

# Refining Action Boundaries for One-stage Detection

Hanyuan Wang

Majid Mirmehdi

Dima Damen

Toby Perrett

Department of Computer Science, University of Bristol, Bristol, U.K.

{hanyuan.wang, dima.damen, toby.perrett}@bristol.ac.uk, majid@cs.bris.ac.uk

## Abstract

Current one-stage action detection methods, which simultaneously predict action boundaries and the corresponding class, do not estimate or use a measure of confidence in their boundary predictions, which can lead to inaccurate boundaries. We incorporate the estimation of boundary confidence into one-stage anchor-free detection, through an additional prediction head that predicts the refined boundaries with higher confidence. We obtain state-of-the-art performance on the challenging EPIC-KITCHENS-100 action detection as well as the standard THUMOS14 action detection benchmarks, and achieve improvement on the ActivityNet-1.3 benchmark.

## 1. Introduction

Current video understanding approaches [12, 6, 40] recognise actions on short, trimmed videos. These assume the boundaries of actions are already given, and thus focus on the class prediction problem solely. However, most real-life videos are untrimmed and contain irrelevant visual content. Temporal action detection aims to temporally locate the boundaries of actions and classify them in longer, unscripted and untrimmed videos [9, 14, 16, 4], which is crucial for video analysis.

Two-stage action detection approaches, such as [51, 22, 2, 20, 7, 37], were built on top of successful recognition models [12, 11, 45] and widely used as reference methods on simple action detection baselines [16, 4]. They first generate candidate proposals based on pre-defined sliding windows or matching locations with high probabilities scores, and then classify them to obtain the final predictions. However, such two-stage methods are inefficient for the wider variety of actions, action lengths and action/background densities found in longer untrimmed videos, since a large number of redundant candidate proposals are produced by

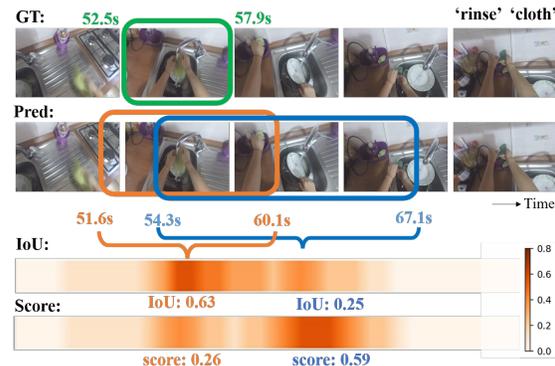


Figure 1. An illustration of the misalignment between the value of tIoU and classification scores of predicted proposals, caused by the absence of boundary confidences. Green denotes the ground truth, blue and orange denote predictions produced by ActionFormer [48]. Specifically, when the boundary confidence is not considered as the ranking metric, the prediction with a higher classification score but poor boundaries (blue) is chosen, rather than the prediction with better boundaries (orange).

sliding windows and location matching.

More recently, one-stage methods have been proposed, where the network simultaneously predicts the current action for each timestep and its associated boundaries [47, 19, 48]. In this paper, we show that these methods are missing the boundary confidence in proposal regression and evaluation. This can lead to imprecise localisation due to insufficient boundary information, especially in the case of actions of various lengths found in egocentric data, such as EPIC-KITCHENS[9]. An example of the action ‘rinse cloth’ from [9] is shown in Figure 1, where the prediction with a higher classification score has a lower overlap between boundaries and ground truth (blue), while the prediction with better boundaries has a lower classification score (orange). This is due to the absence of boundary confidences, resulting in poor regression and unreliable scores.

In this paper, we consider the extent of an action proposal and estimate the confidence of the *start* and *end* frames of the action segment, jointly. We supervise the

confidence from the relative distance between the estimated frame and the ground truth frame, for both the start and end boundaries of the action. This confidence information is leveraged to refine the boundaries of proposals which leads to state-of-the-art action detection results on EPIC-KITCHENS-100 [9] and THUMOS14 [16], and improvement on ActivityNet-1.3 [4].

In summary, we introduce a boundary head for one-stage anchor-free action detection which estimates boundary confidence scores based on relative distances. We obtain state-of-the-art results on EPIC-KITCHENS-100 and THUMOS14 action detection, using the same backbone as the current state-of-the-art. Notably, significant improvement is achieved on EPIC-KITCHENS-100, which indicates that our method performs well on complex actions of various lengths. Further, we provide detailed ablations, including investigating confidence scores and the effect of action lengths.

## 2. Related Work

Action detection methods can be grouped into two categories: two-stage and one-stage.

**Two-stage action detection:** Two-stage methods first generate a set of candidate proposals and then classify each proposal. They typically generate proposals by pre-defined sliding window and grouping temporal locations with high probabilities of being within an action [51] or close to a boundary [22, 2]. Action and boundary combinations can be selected based on high boundary confidence [20, 7], or a combination of separately calculated boundary and action scores [37]. This generation process can struggle when presented with sequences containing many dense actions of varying lengths, such as EPIC-KITCHENS-100 [9].

**One-stage action detection:** One-stage methods improve detection efficiency by simultaneously predicting action proposals and their associated classes. One approach is to generate candidate boundaries by modelling temporal relationships [13, 21, 29, 43, 24]. However, these methods rely on pre-defined anchors, causing them to struggle when presented with a wide range of action durations. Inspired by the DETR framework [5] for object detection, some works use learned action [27] or graph [30] queries as input to a transformer decoder. Whilst a promising direction, these methods are not suitable for long videos due to attention scaling issues. Anchor-free methods [47, 19, 48] simultaneously predict classification scores and a pair of relative distances to boundaries for each timestep.

Recently, ActionFormer [48] generated these predictions with a multi-scale transformer encoder to model both short- and long-range temporal dependencies, with simple classification and boundary regression heads, and achieved state-of-the-art results on a number of benchmarks. In this work, we adopt the same multi-scale transformer encoder and

pipeline as ActionFormer [48], but incorporate the ability to estimate boundary confidences.

## 3. Method

We first briefly review ActionFormer [48], and then introduce our novel boundary head, which is incorporated into ActionFormer to achieve better performance.

### 3.1. Overview of ActionFormer

ActionFormer first extracts a feature pyramid based on local self-attention, and then uses light-weight heads to simultaneously predict classification scores and a pair of relative distances to boundaries for each timestep.

**Transformer-based feature pyramid:** ActionFormer extracts features from an untrimmed sequence and passes them to a multi-scale transformer encoder [48] to construct a feature pyramid sequence. The feature pyramid sequence contains multiple resolutions for each timestep, which allows a single timestep to detect short and long actions.

**Prediction heads:** A classification head uses the feature pyramid sequence to predict action labels and classification scores for each timestep in multiple resolutions, and similarly, a regression head predicts relative distances to the predicted start and end boundary locations, for every timestep in the feature pyramid.

**Training and Inference:** The network is trained by minimizing the multi-part losses of the classification head and the regression head. For the classification head, a focal loss [23] is used to balance loss weights between easy and hard examples. For the regression head, it minimises the distance between the ground truth boundaries and the predicted boundaries using the GIoU loss [35]. At inference, they predict a pair of relative distances to boundaries and a classification score to give a proposal for each timestep across all pyramid levels. These candidate proposals are ranked by classification scores and further filtered to obtain the final outputs of actions.

### 3.2. Boundary Head

In ActionFormer, the regression head nominates where the boundaries are, without providing any confidence of the locations as boundaries. To address this, we compute the boundary confidence at the same time as the boundary location prediction. One approach could be using a separate branch to directly predict boundary confidence. However, this may lead to learning conflicts in the anchor-free pipeline, where the original network must learn the relative distances between the current temporal location and ground truth boundaries, rather than the confidence that the current location is a boundary (demonstrated in Section 4.3).

We design a simple but effective boundary head, which computes boundary probabilities via a confidence scaling,

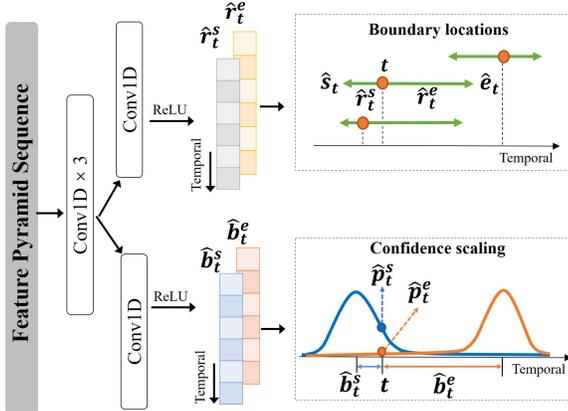


Figure 2. Overview of our boundary head. Taking in the feature pyramid sequence, two branches share most of the weights but have separate top layers. The first branch predicts start and end boundary **locations**. The second branch predicts boundary **confidence**. A confidence scaling encodes the assumption that a location  $t$  can be more confident about boundaries which are closer.

weighted by how close boundaries are. This approach, instead of directly producing boundary probabilities, means the network is better able to keep the consistency of two parallel branches in predicting and optimizing and make the learning process easier. As shown in Figure 2, the boundary head takes in the feature pyramid sequence, and contains two branches. These two branches share most of the weights, as there is shared information between the two tasks (e.g. high start confidence and small relative start offset are related), but they have separate top layers. The first branch predicts relative distances ( $\hat{r}_t^s$ ,  $\hat{r}_t^e$ ) to the start and end boundaries for each location  $t$  in the feature pyramid [19, 48]. Thus, the corresponding predicted start and end *locations* are obtained as  $\hat{s}_t = t - \hat{r}_t^s$  and  $\hat{e}_t = t + \hat{r}_t^e$ . The second branch predicts start and end confidence tokens ( $\hat{b}_t^s$ ,  $\hat{b}_t^e$ ) that are further fed into scaling processing (will be introduced next) to calculate the boundary *confidence* that location  $t$  is either an action start or an action end.

**Confidence scaling:** To assign a confidence that location  $t$  is a start ( $\hat{p}_t^s$ ) or end ( $\hat{p}_t^e$ ) boundary, the confidence tokens are weighted such that

$$\hat{p}_t^s = \exp\left(-(\hat{b}_t^s)^2/2\sigma^2\right) \text{ and } \hat{p}_t^e = \exp\left(-(\hat{b}_t^e)^2/2\sigma^2\right). \quad (1)$$

Here,  $\sigma$  is a scaling hyperparameter, determined experimentally in Section 4.3. Scaling in this manner encodes the assumption that a location  $t$  can be more confident about boundaries which are closer. This has previously been explored to improve online detection [18, 31], but not for offline use cases.

**Label assignment:** We require ground truth confidence values as supervision signals. We omit all locations where the GIoU between the ground truth and the predicted bound-

ary locations is less than a threshold  $\beta$ . We then adopt the BSN approach [22] and denote the ground truth start confidence for location  $t$  as  $p_t^s$ . Specifically, we define a region of length 1 timestep centered on location  $t$ , and calculate  $p_t^s$  as its overlap ratio to a region of length  $d/5$  centered on the ground truth start location, where  $d$  is the length of the ground truth action. This can be thought of as the value at location  $t$  of a start probability curve [22, 20, 50], and is visualised in Figure 2. In the case of multiple overlapping ground truth start locations, we take the maximum single overlap ratio. The ground truth end confidence  $p_t^e$  is calculated similarly.

**Training:** We minimise the difference between the ground truth and predicted confidences using the following two loss functions:

$$L_{conf}^s = \frac{1}{T} \sum_t (\hat{p}_t^s - p_t^s)^2 \text{ and } L_{conf}^e = \frac{1}{T} \sum_t (\hat{p}_t^e - p_t^e)^2, \quad (2)$$

where  $T$  is the total number of locations used for training from all levels of the feature pyramid sequence.

In an end-to-end manner, we incorporate our confidence losses into the total loss and optimize a weighted combination of all the losses:

$$L_{total} = L_{cls} + \gamma L_{GIoU} + \omega(L_{conf}^s + L_{conf}^e), \quad (3)$$

where  $L_{cls}$  and  $L_{GIoU}$  are losses for classification and regression and are the same as in ActionFormer.

### 3.3. Post-processing

The network produces a proposal from each location  $t$  in the feature pyramid containing a predicted start boundary  $\hat{s}_t$  and a predicted end boundary  $\hat{e}_t$ . It also gives an action  $\hat{a}_t$  and an action confidence score  $\hat{p}_t^a$  taken from the logits. We can obtain the boundary confidences for this proposal by looking up the start boundary confidence at location  $\hat{s}_t$  and the end boundary confidence at location  $\hat{e}_t$ . These confidences are  $\hat{p}_{\hat{s}_t}^s$  and  $\hat{p}_{\hat{e}_t}^e$ .

We first multiply the start and end confidences to get the boundary confidence, then combine it with the action confidence to give a single final confidence for the proposal:

$$\hat{c} = \hat{p}_t^a \sqrt{\hat{p}_{\hat{s}_t}^s \hat{p}_{\hat{e}_t}^e}, \quad (4)$$

where a square root is used to balance the contributions of boundary and action confidences (demonstrated in Section 4.3). Finally, we follow standard practice [22, 20, 37, 48] and suppress redundant proposals with Soft-NMS [3] to obtain a final set of  $M$  predictions  $\hat{\Phi} = \{(\hat{s}, \hat{e}, \hat{c}, \hat{a})_m\}_{m=1}^M$ .

## 4. Experiments

### 4.1. Setup

**Datasets:** We evaluate our method on three datasets: EPIC-KITCHENS-100 [9], ActivityNet-1.3 [4] and THU-

Method	Venue	Feature	mAP@IoU					
			0.1	0.2	0.3	0.4	0.5	Avg.
BMN [20, 9]	IJCV 2022	SF [12]	6.95	6.10	5.22	4.36	3.43	5.21
AGT [30]	arXiv 2021	I3D [6]	7.78	6.92	5.53	4.22	3.86	5.66
TSN [20, 15]	ICLR 2022	TSN [40]	10.24	9.61	8.94	7.96	6.79	8.71
OWL [34]	arXiv 2022	SF(A,V)[12, 17]	11.01	10.37	9.47	8.24	7.26	9.29
TAda2D [20, 15]	ICLR 2022	TAda2D [15]	15.15	14.32	13.59	12.18	10.65	13.18
AF [48]*	ECCV 2022	SF [12]	<u>18.02</u>	<u>17.41</u>	<u>16.44</u>	<u>15.17</u>	<u>13.23</u>	<u>16.05</u>
Ours	-	SF [12]	<b>19.19</b>	<b>18.61</b>	<b>17.47</b>	<b>16.30</b>	<b>14.33</b>	<b>17.18</b>

Table 1. Comparative results on the the EPIC-KITCHENS-100 validation set for the action task (i.e. predict both verb and noun). \*AF or ActionFormer only provides results on verb and noun detection separately, so we produce results by modifying it with our multitask classification head. **Bold** for best model and underline for second best.

MOS14 [16]. Two of the most widely used baselines for action detection are ActivityNet-1.3 [4] and THUMOS14 [16]. ActivityNet-1.3 contains 19,994 videos with 23,064 instances of 200 action classes, at an average of 1.5 instances per video. THUMOS14 consists of 413 videos with 38,690 instances of 20 action classes, and an average of 15.4 instances per video.

More recently, interest in wearable cameras has led to the collection and annotation of large-scale egocentric datasets, such as EPIC-KITCHENS-100 [9], which consists of 700 videos with 89,977 verb/noun action instances. There are 96 verb and 300 noun classes, with 4,053 complicated action instances (e.g. verb/noun pairs such as “pickup fork” rather than “cycling”), with much longer sequences containing an average of 128 action instances per video. They also contain a wider range of action durations, along with significant overlap between actions due to participants acting in a natural manner in familiar environments. These provide a much more difficult test for action detection methods [10].

**Evaluation Metrics:** Following the official settings, we use mean Average Precision (mAP) at different Intersection over Union (tIoU) thresholds to evaluate the performance of action detection. The mAP is the average precision across all action classes. On EPIC-KITCHENS-100, the tIoU thresholds are set to  $\{0.1, \dots, 0.5\}$  at step size of 0.1. On ActivityNet-1.3 they are  $\{0.5, 0.75, 0.95\}$ , and on THUMOS14 they are  $\{0.3, \dots, 0.7\}$  at step size of 0.1.

**Baselines:** We compare against state-of-the-art (SOTA) approaches on EPIC-KITCHENS-100 [20, 30, 40, 34, 15, 48], ActivityNet-1.3 and THUMOS14 [22, 20, 44, 2, 37, 32, 39, 7, 41, 42, 26, 51, 29, 27, 19, 8, 25] benchmarks. As ActionFormer only provides results and code for separate verb and noun detections (not combined action detection which is standard practice) on EPIC-KITCHENS-100, we modify it with our classification head to provide comparable results. We also provide separate verb and noun detection results of our own to compare.

**Implementation details:** For feature extraction and pyramid generation, we follow ActionFormer [48] for all datasets. Briefly, we use SlowFast [12] features for EPIC-KITCHENS-100, TSP [1] features for ActivityNet-1.3 and I3D [6] for THUMOS14. The transformer encoder gener-

ates a feature pyramid with 6 levels, with a level scaling factor of 2. The length of the first pyramid level is 2304 for EPIC-KITCHENS-100 and THUMOS14, and 768 for ActivityNet-1.3.

For label assignment and training, we find the ground truth action that location  $t$  is in and take its start  $s^*$  and end  $e^*$  boundaries. When multiple ground truth regions overlap with  $t$ , only the shortest ground truth region is used to make regression during training easier. Following [38, 49, 48], if a location  $t$  is not in an action, or is too close to a ground truth action boundary (i.e. not in the middle  $\alpha$  timesteps of an action), it is omitted from the loss calculation. The value of  $\alpha$  is set as 3. In Equation 1, we select  $T$  samples to train the network. Specifically, we set the GIoU threshold for rejecting proposals used to train boundary confidence prediction as  $\beta = 0.5$ . For the total loss in Equation 3, the weights are set as  $\gamma = 0.5$  and  $\omega = 0.5$ . These are the same for all datasets. On EPIC-KITCHENS-100, the classification loss weight for verb/noun is set as 0.5.

In the classification head, ActionFormer just returns logits and predicts actions for a single action task (as in ActivityNet-1.3 and THUMOS14). For the compound actions found in EPIC-KITCHENS-100, which are verb/noun pairs, it is not practical to return confidences for every possible verb and noun combination. Instead, we take the top- $v$  verb and top- $n$  noun predictions. The candidate actions are every combination of top- $v$  verbs and top- $n$  nouns, and their confidences are the verb and noun logits multiplied together. Our main results use  $v = 10$  and  $n = 30$ .

## 4.2. Results

**Main Results:** Tab. 1 shows the EPIC-KITCHENS-100 results, which is the most challenging dataset we use. Our method outperforms all previous work in all mAP@IoUs. Most importantly, it surpasses the current SOTA work, i.e. ActionFormer [48] modified with our verb/noun classification head. This establishes a direct comparison against the same architecture, but without boundary confidence.

Tab. 2 shows a comparison of noun detection and verb detection separately (i.e. separate training and testing runs for each). Although this is a simpler task and not the rec-

Task	Method	mAP@IoU					
		0.1	0.2	0.3	0.4	0.5	Avg.
Verb	G-TAD [44, 48]	12.1	11.0	9.4	8.1	6.5	9.4
	AF [48]	<u>26.6</u>	<u>25.6</u>	<u>24.4</u>	<u>22.4</u>	<u>18.3</u>	<u>23.4</u>
	Ours	<b>28.0</b>	<b>27.2</b>	<b>25.7</b>	<b>23.7</b>	<b>20.1</b>	<b>25.0</b>
Noun	G-TAD [44, 48]	11.0	10.0	8.6	7.0	5.4	8.4
	AF [48]	<u>25.5</u>	<u>24.3</u>	<u>22.6</u>	<u>20.3</u>	<u>16.6</u>	<u>21.9</u>
	Ours	<b>26.0</b>	<b>24.4</b>	<b>23.0</b>	<b>20.4</b>	<b>16.7</b>	<b>22.1</b>

Table 2. Comparative results for separate verb and noun models on the EPIC-KITCHENS-100 validation set. All methods use the same SlowFast features [12]. **Bold** for best model and underline for second best.

Method	Venue	ActivityNet-1.3					THUMOS14						
		Feature	0.5	0.75	0.95	Avg.	Feature	0.3	0.4	0.5	0.6	0.7	Avg.
BSN [22]	ECCV 18	TSN [40]	46.5	30.0	8.0	30.0	TSN [40]	53.5	45.0	36.9	28.4	20.0	36.8
BMN [20]	ICCV 19	TSN [40]	50.1	34.8	8.3	33.9	TSN [40]	56.0	47.4	38.8	29.7	20.5	38.5
G-TAD [44]	CVPR 20	TSP [1]	51.3	37.1	9.3	35.8	TSN [40]	54.5	47.6	40.2	30.8	23.4	39.3
BC-GNN [2]	ECCV 20	TSN [40]	50.6	34.8	9.4	34.3	TSN [40]	57.1	49.1	40.4	31.2	23.1	40.2
BSN++ [37]	AAAI 21	TSN [40]	51.3	35.7	8.3	34.9	TSN [40]	59.9	49.5	41.3	31.9	22.8	41.1
TCANet [32]	CVPR 21	SF [12]	54.3	39.1	8.4	37.6	TSN [40]	60.6	53.2	44.6	36.8	26.7	44.3
TVNet [39]	VISAPP 22	TSN [40]	51.4	35.0	<u>10.1</u>	34.6	TSN [40]	64.7	58.0	49.3	38.2	26.4	47.3
DCAN [7]	AAAI 22	TSN [40]	51.8	36.0	9.5	35.4	TSN [40]	68.2	62.7	54.1	43.9	<u>32.6</u>	52.3
RCL[41]	CVPR 22	TSP [1]	<u>55.2</u>	<u>39.0</u>	8.3	<u>37.7</u>	TSN [40]	70.1	62.3	52.9	42.7	30.7	51.7
RefactorNet [42]	CVPR 22	I3D [6]	<b>56.6</b>	<b>40.7</b>	7.4	<b>38.6</b>	I3D [6]	<u>70.7</u>	<u>65.4</u>	<u>58.6</u>	<u>47.0</u>	32.1	<u>54.8</u>
MUSES [46, 26]	AAAI 22	I3D [6]	53.2	36.2	<b>10.6</b>	35.5	I3D [6]	<b>71.5</b>	<b>67.0</b>	<b>60.0</b>	<b>48.9</b>	<b>33.0</b>	<b>56.1</b>
SSN [51]	ICCV 17	TS [36]	43.3	28.7	5.6	28.3	TS [36]	51.9	41.0	29.8	-	-	-
GTAN [29]	CVPR 19	P3D [33]	52.6	34.1	<u>8.9</u>	34.3	P3D [33]	57.8	47.2	38.8	-	-	-
TadTR [27]	TIP 22	I3D [6]	49.1	32.6	8.5	32.3	I3D [6]	62.4	57.4	49.2	37.8	26.3	46.6
AFSD [19]	CVPR 21	I3D [6]	52.4	35.3	6.5	34.4	I3D [6]	67.3	62.4	55.5	43.7	31.1	52.0
TALLFormer [8]	ECCV 22	SW [28]	<u>54.1</u>	36.2	7.9	35.6	SW [28]	76.0	-	63.2	-	34.5	59.2
MRBD [25]	CVPR 22	SF [12]	50.5	36.0	<b>10.8</b>	35.1	SF [12]	69.4	64.3	56.0	46.4	34.9	54.2
AF [48]	ECCV 22	TSP [1]	<u>54.1</u>	36.3	7.7	<u>36.0</u>	I3D [6]	75.5	72.5	65.6	56.6	42.7	62.6
AF [48] <sup>†</sup>	ECCV 22	TSP [1]	<b>54.2</b>	<u>36.9</u>	7.6	<u>36.0</u>	I3D [6]	82.1	<u>77.8</u>	<u>71.0</u>	<u>59.4</u>	<u>43.9</u>	<u>66.8</u>
Ours	-	TSP [1]	<u>54.1</u>	<b>37.3</b>	8.0	<b>36.1</b>	I3D [6]	<b>82.7</b>	<b>79.0</b>	<b>71.7</b>	<b>60.9</b>	<b>46.3</b>	<b>68.1</b>

Table 3. Comparative results on ActivityNet-1.3 and THUMOS14, grouped by two-stage methods (top) and one-stage methods (bottom). <sup>†</sup> means results from latest ActionFormer codebase. **Bold** for best model and underline for second best.

ommended evaluation metric, it allows us to compare our method with the reported results from ActionFormer, which we also outperform.

Tab. 3 shows results for ActivityNet-1.3 and THUMOS14. In general, two-stage methods (top) are strong on ActivityNet-1.3 as it is a simple test, with one action per video, but our method still outperforms other one-stage methods (bottom). On the more challenging THUMOS14 dataset with multiple actions per video, one-stage methods are better. Our method achieves SOTA results using the same features as the best two-stage and one-stage methods.

**Qualitative Results:** Figure 3 shows the qualitative illustrations on EPIC-KITCHENS-100 validation set. The global results (bottom two lines) indicate the model’s capability to effectively detect dense actions with varying classes and lengths. The local results (middle three lines) show that our method could predict boundaries closer to the ground truth than ActionFormer, demonstrating that predicting boundary confidence is critical for one-stage action detection. In addition, our method captures more hard ac-

tions with overlapping than ActionFormer, such as “close cupboard” in the 73rd second and “put box” in the 72th second. ActionFormer misses these actions most likely due to imprecise predictions that lack a measure of confidence, especially for actions with similar visual content (such as frames in the top line: “move board” and “put box”, “close cupboard” and “open cupboard”).

### 4.3. Ablations

**Effect of confidence scaling:** In Section 3.2, we assume two ways to produce the boundary confidence. The first is to directly produce boundary confidence from the top layers. The second is to produce confidence tokens and scale them to boundary confidence, which achieves better performance (see Table 4). The results demonstrate that designing with scaling can better keep the consistency of predicting and optimizing, and boost the performance.

**Boundary refinement:** To demonstrate that our boundary head can achieve more refined boundaries, we present the accumulative percentage of actions which correctly de-

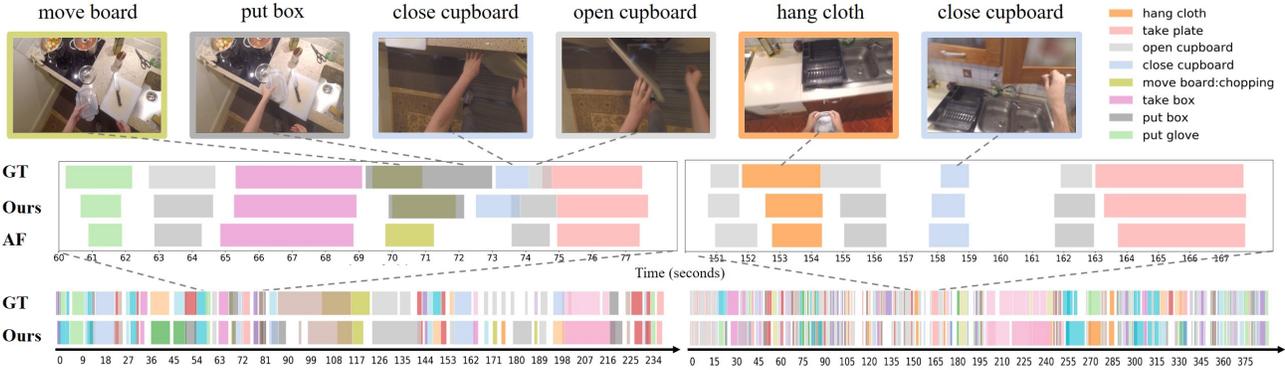


Figure 3. Qualitative results on the EPIC-KITCHENS-100 validation set. Ground truth and predictions are shown with colour-coded class labels (see legend). The bottom two lines are ground truth (GT) and our prediction (Ours) for the whole video sequence. The middle three lines show ground truth (GT), our prediction (Ours) and ActionFormer’s [48] prediction (AF) for a zoomed-in region. The top line shows visual content of selected frames. Results demonstrate that our method is good at dense actions and boundary refinement.

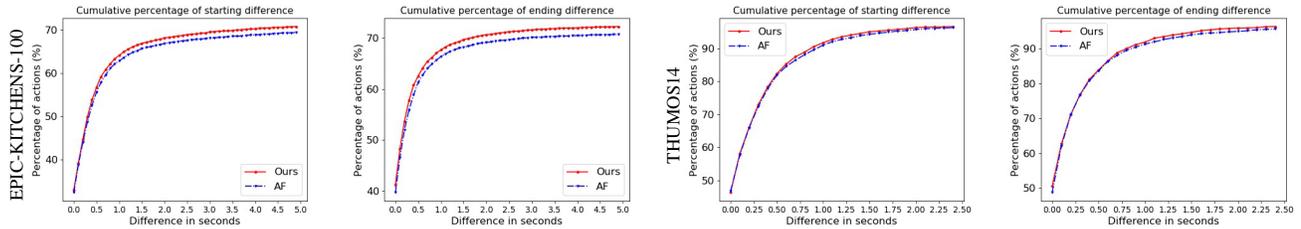


Figure 4. Accumulative percentage of actions which is correctly detected within  $x$  seconds difference from ground truth. Acceptance difference  $x$  is shown on the  $x$ -axis. Results are shown for starting and ending on EPIC-KITCHENS-100 (left) and THUMOS14 (right) dataset. Our model outperforms ActionFormer [48] in all cases.

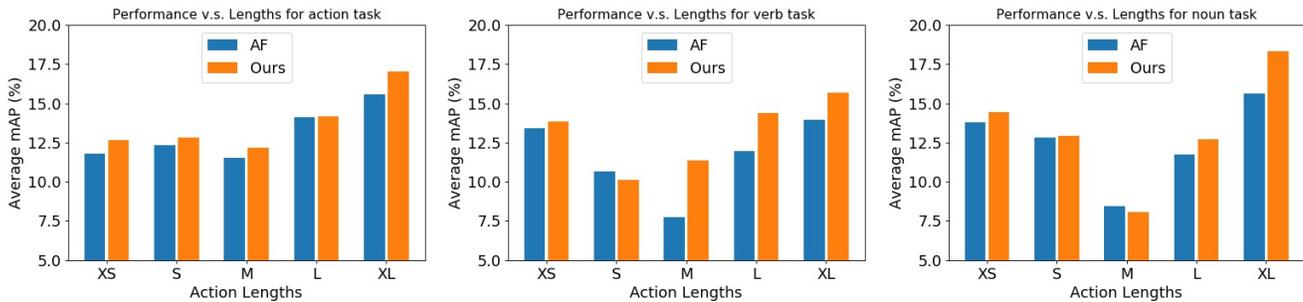


Figure 5. Comparing detection results to ActionFormer [48] at different action lengths on the validation set of EPIC-KITCHENS-100. We divide actions to five groups based on lengths (in seconds): XS (0, 2], S (2, 4], M (4, 6], L (6, 8], and XL (8, inf). Left: action task. Middle: verb task. Right: noun task.

tect within a certain threshold of the boundary error. We define the boundary error as temporal difference between ground truth start/end and predicted start/end, in seconds. As shown in Figure 4, our boundary head can detect actions with more precise boundaries compared to ActionFormer [48] on both the EPIC-KITCHENS-100 and THUMOS14 datasets. We observe a greater improvement achieved on the EPIC-KITCHENS-100 dataset, which contains actions with a wider range of lengths, demonstrating that our method is

more effective for more complex sequences.

**Effect of sigma:** The hyperparameter  $\sigma$  in Equation 1 determines the degree of scaling when weighting confidence tokens to output boundary confidence. Table 5 shows that the results change slightly with  $\sigma$  from 4 to 6 at a step size of 0.5, with the best results at  $\sigma = 5.5$ . In addition, the results show reasonable robustness to the change in  $\sigma$ .

**Confidence score combinations:** Our method produces confidence scores for its three outputs: action, start bound-

Scaling	Task	Val (mAP@IoU)					
		0.1	0.2	0.3	0.4	0.5	Avg.
✗	Verb	23.66	22.61	21.09	19.02	16.54	20.58
	Noun	22.92	21.68	20.30	18.64	16.07	19.92
	Action	18.19	17.63	16.59	15.38	13.64	16.29
✓	Verb	23.75	22.68	22	19.19	16.73	<b>20.71</b>
	Noun	23.58	22.40	21.03	19.27	16.39	<b>20.53</b>
	Action	19.19	18.61	17.47	16.30	14.33	<b>17.18</b>

Table 4. Effect of confidence tokens scaling. Without scaling (✗) means directly using the predicted confidence token as the boundary confidence. Scaling (✓) means scaling to boundary confidence using Equation 1. Results are shown for the action task in the validation set of EPIC-KITCHENS-100. **Bold** for best.

$\sigma$	Val (mAP@IoU)					
	0.1	0.2	0.3	0.4	0.5	Avg.
4	18.83	18.33	17.12	15.90	13.86	16.81
4.5	18.78	18.28	17.19	15.89	14.16	16.86
5	18.96	18.46	17.24	16.04	14.31	17.00
5.5	<b>19.19</b>	<b>18.61</b>	<b>17.47</b>	<b>16.30</b>	<b>14.33</b>	<b>17.18</b>
6	18.93	18.43	17.22	16.02	14.29	16.98

Table 5. Comparison of different values of hyper-parameters  $\sigma$  in Equation 1 of boundary head. Results are shown for the action task in the validation set of EPIC-KITCHENS-100. **Bold** for best.

ary and end boundary. At inference, just one confidence score is required for proposal suppression. Table 6 ablates how our three scores can be combined. We can see that the best performance is found when multiplying the action confidence by the square root of the product of the boundary confidences. Note that the boundary confidence on its own does not work, since it is hard to distinguish between whether a proposal is an action or background, and what action it is based just on the boundary.

**Effect of action length:** A successful action detection method should perform well over a wide range of action lengths. Figure 5 shows the average mAP on EPIC-KITCHENS-100 verb, noun and action of our method compared to ActionFormer [48] with our classification head, which demonstrates the effect of including boundary confidence. Although improvements are found at all action lengths, the largest improvement is on the longest actions. This is most likely due to longest actions being hard to regress given the long relative distances, but our boundary head helps to alleviate this issue because it uses the region around the boundary to compute the confidence.

**Multi-task Classification:** On EPIC-KITCHENS-100, unlike ActionFormer, our classification head handles compound verb/noun actions by only selecting the combinations of the top- $v$  predicted verbs and top- $n$  predicted nouns. Table 7 ablates this choice. In all cases,  $n = 3v$  as there are three times as many noun classes compared to verb classes. Clearly just relying on very low numbers causes a signif-

icant performance penalty, as the redundant proposal suppression relies on multiple action confidences. Performance is relatively stable above  $v = 5$  and  $n = 15$ . Our main results use  $v = 10$  and  $n = 30$ .

## 5. Conclusion

In this paper, we propose a novel boundary head and incorporate it into a one-stage anchor-free pipeline to estimate boundary confidence scores and produce refined boundaries for temporal action detection. Our method achieves state-of-the-art results on commonly used action detection benchmarks, including a significant improvement on the challenging EPIC-KITCHENS-100 dataset, which contains dense actions of various lengths. Detailed ablations show the benefits of incorporating boundary confidences and the effect of different parameters, establishing their importance in handling compound, egocentric actions. In future work, we aim to explore temporal action sequence contexts and multi-modality based on language models.

**Acknowledgement:** Funded by the University of Bristol and the China Scholarship Council.

## References

- [1] H. Alwassel, et al. Tsp: Temporally-sensitive pretraining of video encoders for localization tasks. In *ICCV*, 2021. 4, 5
- [2] Y. Bai, et al. Boundary content graph neural network for temporal action proposal generation. In *ECCV*, 2020. 1, 2, 4, 5
- [3] N. Bodla, et al. Soft-nms improving object detection with one line of code. In *ICCV*, 2017. 3
- [4] F. Caba Heilbron, et al. ActivityNet: A large-scale video benchmark for human activity understanding. In *CVPR*, 2015. 1, 2, 3, 4
- [5] N. Carion, et al. End-to-end object detection with transformers. In *ECCV*, 2020. 2
- [6] J. Carreira et al. Quo vadis, action recognition? a new model and the Kinetics dataset. In *CVPR*, 2017. 1, 4, 5
- [7] G. Chen, et al. Dcan: improving temporal action detection via dual context aggregation. In *AAAI*, 2022. 1, 2, 4, 5
- [8] F. Cheng et al. Tallformer: Temporal action localization with long-memory transformer. *arXiv preprint arXiv:2204.01680*, 2022. 4, 5
- [9] D. Damen, et al. Rescaling egocentric vision. *arXiv preprint arXiv:2006.13256*, 2020. 1, 2, 3, 4
- [10] D. Damen, et al. Epic-kitchens-100- 2021 challenges report. Report, 2021. 4
- [11] H. Duan, et al. Revisiting skeleton-based action recognition. In *CVPR*, 2022. 1
- [12] C. Feichtenhofer, et al. Slowfast networks for video recognition. In *ICCV*, 2019. 1, 4, 5
- [13] J. Gao, et al. Turn tap: Temporal unit regression network for temporal action proposals. In *ICCV*, 2017. 2
- [14] K. Grauman, et al. Ego4d: Around the world in 3,000 hours of egocentric video. In *CVPR*, 2022. 1

Combinations	Val (mAP@IoU)					
	0.1	0.2	0.3	0.4	0.5	Avg.
$\hat{p}_t^s * \hat{p}_t^e$	12.83	11.61	10.41	9.03	7.31	10.24
$\hat{p}_t^a$	18.74	18.16	17.23	15.95	13.90	16.80
$\hat{p}_t^a * \hat{p}_t^s$	19.02	18.43	17.34	16.00	14.04	16.96
$\hat{p}_t^a * \hat{p}_t^e$	19.12	18.57	<b>17.54</b>	16.19	14.23	17.13
$Avg.(\hat{p}_t^a, \hat{p}_t^s, \hat{p}_t^e)$	18.74	18.17	17.06	15.60	13.68	16.65
$\hat{p}_t^a * \hat{p}_t^s * \hat{p}_t^e$	19.18	18.60	17.46	16.27	14.28	17.16
$\hat{p}_t^a * \sqrt{\hat{p}_t^s * \hat{p}_t^e}$	<b>19.19</b>	<b>18.61</b>	17.47	<b>16.30</b>	<b>14.33</b>	<b>17.18</b>

Table 6. Comparison of different combinations of confidence scores. Results are shown for the action task in the validation set of EPIC-KITCHENS-100. **Bold** for best.

top-v	top-n	Val (mAP@IoU)					
		0.1	0.2	0.3	0.4	0.5	Avg.
1	3	17.05	16.47	15.54	14.53	12.67	15.25
2	6	18.33	17.82	16.82	15.75	14.00	16.54
5	15	19.16	18.57	17.40	16.23	14.24	17.12
10	30	<b>19.19</b>	<b>18.61</b>	<b>17.47</b>	<b>16.30</b>	14.33	<b>17.18</b>
15	45	19.16	18.57	17.44	16.27	14.30	17.15
20	60	18.88	18.35	17.32	16.23	14.36	17.03
97	300	18.89	18.36	17.33	16.23	<b>14.37</b>	17.04

Table 7. Comparison of different numbers of top-v, top-n for combining action labels. Results are shown for the action task on the validation set of EPIC-KITCHENS-100. **Bold** for best.

- [15] Z. Huang, et al. Tada! temporally-adaptive convolutions for video understanding. In *ICLR*, 2022. 4
- [16] Y. Jiang, et al. THUMOS challenge: Action recognition with a large number of classes. In *ECCV Workshop*, 2014. 1, 2, 4
- [17] E. Kazakos, et al. Slow-fast auditory streams for audio recognition. In *ICASSP*, 2021. 4
- [18] Y. Li, et al. Online human action detection using joint classification-regression recurrent neural networks. In *ECCV*, 2016. 3
- [19] C. Lin, et al. Learning salient boundary feature for anchor-free temporal action localization. In *CVPR*, 2021. 1, 2, 3, 4, 5
- [20] T. Lin, et al. BMN: Boundary-matching network for temporal action proposal generation. In *ICCV*, 2019. 1, 2, 3, 4, 5
- [21] T. Lin, et al. Single shot temporal action detection. In *Proceedings of the 25th ACM international conference on Multimedia*, 2017. 2
- [22] T. Lin, et al. BSN: Boundary sensitive network for temporal action proposal generation. In *ECCV*, 2018. 1, 2, 3, 4, 5
- [23] T.-Y. Lin, et al. Focal loss for dense object detection. In *ICCV*, 2017. 2
- [24] Q. Liu et al. Progressive boundary refinement network for temporal action detection. In *AAAI*, 2020. 2
- [25] X. Liu, et al. An empirical study of end-to-end temporal action detection. In *CVPR*, 2022. 4, 5
- [26] X. Liu, et al. Multi-shot temporal event localization: a benchmark. In *CVPR*, 2021. 4, 5
- [27] X. Liu, et al. End-to-end temporal action detection with transformer. *IEEE Transactions on Image Processing*, 2022. 2, 4, 5
- [28] Z. Liu, et al. Video swin transformer. In *CVPR*, 2022. 5
- [29] F. Long, et al. Gaussian temporal awareness networks for action localization. In *CVPR*, 2019. 2, 4, 5
- [30] M. Nawhal et al. Activity graph transformer for temporal action localization. *arXiv preprint arXiv:2101.08540*, 2021. 2, 4
- [31] T. Perrett et al. Ddlstm: dual-domain lstm for cross-dataset action recognition. In *CVPR*, 2019. 3
- [32] Z. Qing, et al. Temporal context aggregation network for temporal action proposal refinement. In *CVPR*, 2021. 4, 5
- [33] Z. Qiu, et al. Learning spatio-temporal representation with pseudo-3d residual networks. In *ICCV*, 2017. 5
- [34] M. Ramazanova, et al. Owl (observe, watch, listen): Localizing actions in egocentric video via audiovisual temporal context. *arXiv preprint arXiv:2202.04947*, 2022. 4
- [35] H. Rezatofighi, et al. Generalized intersection over union: A metric and a loss for bounding box regression. In *CVPR*, 2019. 2
- [36] K. Simonyan et al. Two-stream convolutional networks for action recognition in videos. In *NeurIPS*, 2014. 5
- [37] H. Su, et al. BSN++: Complementary boundary regressor with scale-balanced relation modeling for temporal action proposal generation. In *AAAI*, 2021. 1, 2, 3, 4, 5
- [38] Z. Tian, et al. Fcos: Fully convolutional one-stage object detection. In *ICCV*, 2019. 4
- [39] H. Wang, et al. Tvnet: Temporal voting network for action localization. *VISAPP*, 2022. 4, 5
- [40] L. Wang, et al. Temporal segment networks: Towards good practices for deep action recognition. In *ECCV*, 2016. 1, 4, 5
- [41] Q. Wang, et al. Rcl: Recurrent continuous localization for temporal action detection. In *CVPR*, 2022. 4, 5
- [42] K. Xia, et al. Learning to refactor action and co-occurrence features for temporal action localization. In *CVPR*, 2022. 4, 5
- [43] H. Xu, et al. R-c3d: Region convolutional 3d network for temporal activity detection. In *ICCV*, 2017. 2
- [44] M. Xu, et al. G-TAD: Sub-graph localization for temporal action detection. In *CVPR*, 2020. 4, 5
- [45] Y. Xu, et al. Cross-model pseudo-labeling for semi-supervised action recognition. In *CVPR*, 2022. 1
- [46] H. Yang, et al. Temporal action proposal generation with background constraint. In *AAAI*, 2022. 5
- [47] L. Yang, et al. Revisiting anchor mechanisms for temporal action localization. *IEEE Transactions on Image Processing*, 29, 2020. 1, 2
- [48] C. Zhang, et al. Actionformer: Localizing moments of actions with transformers. *arXiv preprint arXiv:2202.07925*, 2022. 1, 2, 3, 4, 5, 6, 7
- [49] S. Zhang, et al. Bridging the gap between anchor-based and anchor-free detection via adaptive training sample selection. In *CVPR*, 2020. 4
- [50] P. Zhao, et al. Bottom-up temporal action localization with mutual regularization. In *ECCV*, 2020. 3
- [51] Y. Zhao, et al. Temporal action detection with structured segment networks. In *ICCV*, 2017. 1, 2, 4, 5