A Sliding Window Scheme for Online Temporal Action Localization (Supplementary Material)

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

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

1 Self-attention of Decoder

The transformer decoder of our model makes the anchor features from the input feature queue. It takes the class, offset, and length information of the each anchor as the supervision, but the supervision does not provide the specific location of the anchors in the queue. To analyze how the decoder works, we visualize the attention weights of the transformer decoder.

From the THUMOS'14 settings of input queue size $L_q = 64$ and $K = 6$ anchors of size $\{4, 8, 16, 32, 48, 64\}$, we extract and stack the attention weights ($\mathbb{R}^{K \times L_q}$) of the decoder's last block for all input data in the test set. We use the average of the stacked attention weights for the visualization. The visualization map in Fig. 1 shows that the self-attention mechanism successfully finds the anchor area in the input feature queue. For the anchors with short length, the decoded representations highly depend on the recent input features. As the anchor length increases, the decoder considers the wide range of features and pay more attention to past features than the recent features.

Our model can generate the coarse-to-fine anchor representations by one step, that means the transformer decoder integrates the complex anchor encoding steps of the conventional anchor-based TAL methods into one component.

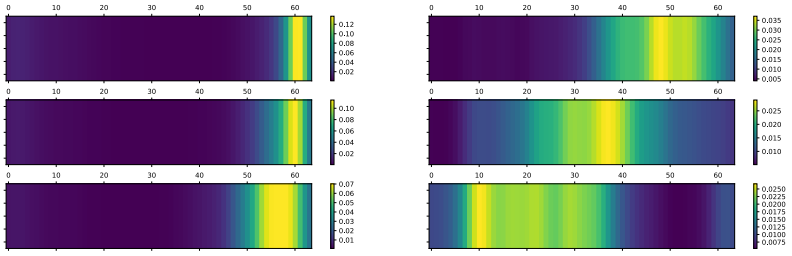

Fig. 1: The attention visualization map of decoded $K = 6$ anchors of size $\{4, 8, 16, 32, 48, 64\}$ on THUMOS'14. The bright area indicates where the decoder pays the attention in the input feature queue.

2 Details of Online Post-processing

Algorithm 1 Online post-processing

Input:
Newly generated action instance set Φ at timestamp t,
Proposal history queue q_h,
Previous action instance set Ψ_{t-1}.
Output:
Action instance set Ψ_t,
Proposal history queue q_h.
$\Psi_t \leftarrow \Psi_{t-1}$
$\Phi \leftarrow \text{NMS}(\Phi)$
for each action instance ϕ in Φ **do**
  $q_h[\phi.class]$.dequeue()
  $q_h[\phi.class]$.enqueue($\phi.score$)
  $s \leftarrow \text{SuppressNet}(q_h[\phi.class])$
  if $s > \theta_s$ **then**
    $\psi \leftarrow$ find ($\text{IoU}(\phi, \psi) > \theta_o$ and the same class) in Ψ_{t-1}
    if not exist ψ **then**
      Append ϕ in Ψ_t
    end if
  end if
end for

3 Action Proposal Analysis

For the analysis of the predicted action instances, we show detailed statistics on action length and classes.

First, Fig. 2 shows the per-class APs of our model on THUMOS'14. Our model performs over 50% APs on the half of classes and performs the best on *CleanAndJerk* and *JavelinThrow*. On the contrary, it shows the weakness on the *FrisbeeCatch* and *PoleVault* classes.

Second, we show the action length distribution of predicted action instances by our model and the ground truth in Fig 3. If tick and fragmentation are frequently occurred, the distribution of short actions are increase compared to the ground truth. In other cases, if merging are frequently occurred, the distribution of long actions are increase. Our model has the similar distribution on both short and long actions, and shows the robustness on three major issues of the OAD-base models.

4 Discussion

OAT outperforms the previous work (CAG-QIL) with a significant margin on the On-TAL task, and it is able to detect an action instance before the action

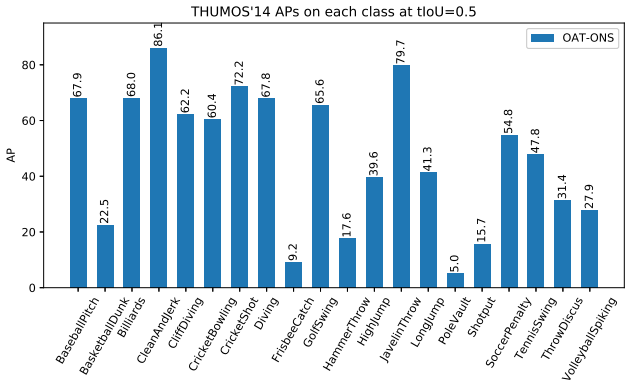

Fig. 2: The per-class APs at tIoU=0.5 on THUMOS'14.

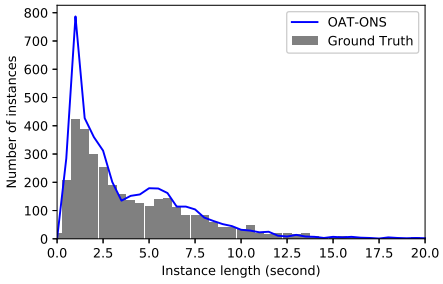

Fig. 3: The distribution of action instance length on THUMOS'14.

actually ends. However, the immediate detection of action start is not feasible in our model as the model focuses more on detecting the end. CAG-QIL, the OAD-base model that can easily extended to frame-level detection, can detect action starting points in the process of grouping action frames, and evaluates its results on the ODAS task. Our model works exclusively on On-TAL, and is incompatible to other online tasks such as OAD or ODAS. However, not all On-TAL scenario requires the prompt detection of action start. For example, the extraction of highlight candidate clips on live streams do not need the prompt detection of action start, but requires to detect an action instance before (at least, simultaneously) the action end. In the aforementioned case, our model is more suitable than the previous work.