshold_{tIOU}$. For instance, a proposal at $T = 9$ is suppressed due to its tIOU with the registered proposal (the proposal at $T = 8$) exceeding $threshold_{tIOU}$, independent of OSN output. This approach accounts for OSN's lack of awareness of its previous outputs and inability to adjust subsequent outputs accordingly. Nonetheless, this "hard" suppression limits the capability of OAT.

Moreover, while CAG-QIL and SimOn are capable of immediately detecting the start of an action instance by comparing consecutive decisions, OAT identifies the action start only upon finalizing its prediction. (See Fig. 1. At $T = 8$, it pinpoints the action start $T = 4$, while CAG-QIL and SimOn immediately identify it by detecting the decision change.) This approach, though it allows OAT to fully exploit the whole context of action instances, suggests it addresses a simpler challenge than CAG-QIL and SimOn, and restricts its effectiveness in situations where prompt detection of action is critical.

B Discussion about Metric

In our main paper, we suggest using the F1 score with Hungarian matching [7] to evaluate class-agnostic On-TAL performance. In this section, we discuss this further.

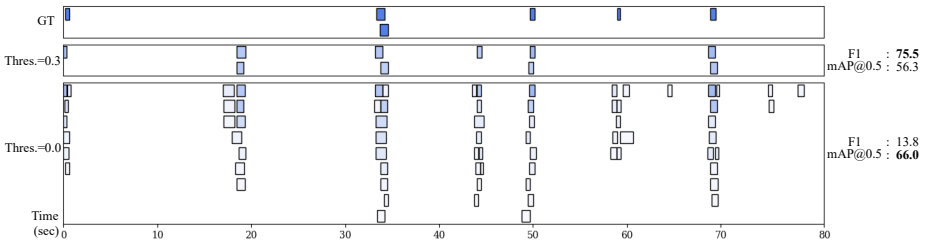

Fig. 3. Visualization of detection results with varying confidence threshold values. In addition to the ground truth (GT), we show the output of ActionFormer [21] with the best F1 score and mAP, respectively. The transparency of the rectangles indicates the confidence score, with higher transparency indicating a lower confidence score.

Video ID	1	2	3	4	5	6	7	8	9	10	Avg.
Preference (%)	98.33	93.33	93.33	91.67	95.00	98.33	96.67	98.33	95.00	96.67	95.67

Table 1. Human preference results. "Preference" here denotes preference of F1-best detection results over mAP counterpart.

B.1 Relationship between F1 and mAP

We find that mAP does not appropriately penalize false positives, which is critical in real-world scenarios. To provide concrete evidence, we analyze Actionformer [21], OAT [6], and SimOn [17] on two different datasets [4,**?**]. Blue dotted lines in Fig. 2 show that the threshold value that produces the best mAP metric is consistently the same as the threshold value that yields the best *recall*. This indicates that the mAP metric is solely dependent on recall, thereby suggesting that in general, lower thresholds with an excessive number of action proposals lead to better mAP results[1]. The finding supports the prevailing practice in offline TAL literature of using Average Recall (AR) across varying tIOU thresholds for evaluating class-agnostic action proposal generation [8,**?**,**?**,**?**,**?**] as recall aligns with the mAP metric, the primary evaluation measure in standard TAL research. However, it contrasts with online scenarios where retrospective modification of predictions is not permitted, thereby making the suppression of false positives a challenging task.

To this end, we suggest using the F1 score as the evaluation metric. This metric employs the Hungarian matching algorithm [7], which provides optimal bipartite matching between class-agnostic ground truth $\{(t_m^s, t_m^e)\}_{m=1}^M$ and the model's predictions $\{(\hat{t}_m^s, \hat{t}_m^e)\}_{m=1}^{\hat{M}}$ in terms of tIoU, and computes the F1 score in a certain tIOU threshold. Unlike the mAP metric, it appropriately penalizes

[1] Using extremely low threshold values in SimOn [17] leads to overly merged proposals. Thus, while a lower threshold value does not necessarily result in higher recall in SimOn, it is true for Actionformer and OAT.

false positives, which is evident from the observation in Fig. 2 where adequately suppressing low-confidence proposals leads to better F1 scores.

To illustrate this visually, we present detection results from ActionFormer [21] at two different confidence thresholds (Figure 3). The confidence threshold here is employed to eliminate low-confidence proposals, so a zero-threshold setting implies that all detection results are considered final without any suppression. Intriguingly, despite the mAP metric's inherent design to penalize false positives, the method with no suppression yields much higher mAP scores, while the method with suppression yields higher (Hungarian) F1 scores.

Numerous low-confidence proposals are not only unsuitable for many online detection scenarios but also misalign with the natural human perception. To support our findings, we carried out a survey involving 50 participants from Amazon Mechanical Turk. The participants were shown the original video and two detection results (one with the highest F1 score and another with the highest mAP score) and were asked to indicate their preference. The survey results, presented in Table 1, suggest that the detection setting with a higher F1 score better aligns with the general human perception.

B.2 Problem of Using Confidence Score

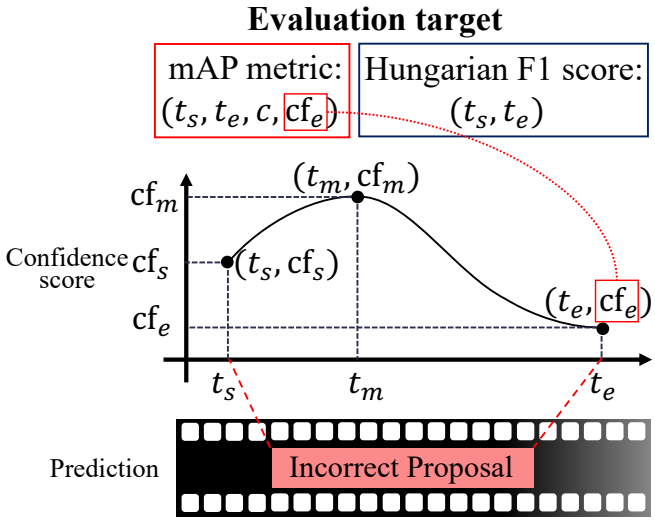

Fig. 4. Problem of mAP metric for evaluating state-aware On-TAL. Here, cf_t refers to the instantaneous confidence score of the action instance (t_s, t) at timestep t.

When evaluating action proposals using mAP, proposals are ranked based on their confidence scores. It is important to note that this confidence score is

about the entire action proposal and thus is determined only after observing the full action instance. This approach, however, poses challenges in scenarios where the model must be aware of the ongoing action status, a situation frequently encountered in streaming perception [13,16,**?**]. In such cases, the reliance on confidence scores, which are contingent upon the full action instance, becomes problematic.

To provide a concrete example, we reference Fig. 4. Suppose the model generates an incorrect action proposal. At both t_s and t_m, the model makes an incorrect action proposal because it detects the action with a high confidence score. However, only when it reaches the endpoint t_e does the model realize that the action does not exist, and significantly lowers the confidence score. This means that the model has made incorrect detections during the action progress, but when evaluated using mAP scores, the lowered confidence score at t_e is used, so that the model is less penalized for its significant misdetection during $t_s < t < t_e$.

On the other hand, the F1 score avoids this problem by considering only the start and end timesteps of the action instance when evaluating the performance of the action proposal. Unlike the mAP metric, which can lower its confidence score afterward to revert its previous misdetection, the F1 score does not rely on confidence scores in its evaluation and thus prevents action proposals from being suppressed in hindsight. Therefore, the F1 metric provides a more accurate assessment of the model's On-TAL performance in terms of identifying the instantaneous status of ongoing action.

In addition, numerous real-world applications of On-TAL, such as autonomous driving and robotic interactions, prioritize accurate detection. In such cases, concise and suppressed detection results are preferable to superfluous low-confidence detection. We argue that, under these circumstances, evaluating final hard predictions (binary existence/non-existence format after filtering out low-confidence proposals using a predefined threshold) bears greater significance than assessing intermediate soft predictions (raw detection outcomes accompanied by confidence scores). By concentrating solely on the final output, the F1 metric delivers a more practical and representative assessment of a model's performance in real-world situations where accuracy and brevity in detection are of utmost importance.

B.3 Pseudocode of F1 metric

For clear demonstration, we provide a Python implementation of the algorithm calculating the F1 score. (Algorithm 1)

C Dataset Analysis

In this section, we provide a detailed analysis of dataset candidates for proper On-TAL performance assessment.

Algorithm 1 `get_hungarian_score` in Python.

```
def get_hungarian_score(answer:list, prediction:list, iou_threshold=0.5):
    '''
    IN: answer[[st,ed],[st,ed]...],
    prediction[[st,ed],[st,ed]...]
    OUT: f1_score (float)
    '''
    profit = np.zeros((len(answer), len(prediction)))
    #calculate pairwise iou
    for i in range(len(answer)):
        for j in range(len(prediction)):
            profit[i][j] = calculate_iou(answer[i], prediction[j])
    #perform Hungarian Matching
    r, c = optimize.linear_sum_assignment(profit, maximize=True)
    tp = np.sum(np.where(profit[r, c] >= iou_threshold, 1, 0))
    a = answer.shape[0]
    p = prediction.shape[0]
    #return F1 score
    return 2*tp/(a+p)
```

THUMOS14 [4] is composed of 413 sports videos, with 200 allocated for training and 213 for testing. It encompasses 6316 action instances, representing 20 different action classes with an average of 15 action instances per video. 17.5% of the action instances have mutual overlapping, making the dataset more challenging. However, while the dataset offers many fine-grained action instances in a complex manner, its primary drawback is its **relatively small number of total videos**.

FineAction [12] is a large-scale collection of 16,732 daily event videos, featuring a total of 103,324 action instances across 106 distinct action classes. On average, each video contains 6 action instances, and approximately 11.5% of these instances exhibit mutual overlap. FineAction can be considered an expanded and more challenging version of the THUMOS14 dataset, providing fine-grained and complex action instances. To the best of our knowledge, FineAction is one of the most suitable datasets for assessing On-TAL performance, as its size and variety of action instances exceed most temporally annotated datasets.

MUSES [9] includes 3,697 videos, with a total duration of 716 hours, and has annotations for 25 action classes sourced from television and movie dramas. On average, each video has 8.5 action instances and 3.3 action classes. A unique characteristic of this dataset is that its action instances span multiple shots, which adds to its complexity. However, from an On-TAL perspective, the dataset has some limitations. First, the action instances are not fine-grained, with an average duration of 71.69 seconds. Additionally, in many cases, the action proposals are strictly adjacent (Fig. 5 (a)), making it impossible to distinguish them in the conventional setting.

To tackle this issue, we introduce an additional state called the "Sep" state (Fig. 5 (b)), which is only reached at the end of the action instance. By incorporating this state into our ActionSwitch framework, we obtain a "5-state"

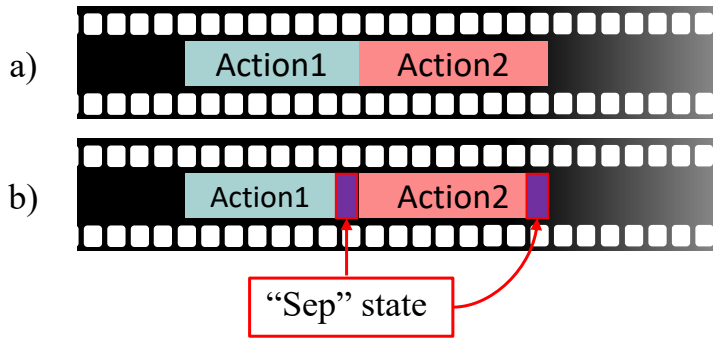

Fig. 5. (a) strictly adjacent proposals, which are prevalent in MUSES dataset. (b) "Sep" states to distinguish strictly adjacent proposals.

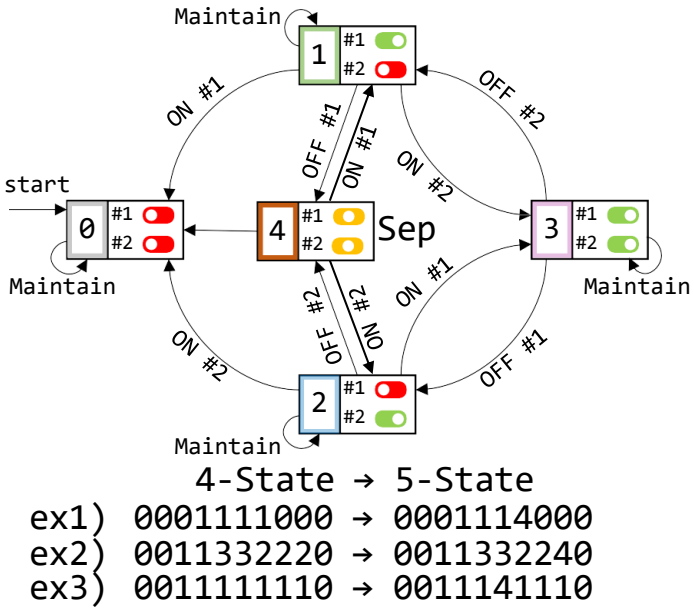

Fig. 6. 5 state formulation with additional "Sep" state. For clear demonstration, we offer three simple examples that show the differences between this new formulation and the previous 4-state one. With an additional "Sep" state, strictly adjacent proposals can be easily separated.

finite-state machine (as illustrated in Fig. 6), which requires minor modifications to the standard ActionSwitch formulation. The experimental result in the following section shows that this additional modification is critical for the MUSES dataset.

Activitynet1.3 [1] and Human Action Clips and Segments (**HACS-v1.1**) [22] are large-scale untrimmed video datasets that respectively contain 20,000 and 50,000 untrimmed videos encompassing 200 action categories. Despite their scale, they differ from THUMOS14 and FineAction in that each video only has an average of 1-2 very long action instances. Thus, these datasets may not be suitable for the objective of On-TAL, which is to identify multiple action instances in real-time streaming settings.

EPIC-Kitchens 100 [3] is a large-scale egocentric action dataset, comprising 100 hours of footage from 700 cooking sessions across various kitchens. While it has fewer videos than ActivityNet-1.3, it boasts a considerably higher average number of instances per video (128 compared to ActivityNet-1.3's 1.5). Compared to THUMOS14, EPIC-Kitchens 100 is three times larger in video hours and exceeds tenfold in the number of action instances. Additionally, these egocentric videos feature substantial camera movement, presenting unique challenges for Temporal Action Localization (TAL)

BBDB [14] dataset contains 1,172 complete baseball games, with a total of 4,254 hours of video footage, and 30 unique classes. Each video covers a full game, encompassing almost all of the classes and averaging 405 action instances per video. Although the characteristics of action instances in BBDB are well-suited for On-TAL, its heavy bias towards baseball limits its usefulness in assessing general On-TAL performance.

Charades [15] is a dataset that contains 9,848 30-second clips depicting everyday indoor activities, gathered via Amazon Mechanical Turk. The dataset includes 66,500 time-specific annotations across 157 different categories of actions. However, the short duration of clips renders the Charades less appropriate for the practical evaluation needs of On-TAL, which focuses on the analysis of long-form streaming video content.

Multithumos [20] utilizes the same video set as THUMOS14 [4], but features different annotations. It comprises 413 sports videos spanning 65 classes. On average, each video in MultiTHUMOS contains 10.5 distinct action categories, compared to 1.1 in THUMOS14. Notably, 25% of action frames in the MultiTHUMOS dataset include three or more concurrent actions, emphasizing its considerable complexity.

S3 (id=4)	S2 (id=2)	S1 (id=1)			State Label
0	0	0	$000_{(2)}$	=	0
0	0	1	$001_{(2)}$	=	1
0	1	0	$010_{(2)}$	=	2
0	1	1	$011_{(2)}$	=	3
1	0	0	$100_{(2)}$	=	4
1	0	1	$101_{(2)}$	=	5
1	1	0	$110_{(2)}$	=	6
1	1	1	$111_{(2)}$	=	7

Fig. 7. Assignment of State Labels in a 3-Switch Configuration. In this setup, "1" signifies the activation of a switch, while "0" indicates its deactivation. Using this binary logic, the current status of the switches can be represented by a unique binary sequence. By converting this sequence into a decimal number, the corresponding state label is obtained.

D Implementation Details

D.1 State Assignment

Fig. 7 illustrates the method for assigning state labels in a configuration involving 3 switches. This strategy offers several benefits. Firstly, it can be effortlessly extended to accommodate any number of switches, requiring only the addition of columns to the table shown in Fig. 7. Secondly, as highlighted in the teaser figure of our main paper, we can straightforwardly determine the corresponding state label by *adding the switch IDs*. For example, as depicted in Fig. 7, if switches S3 and S1 are activated, the corresponding state label is calculated as 4+1=5. Here, it's important to note that the ID for S3 is **4**, not 3. Similarly, in the 4 switch configuration, S4's id would be 8. With these advantages, our formulation greatly simplifies the implementation in scenarios involving multiple switches.

D.2 State Decoding

We present Python codes for the algorithms that decode a state label sequence to action instances, as shown in Algorithm 2 and 3. Specifically, `2state_decoding` implements a single switch formulation, whereas `5state_decoding` instantiates a two switch formulation by utilizing `2state_decoding` internally. Note that `5state_decoding` can also handle the "Sep" state. In Algorithm 3, we iterate through each switch and collect the proposals specific to that switch. This is achieved by deactivating all switches except the one currently being targeted. In 3 or more switch configurations, a similar process can be applied.

Algorithm 2 `2state_decoding` in Python.

```
def 2state_decoding(action_list):
    '''
    IN:
    action_list (List), ex) [0,1,1,0,0,1,1,1,0]
    should contain state labels in {0,1}
    OUT:
    proposals (List), ex) [[1,3],[5,8]]
    '''
    cur_state, st, ed, proposals = 0, 0, 0, []
    for i, state in enumerate(action_list):
        if cur_state == 0 and state == 0:
            cur_state = 0
        elif cur_state == 0 and state == 1:
            st, cur_state = i, 1
        elif cur_state == 1 and state == 0:
            ed, cur_state = i, 0
            proposals.append([st, ed])
        else:
            cur_state = 1
    return proposals
```

Algorithm 3 `5state_decoding` in Python.

```
def 5state_decoding(action_list):
    '''
    IN:
    action_list (List), ex) [0,1,1,3,3,2,2,4,0]
    should contain state labels in {0,1,2,3,4}
    OUT:
    proposals (List), ex) [[1,5],[3,7]]
    '''
    # convert state 4 to state 0...
    action_list = [i if i != 4 else 0 for i in action_list]
    #turn off button2 and make button1_proposals...
    convert_dict = {0: 0, 1: 1, 2: 0, 3: 1}
    button1_proposals = 2state_decoding([convert_dict[i] for i in action_list])
    #turn off button1 and make button2_proposals...
    convert_dict = {0: 0, 1: 0, 2: 1, 3: 1}
    button2_proposals = 2state_decoding([convert_dict[i] for i in action_list])
    #merge two lists...
    button1_proposals.extend(button2_proposals)
    return button1_proposals
```

D.3 Additional Implementation Details

Our model features a unidirectional encoder with a 4-layer stacked GRU [2]. Additionally, we have implemented an MLP-only architecture [18] as a state classifier.

E Additional Experimental Results

E.1 MUSES Experimental Result

Table 2 displays the experimental results obtained on the MUSES dataset. In terms of the F1 score, we observed that our ActionSwitch model performs on par with the TAL-Extension model [6], with significantly superior performance in the

Setting		Method	F1 0.3	F1 0.4	F1 0.5	F1 0.6	F1 0.7	Recall 0.3	Recall 0.4	Recall 0.5	Recall 0.6	Recall 0.7	# proposal
Offline TAL		ActionFormer [21]	7.80	7.62	7.27	6.67	5.76	97.24	94.99	90.65	83.15	71.85	222000
		ActionFormer† [21]	**62.81**	**59.12**	**54.00**	**47.90**	**40.97**	**57.06**	**53.71**	**49.05**	**43.51**	**37.22**	7579
Online TAL	TAL-Extension	OAT [6]	**45.65**	**42.04**	**37.47**	**31.17**	**23.68**	**81.39**	**74.91**	**66.76**	**55.54**	**42.19**	23786
	OAD-Extension	CAG-QIL [5]	45.77	39.24	32.48	25.51	19.98	34.78	29.82	24.68	19.39	15.19	4824
		SimOn [17]	35.93	30.89	25.59	20.26	15.61	54.12	46.53	38.54	30.51	23.52	18674
		ActionSwitch – 4S (Ours)	46.52	40.83	33.87	26.88	20.39	42.49	37.29	30.93	24.55	18.62	7670
		ActionSwitch – 5S (Ours)	**51.84**	**45.68**	**37.79**	**30.22**	**22.47**	**57.65**	**50.81**	**42.02**	**33.61**	**25.00**	11359

Table 2. Comparison to other TAL methods on MUSES dataset [10]. **4S** and **5S** denote 4-state and 5-state settings respectively. The number of ground truth proposals is 9278.

Setting		Method	F1 0.3	F1 0.4	F1 0.5	F1 0.6	F1 0.7	Recall 0.3	Recall 0.4	Recall 0.5	Recall 0.6	Recall 0.7	# proposal
Offline TAL		G-TAD [19]	7.11	6.88	6.36	5.60	4.52	93.21	90.29	83.38	73.50	59.29	84752
		TadTR [11]	3.91	3.77	3.54	3.21	2.70	97.28	93.78	88.07	79.73	67.23	161341
		ActionFormer [21]	14.30	14.17	13.80	12.94	11.31	**97.41**	**96.55**	**94.01**	**88.18**	**77.04**	42400
		G-TAD† [19]	61.34	57.26	51.07	44.10	35.41	59.95	55.96	49.91	43.09	34.60	3205
		TadTR† [11]	75.67	72.81	68.42	60.64	50.04	69.65	67.02	62.97	55.81	46.06	2784
		ActionFormer† [21]	**82.91**	**80.71**	**75.47**	**68.26**	**57.00**	80.91	78.77	73.65	66.62	55.63	3196
Online TAL	TAL-Extension	OAT [6]	**73.84**	**69.90**	**62.87**	**53.07**	**38.10**	**82.58**	**78.17**	**70.31**	**59.35**	**42.62**	4153
	OAD-Extension	CAG-QIL [5]	60.94	53.90	45.80	36.95	28.79	67.00	59.26	50.36	40.62	31.66	4026
		SimOn [17]	39.32	34.82	28.73	22.85	16.52	71.28	63.11	52.09	41.42	29.95	8811
		ActionSwitch (Ours)	**67.20**	**61.11**	**53.20**	**43.58**	**34.01**	**75.90**	**69.03**	**60.10**	**49.23**	**38.42**	4228

Table 3. Comparison to other TAL methods on THUMOS14 dataset [4]. The number of ground truth proposals is 3358.

Setting		Method	F1 0.3	F1 0.4	F1 0.5	F1 0.6	F1 0.7	Recall 0.3	Recall 0.4	Recall 0.5	Recall 0.6	Recall 0.7	# proposal
Offline TAL		ActionFormer [21]	4.49	4.12	3.62	2.98	2.23	**77.84**	**71.42**	**62.83**	**51.60**	**38.72**	816100
		ActionFormer† [21]	**38.10**	**34.59**	**30.14**	**24.90**	**18.84**	31.92	28.97	25.25	20.86	15.78	16369
Online TAL	TAL-Extension	OAT (*downsampled by 4*) [6]	**25.54**	**22.86**	**18.64**	**13.17**	**7.98**	**52.20**	**46.72**	**38.10**	**26.92**	**16.30**	74823
		OAT [6]	11.38	9.38	7.21	5.23	3.26	12.85	10.60	8.14	5.90	3.69	30496
	OAD-Extension	CAG-QIL [5]	24.15	19.89	15.67	12.08	8.90	19.62	16.15	12.73	9.82	7.23	15138
		SimOn [17]	17.29	13.93	10.53	7.69	5.24	29.68	23.90	18.06	13.20	9.00	58956
		ActionSwitch (Ours)	**29.80**	**24.97**	**19.43**	**14.55**	**10.08**	**33.37**	**27.96**	**21.76**	**16.29**	**11.28**	29972

Table 4. Comparison to other TAL methods on FineAction dataset [12]. The number of ground truth proposals is 24236.

Method	Offsets 1	2	3	4	5	6	7	8	9	10	# detection	# GT
CAG-QIL [5]	28.30	42.86	50.12	54.27	57.00	58.22	59.77	60.33	61.03	61.50	4026	3358
SimOn [17]	31.45	46.22	54.11	**59.68**	**62.69**	**65.07**	**66.73**	**67.71**	**69.25**	**69.93**	10086	3358
ActionSwitch (Ours)	**33.06**	**47.06**	**54.44**	58.74	61.13	62.82	63.74	64.71	65.34	66.02	4091	3358

Table 5. Comparison of other state-of-the-art ODAS methods on THUMOS14 dataset [4]. As Kinetics pretrained features are utilized in all three models, reported mAP is much higher than those in the original papers [5,**?**]. For each metric, the best is bolded.

low tIoU setting and slightly lower in the high tIoU setting, thus highlighting ActionSwitch's exceptional efficacy. Nonetheless, it is worth noting that the 5-state formulation (Section C) is crucial for achieving good performance on the MUSES dataset due to its unique characteristics.

E.2 Additional Experimental Results

Table 3 and 4 present experimental results in THUMOS14 [4] and FineAction [12] on varying tIoU thresholds. It is interesting to see that, compared to other approaches, *ActionSwitch generates a similar number of proposals to the ground truth action instances.*

Moreover, Table 5 presents the full ODAS performance. Although SimOn [17] outperforms ActionSwitch in the case of larger offsets, it is crucial to note that an excessively generous offset setting can unjustly favor models with an excessive number of action start detections. This is because, 1) like the mAP metric, the p-mAP metric does not effectively penalize false positives, and 2) wrong or arbitrary action start detections can be counted as true positives in an overly generous criterion. Since the prompt detection of action starts is the primary objective of the ODAS task, performance at low offset settings is critical, and our ActionSwitch model performs well despite its relatively lower detection

References

1. Caba Heilbron, F., Escorcia, V., Ghanem, B., Carlos Niebles, J.: Activitynet: A large-scale video benchmark for human activity understanding. In: IEEE/CVF International Conference on Computer Vision (ICCV) (2015) 8
2. Chung, J., Gulcehre, C., Cho, K., Bengio, Y.: Empirical evaluation of gated recurrent neural networks on sequence modeling. In: NIPS 2014 Workshop on Deep Learning, December 2014 (2014) 10
3. Damen, D., Doughty, H., Farinella, G.M., Furnari, A., Kazakos, E., Ma, J., Moltisanti, D., Munro, J., Perrett, T., Price, W., et al.: Rescaling egocentric vision. arXiv preprint arXiv:2006.13256 (2020) 8
4. 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) 3, 6, 8, 11, 12
5. 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: IEEE/CVF International Conference on Computer Vision (ICCV) (2021) 1, 11
6. Kim, Y.H., Kang, H., Kim, S.J.: A sliding window scheme for online temporal action localization. In: European Conference on Computer Vision (ECCV) (2022) 1, 3, 10, 11
7. Kuhn, H.W.: The hungarian method for the assignment problem. Naval research logistics quarterly (1955) 2, 3
8. Lin, T., Zhao, X., Su, H., Wang, C., Yang, M.: Bsn: Boundary sensitive network for temporal action proposal generation. In: European Conference on Computer Vision (ECCV) (2018) 3

9. Liu, X., Hu, Y., Bai, S., Ding, F., Bai, X., Torr, P.H.: Multi-shot temporal event localization: a benchmark. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (2021) 6
10. Liu, X., Hu, Y., Bai, S., Ding, F., Bai, X., Torr, P.H.: Multi-shot temporal event localization: a benchmark. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (2021) 11
11. 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) 11
12. Liu, Y., Wang, L., Wang, Y., Ma, X., Qiao, Y.: Fineaction: A fine-grained video dataset for temporal action localization. IEEE Transactions on Image Processing **31**, 6937–6950 (2022) 3, 6, 11, 12
13. Manolis Savva*, Abhishek Kadian*, Oleksandr Maksymets*, Zhao, Y., Wijmans, E., Jain, B., Straub, J., Liu, J., Koltun, V., Malik, J., Parikh, D., Batra, D.: Habitat: A Platform for Embodied AI Research. In: IEEE/CVF International Conference on Computer Vision (ICCV) (2019) 5
14. 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: European Conference on Computer Vision (ECCV) (2018) 8
15. Sigurdsson, G.A., Varol, G., Wang, X., Farhadi, A., Laptev, I., Gupta, A.: Hollywood in homes: Crowdsourcing data collection for activity understanding. In: European Conference on Computer Vision (ECCV) (2016) 8
16. Szot, A., Clegg, A., Undersander, E., Wijmans, E., Zhao, Y., Turner, J., Maestre, N., Mukadam, M., Chaplot, D., Maksymets, O., Gokaslan, A., Vondrus, V., Dharur, S., Meier, F., Galuba, W., Chang, A., Kira, Z., Koltun, V., Malik, J., Savva, M., Batra, D.: Habitat 2.0: Training home assistants to rearrange their habitat. In: Advances in Neural Information Processing Systems (NeurIPS) (2021) 5
17. Tang, T.N., Park, J., Kim, K., Sohn, K.: Simon: A simple framework for online temporal action localization. arXiv preprint arXiv:2211.04905 (2022) 1, 3, 11, 12
18. Tolstikhin, I.O., Houlsby, N., Kolesnikov, A., Beyer, L., Zhai, X., Unterthiner, T., Yung, J., Steiner, A., Keysers, D., Uszkoreit, J., et al.: Mlp-mixer: An all-mlp architecture for vision. Advances in Neural Information Processing Systems (NeurIPS) (2021) 10
19. Xu, M., Zhao, C., Rojas, D.S., Thabet, A., Ghanem, B.: G-tad: Sub-graph localization for temporal action detection. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (2020) 11
20. Yeung, S., Russakovsky, O., Jin, N., Andriluka, M., Mori, G., Fei-Fei, L.: Every moment counts: Dense detailed labeling of actions in complex videos. International Journal of Computer Vision (IJCV) (2018) 8
21. Zhang, C.L., Wu, J., Li, Y.: Actionformer: Localizing moments of actions with transformers. In: European Conference on Computer Vision (ECCV) (2022) 3, 4, 11
22. Zhao, H., Torralba, A., Torresani, L., Yan, Z.: Hacs: Human action clips and segments dataset for recognition and temporal localization. In: IEEE/CVF International Conference on Computer Vision (ICCV) (2019) 8