Supplementary Material – HAT: History-Augmented Anchor Transformer for Online Temporal Action Localization

Sakib Reza, Yuexi Zhang, Mohsen Moghaddam, and Octavia Camps

Northeastern University, Boston, MA 02115, USA
{reza.s,zhang.yuex,mohsen,o.camps}@northeastern.edu

1 Method

1.1 Anchor Feature Generation

The process of generating anchor features from short-term windows involves two main stages. Initially, a transformer encoder, composed of N_e sequential blocks, is utilized to convert local attributes into a temporal contextual feature. Subsequently, this temporal feature is supplied to a transformer decoder, which incorporates learnable anchor tokens to synthesize features for each distinct anchor, similar to the approach in [7].

Each block within the transformer encoder, denoted as e, comprises a multihead self-attention mechanism and a feedforward network (FFN). The operational mechanism of a single encoder block can be represented as follows:

$$e(X) = \text{Norm}(X_{\text{attn}} + \text{FFN}(X_{\text{attn}})), \tag{1}$$

$$X_{\text{attn}} = \text{Norm}(X + \text{SelfAttn}(X)), \tag{2}$$

where FFN stands for the feedforward network, and SelfAttn denotes the multihead self-attention layer. As the model encompasses N_e encoder blocks, the final output is expressed as $S_{\text{enc}} = e_{N_e} \circ \ldots \circ e_1(S) \in \mathbb{R}^{L_s \times D}$, where L_s is the length of the short-term temporal context, and D represents the dimensionality of the features.

In the proposed model, each new video frame triggers the generation of action proposals, capitalizing on predefined anchor templates reflecting true event boundaries. Traditionally, this required the manual normalization of anchor segments of disparate lengths into a uniform feature format through different pooling strategies. However, following [7], we simplify this by directly deriving anchor representations, thereby eliminating the conventional requirements for manual size adjustment and reshaping. This is achieved by transforming the output sequence from the encoder, S_{enc}, which has a length of L_s, into a new sequence of representations of size M, using a standard transformer decoder.

The transformer decoder receives two types of input: a learnable anchor query $Q_{\text{anc}} \in \mathbb{R}^{M \times D}$, and the encoded short-term temporal context, S_{enc}. As it progresses through N_d decoder blocks, each containing multi-head cross-attention and feedforward network (FFN) layers, it produces an output $A \in \mathbb{R}^{M \times D}$.

1.2 Prediction Module and Online Post-Processing

Following [7], the prediction module consists of two distinct components: an action classifier and a boundary regressor. Both components use the same processed anchor feature representation A for their input. The function of the action classifier is to identify the specific action taking place within a given anchor segment and to decide whether this segment contains segments of non-action (such as background scenes). Conversely, the regressor is designed to compute two key measurements: the distance or offset between the end of the target action and the anchor segment's end, and the ratio of the duration of the target action to the total duration of the anchor segment. Each module consists of a neural network with two layers that produce distinct sets of output values. For the action classification part, this output is a set of classification scores, z_c, in the space $\mathbb{R}^{M\times(C+1)}$, which includes predictions for different action categories as well as the background category. Meanwhile, the boundary regression part generates a pair of scores, $\{z_o, z_l\}$, in the space $\mathbb{R}^{M\times 2}$, that represents the estimated offset and length ratio, respectively.

In addressing the issue of removing repetitive action proposals in video streams without relying on future data, we follow a online post-processing method described in the referenced work [7]. This method involves a neural network that selects the most appropriate action proposal at each time step, akin to Non-Maximum Suppression (NMS). The approach involves generating M proposals for each frame, eliminating those associated with the background, and executing NMS on the outcomes of each single frame. The confidence scores are maintained in a history buffer q_h with dimensions $[0,1]^{Ls\times C}$, which affects the network's probability of NMS selection within the range $[0,1]^C$. If the probability surpasses the preset threshold θ_s, the proposal is included in the final set of instances Ψ, and any following proposals that are similar are disregarded. The training process requires constructing input tables based on the main model's confidence scores and the actual labels derived from NMS outcomes, with the network being optimized via binary cross-entropy loss.

1.3 Training Strategy and Loss Function

Adaptive Focal Loss In the video action understanding datasets, a common issue is the imbalance between action samples and background samples, with the latter usually dominating. The focal loss was used as a solution to this imbalance by assigning varying levels of emphasis to different instances based on their classification confidence [2]. Specifically, it differentiates between hard-to-classify action instances and background instances by applying distinct focal factors, thereby addressing the imbalance between action and background classes.

However, this approach does not address the potential imbalance among different action classes themselves. In response, we introduce a modified version of the focal loss, Adaptive Focal Loss (AFL), tailored for OnTAL. This new loss function dynamically adjusts the focal factor for each action class during training, informed by the gradients, to better manage class imbalance across various action categories. The Adaptive Focal Loss is formulated as follows:

$$AFL(p_i, y_i) = \sum_{j=0}^{C} -y_i^j (1 - p_i^j)^{\lambda^j} \log(p_i^j), \tag{3}$$

where p_i is the predicted probability for the i^{th} anchor, y_i is the corresponding ground truth label, and λ^j is the class-specific focal factor for the j^{th} action category. This class-specific focal factor, λ^j, is defined as:

$$\lambda^j = \begin{cases} \lambda_b, & \text{for } j = C \\ \lambda_b + \lambda_f^j, & \text{otherwise} \end{cases}, \tag{4}$$

$$\lambda_f^j = s(1 - r^j), \tag{5}$$

where λ_b is the base focal factor, and λ_f^j is an adaptive component for the foreground classes, with r^j being the mapped ratio of accumulated gradients for positive versus negative samples for the j^{th} category. The scaling factor s adjusts the influence of λ_f^j. The value of r^j is adjusted so that it ranges only from 0 to 1. The computation of r^j proceeds as follows:

$$r^j = f(g^j), \tag{6}$$

$$g^j = \frac{g_{\text{pos}}^j}{g_{\text{neg}}^j}, \tag{7}$$

$$f(x) = \frac{1}{1 + e^{-(n(x)-\mu)}}, \tag{8}$$

where the terms g_{pos}^j and g_{neg}^j represent the gradients calculated from positive and negative samples, respectively, as used in [8]. $f(x)$ is the mapping function and $n(x)$ is the min-max normalization operation [11]. μ is the mean of all the normalized gradient ratios in the batch. The value of r^j approaches 1 for categories with less imbalance, thereby reducing the additional focal impact, and approaches 0 for categories with greater imbalance, thereby increasing the additional focal impact.

When setting values for equation 4, we empirically found that setting the value of $s = 2\lambda_b$ works well. For instance, in an average case (i.e., the relative imbalance is closer to the batch's mean), if we want to set $\lambda_j = 0.025$ for the background class ($j = C$) and $\lambda_j = 0.05$ for all other (foreground) classes ($j \neq C$), we should set $\lambda_b = 0.025$ and $s = 0.05$. With this setup, when the normalized gradient ratio $n(g^j)$ for a class is around the batch's mean, the adjusted gradient ratio r^j will be approximately 0.5. This makes the λ_j value for foreground classes about 0.05, as intended. If a foreground class is significantly more or less imbalanced compared to the mean, λ_j will adjust accordingly, becoming either larger or smaller than 0.05.

Final Loss In the training process, we guide the output of the prediction module using pre-defined anchors and the output of the action anticipation head using a short-term window. Specifically, for the prediction module, we establish anchors whose end points are aligned with the most recent input frame, creating M different lengths of anchors (such as $\{L_s/8, L_s/4, L_s/2, L_s\}$). We then assess the overlap between these anchors and the actual ground truth instances by calculating the Intersection-over-Union (IoU) for each pair. An anchor is considered a match to a ground truth if their IoU exceeds the defined matching threshold θ_m. Once matched, we compute the losses for the prediction module based on these associations. Separately, for the action anticipation head, we assess its ability to predict actions occurring within the short-term window and compute its loss based on this assessment. The methodology for calculating the losses in our model is detailed below:

$$\mathcal{L}_c = \sum_{i=1}^{K} \text{AFL}(\text{softmax}(z_{c,i}), y_{c,i}), \tag{9}$$

$$\mathcal{L}_o = \sum_{i=1}^{K} \text{L1}\left(z_{o,i} - \frac{y_{e,i} - a_{e,i}}{a_{l,i}}\right), \tag{10}$$

$$\mathcal{L}_l = \sum_{i=1}^{K} \text{L1}\left(z_{l,i} - \log\frac{y_{l,i}}{a_{l,i}}\right), \tag{11}$$

$$\mathcal{L}_a = \text{AFL}(\text{softmax}(z_{c,i}), y_{c,i}), \tag{12}$$

$$\mathcal{L} = \alpha\mathcal{L}_c + \beta(\mathcal{L}_o + \mathcal{L}_l) + \gamma\mathcal{L}_a, \tag{13}$$

Here, we utilize adaptive focal loss (AFL) for classifying anchor segments and apply L1 loss for regression, where y_c, y_e, y_l, a_e, and a_l represent actual class, true action end, true action length, anchor segment end, and anchor segment length, respectively. The total loss is determined by combining various losses using weighted coefficients. For prediction module, anchors meeting specific criteria are marked as foreground and undergo boundary regression; otherwise, they are marked as background, skipping regression. The overall loss is calculated combining different loss with weighted co-efficients $\alpha, \beta,$ and γ. Proposals are generated by fixing anchor ends to the latest frame, ensuring early actions are classified as background, while middle and end actions, aligned with true actions, allow for accurate boundary predictions.

During inference, our framework generates M action proposals $\{(s_i, e_i, c_i)\}_{i=1}^{M}$ for every timestep. These proposals are defined as follows:

$$c_i = \text{softmax}(\text{z}_{c,i}), \tag{14}$$

$$e_i = a_{o,i} + a_{l,i} \cdot \text{z}_{e,i}, \tag{15}$$

$$s_i = e_i - a_{l,i} \cdot \exp(\text{z}_{l,i}). \tag{16}$$

Note that in this inference process, the action anticipation head from the history module is not used and should be ignored.

2 Experiments

2.1 Dataset

Although the primary details about the dataset have already been presented in the main paper, here are some further insights and specifics regarding the datasets provided.

EGTEA [9] We extracted features using the I3D model [1], which was pretrained on the Kinetics dataset. These features were processed at their original 24 FPS frame rate with a stride of 12, leading to two feature vectors being generated every second. We adopted a four-fold cross-validation approach for training and evaluation, similar to [5]. For evaluation purposes, we employed the metric mAP@[0.1:0.1:0.5] and also reported the average mAP score. The action annotations in this dataset were solely based on verb (action) annotations.

EPIC-Kitchens-100 [3] Feature extraction for this dataset was performed using the SlowFast model [4], which had been pretrained on the EPIC Kitchens 100 training set specifically for action classification. This process involved the use of 32-frame clips at a 30 FPS frame rate and a stride of 16 frames, resulting in one feature vector approximately every 0.5333 seconds. Our model underwent training on the training set and was subsequently evaluated on the validation set. The evaluation metric utilized was mAP@[0.1:0.1:0.5], with the average mAP being reported in alignment with previous standards [3, 14]. Actions in this dataset are identified by a combination of a verb (action) and a noun (object), though our analysis solely focused on verb (action) annotations to maintain consistency with other OnTAL research.

THUMOS'14 [6] This dataset comprises 413 untrimmed videos spanning 20 action categories, divided into validation (200 videos) and test (213 videos) sets. Consistent with established practices, we used the validation set for training purposes and reported our findings on the test set. Feature extraction was conducted using the two-stream TSN model [12], pretrained on the Kinetics dataset [1], in accordance with prior methodologies. The evaluation metric applied was mAP@[0.3:0.1:0.7].

MUSES [10] For MUSES, we utilized the I3D [1] features that were officially provided. The training and testing phases were conducted using the original train-test subsets designated for the dataset. The evaluation followed a similar framework to that of THUMOS'14, using mAP@[0.3:0.1:0.7] as the metric and reporting the average mAP.

2.2 Additional Model Analysis

History Length Although adjusting the history length in our model is not a principal focus of our study, we conducted an analysis to illustrate the impact of history length on model performance using the EGTEA dataset (split-1), with findings presented in Table 1. The analysis reveals an absence of definitive trends in performance across various history lengths; however, it indicates that increasing the history length tends to enhance performance up to a certain threshold. For the EGTEA dataset, an optimal history length of 24 seconds was identified.

Table 1: History length analysis of HAT on EGTEA dataset (split-1) across various tIOU thresholds.

History Length L_h (sec.)	0.1	0.2	0.3	0.4	0.5	Avg.
0	24.2	22.7	20.5	16.3	10.9	18.9
8	25.0	23.1	19.7	16.1	12.0	19.2
16	26.5	24.5	21.6	16.7	11.7	20.2
24	27.3	**25.3**	21.6	**16.9**	**12.8**	**20.8**
32	27.0	24.5	21.1	16.6	11.7	20.2
40	**27.5**	25.2	**22.0**	16.6	11.8	20.6

Furthermore, we compared the impact of history length between a procedural egocentric (PREGO) dataset (i.e., EGTEA) and a non-PREGO dataset (i.e., THUMOS), as shown in Figure 1. The results indicate that in the PREGO scenario, increasing the history length positively impacts performance until it reaches a saturation point. Conversely, in the non-PREGO scenario, increasing the history length does not significantly affect performance.

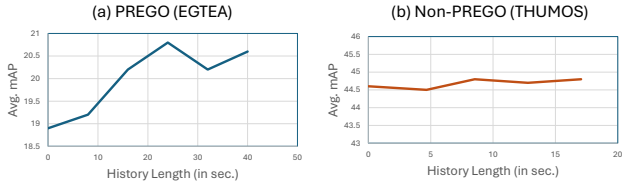

Fig. 1: Impact of history length comparison

History Integration Method Prior research [13] shows that the transformer decoder is the most effective method for combining historical features with current features. Nevertheless, we conducted an analysis to compare various approaches, considering our unique context for integrating historical and anchor features. Table 2 presents the comparison among these methods, demonstrating that the transformer decoder outperforms the alternatives.

Table 2: Comparison of History Integration Methods in History-Augmented Anchor Module on EGTEA (Split-1).

Method	0.1	0.2	0.3	0.4	0.5	Avg.
w/ Average Pooling	24.6	23.0	20.4	16.5	11.7	19.3
w/ Concatenation	25.3	23.0	20.3	16.7	12.1	19.5
w/ Temporal NLB (Cross-Attn.) [13]	26.8	24.7	21.2	**17.0**	12.0	20.3
Ours (w/ Trans. Decoder)	**27.3**	**25.3**	**21.6**	16.9	**12.8**	**20.8**

Early Detected Time Similar to the baseline OAT model [7], our proposed HAT model is also capable of predicting an action instance before the action is completed. While we made no specific efforts to enhance the responsiveness of our model, we conducted an evaluation to determine if incorporating long-term historical data impacts its responsiveness. To assess this, we used the Average Early Detection Time (AEDT) metric [7] for comparison with the baseline. According to the results presented in Table 3, our model exhibits performance on par with the baseline in terms of early detection. On average, both models are able to detect actions approximately 1 second before they conclude. This suggests that the integration of long-term history does not significantly affect the model's promptness.

Table 3: Average Early Detected Time (AEDT) comparison of our HAT model with the baseline OAT model across various tIOU thresholds.

Model	0.3	0.4	0.5	0.6	0.7	Avg.
OAT [7]	**-0.96**	**-1.01**	**-1.02**	-1.00	**-1.03**	**-1.01**
HAT (Ours)	-0.83	-0.90	-1.01	**-1.06**	**-1.03**	-0.97

References

1. Carreira, J., Zisserman, A.: Quo vadis, action recognition? a new model and the kinetics dataset. In: proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 6299–6308 (2017)
2. Chen, J., Mittal, G., Yu, Y., Kong, Y., Chen, M.: Gatehub: Gated history unit with background suppression for online action detection. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 19925–19934 (2022)
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: Collection, pipeline and challenges for epic-kitchens-100. International Journal of Computer Vision pp. 1–23 (2022)

4. Feichtenhofer, C., Fan, H., Malik, J., He, K.: Slowfast networks for video recognition. In: Proceedings of the IEEE/CVF international conference on computer vision. pp. 6202–6211 (2019)
5. Huang, Y., Sugano, Y., Sato, Y.: Improving action segmentation via graph-based temporal reasoning. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 14024–14034 (2020)
6. Idrees, H., Zamir, A.R., Jiang, Y.G., Gorban, A., Laptev, I., Sukthankar, R., Shah, M.: The thumos challenge on action recognition for videos "in the wild". Computer Vision and Image Understanding **155**, 1–23 (2017)
7. Kim, Y.H., Kang, H., Kim, S.J.: A sliding window scheme for online temporal action localization. In: European Conference on Computer Vision. pp. 653–669. Springer (2022)
8. Li, B., Yao, Y., Tan, J., Zhang, G., Yu, F., Lu, J., Luo, Y.: Equalized focal loss for dense long-tailed object detection. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 6990–6999 (2022)
9. Li, Y., Liu, M., Rehg, J.M.: In the eye of beholder: Joint learning of gaze and actions in first person video. In: Proceedings of the European conference on computer vision (ECCV). pp. 619–635 (2018)
10. Liu, X., Hu, Y., Bai, S., Ding, F., Bai, X., Torr, P.H.: Multi-shot temporal event localization: a benchmark. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 12596–12606 (2021)
11. Patro, S., Sahu, K.K.: Normalization: A preprocessing stage. arXiv preprint arXiv:1503.06462 (2015)
12. 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: European conference on computer vision. pp. 20–36. Springer (2016)
13. Xu, M., Xiong, Y., Chen, H., Li, X., Xia, W., Tu, Z., Soatto, S.: Long short-term transformer for online action detection. Advances in Neural Information Processing Systems **34**, 1086–1099 (2021)
14. Zhang, C.L., Wu, J., Li, Y.: Actionformer: Localizing moments of actions with transformers. In: European Conference on Computer Vision. pp. 492–510. Springer (2022)