Supplementary Materials for ActionFormer: Localizing Moments of Actions with Transformers

Chen-Lin Zhang*[1,2], Jianxin Wu[1], and Yin Li[3]

[1] State Key Laboratory for Novel Software Technology, Nanjing University, China
[2] 4Paradigm Inc, Beijing, China
[3] University of Wisconsin-Madison, USA
{zclnjucs, wujx2001}@gmail.com yin.li@wisc.edu

This supplement provides further details of the main paper. We describe (1) additional ablation experiments (Sec. A); (2) further error analysis of our results (Sec. B); (3) implementation details and how to reproduce our results (Sec. C); (4) additional visualizations of our results (Sec. D); and (5) limitation of our approach (Sec. E). For sections, figures, tables, and equations, we use numbers (*e.g*, Sec. 1) to refer to the main paper and capital letters (*e.g*, Sec. A) to refer to this supplement. We hope that this document will complement our main paper.

A Additional Ablation Experiments

Here we present additional ablation experiments, as mentioned in Sec. 4.4 of the main paper. These are omitted from the main paper due to lack of space. All experiments are reported on THUMOS14, consistent with our ablation experiments in the main paper. We follow our best design and use a local window size=19 with layer norm, center sampling, and score fusion enabled.

Loss Weight. We provide additional ablation on the loss weight λ_{reg} in Eq. 7. Specifically, we varied the loss weight $\lambda_{reg} \in [0.2, 0.5, 1, 2, 5]$, retrained the model, and reported the mAP scores. The results are presented in Table A. For a large range of λ_{reg}, our model has quite stable results with a maximum gap of 1.4% in average mAP. $\lambda_{reg} = 1$ yields the best results, as we used in all our experiments.

Maximum Input Sequence Length during Training. A possible explanation of our superior results is that our model might benefit from training using a long sequence (2304 time steps as in our previous experiments). Here we examine the effects of maximum input sequence length during training. Table B reports mAP scores for different training sequence lengths. The results of our model remain fairly consistent even with much shorter input sequence length. Note that when truncating an input sequence, our training scheme is equal to training with sliding windows as in [17]. The differences are (1) the windows are dynamically sampled rather than pre-generated; (2) windows without foreground actions are removed. When using a input seqence length of 512, similar to what

* Work was done when visiting UW Madison.

Table A. Ablation study on loss weight. We report mAP at tIoU=0.5 and 0.7, and the average mAP in [0.3 : 0.1 : 0.7] on THUMOS14 by varying the loss weight λ_{reg} in Eq. 7.

Method	λ_{reg}	0.5	0.7	Avg
Ours	0.2	69.7	40.9	65.6
Ours	0.5	**71.3**	42.4	66.7
Ours	1	71.0	**43.9**	**66.8**
Ours	2	69.5	**43.9**	66.2
Ours	5	69.1	43.0	65.4

was considered in [17] (512), our method only has a minor drop in average mAP (-1.1%) and significantly outperforms [17].

Table B. Ablation study on maximum input sequence length during training. We report mAP at tIoU=0.5 and 0.7, and the average mAP in [0.3 : 0.1 : 0.7] on THUMOS14 by varying the maximum input length T_{max} for training.

Method	T_{max}	0.5	0.7	Avg
Ours	576	69.6	42.5	65.7
Ours	1152	**71.0**	42.7	66.3
Ours	2304	**71.0**	**43.9**	**66.8**

Temporal Feature Resolution. Some of the previous works considered video features with lower temporal resolution. For example, a feature stride of 8 was used by PGCN [18] and ContextLoc [20]. To understand the effects of temporal feature resolution, we downsample our input I3D features and study the performance variation when using different feature strides. Table C report the results. When using a lower resolution (stride=8), the results of our model only drop slightly (-0.5% in average mAP). Further reducing the resolution (*e.g*, stride=16) leads to larger performance degradation, yet our results remains favourable.

Table C. Ablation study on temporal feature resolution. We report mAP at tIoU=0.5 and 0.7, and the average mAP in [0.3 : 0.1 : 0.7] on THUMOS14 by varying the feature stride.

Method	stride	0.5	0.7	Avg
Ours	4	**71.0**	**43.9**	**66.8**
Ours	8	69.8	**43.9**	66.3
Ours	16	65.8	38.4	61.9

B Further Error Analyses

We present further analyses of our results on THUMOS14. These analyses are obtained using the tool provided by [1]. We refer the readers to [1] for more details.

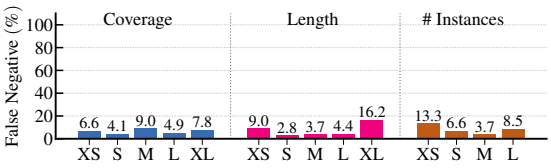

Fig. A. False negative (FN) profiling of our results on THUMOS14 using [1]. This figure shows the FN rates under different video contents. From this figure we can find that our model will suffer from extra short or extra long instances. Also, our model will suffer from video inputs which have a large number of action instances.

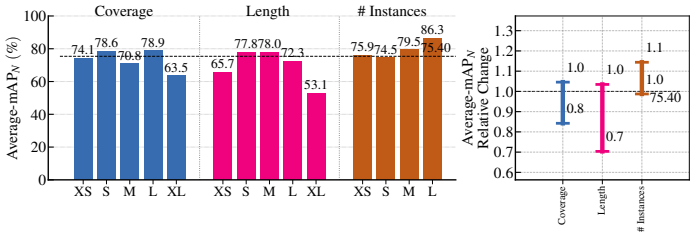

Fig. B. Sensitive analysis of our results on THUMOS14 using [1]. *Left*: normalized mAP at tIoU=0.5 under different video contents. *Right*: The relative normalized mAP change at tIoU=0.5 with respect to different characteristics of the ground truth instances.

Metrics. In [1], several characteristic metrics were defined given a dataset (*e.g* THUMOS14), including coverage, length, and the number of instances. Specifically, coverage presents the relative length of the actions (compared to the whole video), categorized into five bins: Extra Small (XS: (0, 0.02]), Small (S: (0.02, 0.04]), Medium (M: (0.04, 0.06]), Large (L:(0.06, 0.08]), and Extra Large (XL: (0.08, 1.0]). Length denotes the absolute length (in seconds) of actions, organized into five length groups: Extra Small (XS: (0, 3]), Small (S: (3, 6]), Medium (M: (6, 12]), Long (L: (12, 18]), and Extra Long (XL: > 18). Moreover, number of instances refers to the total count of instances (from the same class) in a video. This number is further divided into four parts, including Extra Small (XS: 1); Small (S: [2, 40]); Medium (M: [40, 80]); Large (L: > 80).

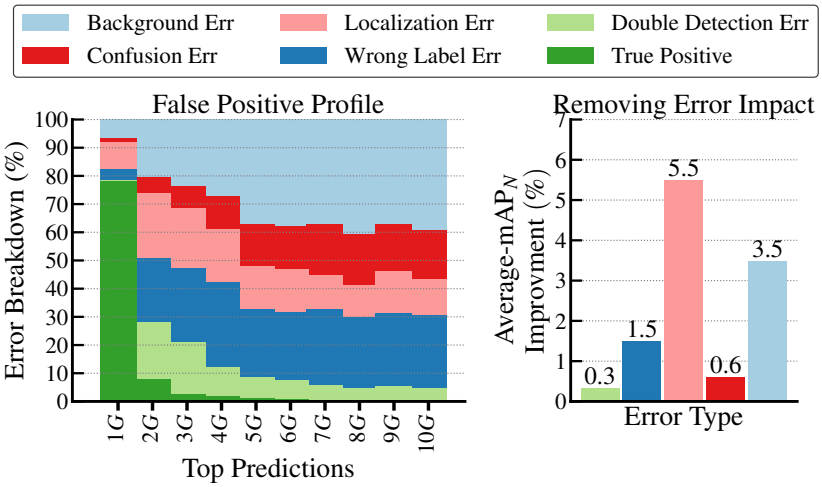

Fig. C. False positive (FP) profiling of our results on THUMOS14 using [1]. *Left*: FP error breakdown when considering the predictions for the top-10 ground-truth (G) instances. *Right*: The impact of error types. Localization error and background confusion are the top two error types.

Results and Analyses. Fig. A presents the false negative profiling. In Fig. A, we breakdown the false negative rates under the different coverage, length, and the number of instances. Our results have similar false negative rates across different coverage categories, yet have much higher false negative rates on action instances that are either very shot or very long (length), and on videos that contains many action instances (#instances). These action instances and videos are naturally more challenging.

Fig. B presents the sensitivity analysis of our results, *i.e*, normalized mAP at tIoU=0.5 under different characteristic metrics (left) and the variance of mAP across categories (right). Our model performs better on simple context scenarios, including XS/S/M/L coverage, S/M length and XS #instances, and worse on more complicated scenarios. The trend is similar to the false negative profiling in Fig. A. Moreover, our model is robust across different categories in coverage, length and #instances with small variances.

C Implementation Details

We now present implementation details including the network architecture, training and inference. Further details can be found in our code.

Network Architecture. We present our network architecture in Table D, as described in Sec. 3. In the ablation study (Sec. 4), we also considered a baseline that replaces the Transformer Units in Table D with convolution blocks, following the design of a bottleneck block in ResNet [7]. Specifically, a stack

Table D. The architecture of our model. Our network consists of (1) a Transformer encoder (first row block) and (2) a lightweight convolutional decoder with the classification / regression heads (last row block). For each layer, we list the layer name, layer parameters, the input to the layer, and the output feature size. We also include its regression range (in seconds for THUMOS14 and EPIC-Kitchens 100 and in number of time steps for ActivityNet-1.3). For convolutional layers, k is the kernel size of 1D convolutions and s is the stride, and c_i, c_o is the input and output feature channel, respectively. For Transformer Unit, ds is the downsampling ratio. T is the temporal length of input sequence and D is the input feature dimension. For classification head, the output dimension is the number of action categories. For regression head, the output dimension is 2, *i.e*, distances to action onset and offset.

	Name	Layer	Input	Output Size (T × D)	Regression Range
encoder	input clip	-	-	T×D	-
	projection1	conv k=3, s=1 (c_i = D, c_o = 512)	input clip	T × 512	-
	projection2	conv k=3, s=1 (c_i = 512, c_o = 512)	projection1	T × 512	-
	transformer0	Transformer Unit, ds=1	projection2	T × 512	-
	transformer1	Transformer Unit, ds=1	transformer0	T × 512	[0, 4)
	transformer2	Transformer Unit, ds=2	transformer1	T/2 ×512	[4, 8)
	transformer3	Transformer Unit, ds=2	transformer2	T/4 ×512	[8, 16)
	transformer4	Transformer Unit, ds=2	transformer3	T/8 ×512	[16, 32)
	transformer5	Transformer Unit, ds=2	transformer4	T/16 ×512	[32, 64)
	transformer6	Transformer Unit, ds=2	transformer5	T/32 ×512	[64, +∞)
decoder (heads)	cls / reg nets	conv k=3, s=1 (c_i = 512, c_o = 512)	transformer1,...,transformer6	[T/32×512,..., T×512]	-
		conv k=3, s=1 (c_i = 512, c_o = 512)	transformer1,...,transformer6	[T/32×512,..., T×512]	-
		conv k=3, s=1 (c_i = 512, c_o = output)	transformer1,...,transformer6	[T/32×output,..., T×output]	-

of three 1D conv layers were used. The kernel size of three conv layers were 1, 3 and 1, respectively. The expansion factor of the bottleneck block was 2. We added an extra strided conv layer with kernel size=1 and stride=2 to perform downsampling when necessary.

Training Details. For training, we considered both fixed length inputs (ActivityNet) and variable length inputs (THUMOS14, ActivityNet, and EPIC-Kitchens 100). For variable length inputs, we capped the input length to 2304 (around 5 minutes on THUMOS14 and around 20 minutes on EPIC-Kitchens 100), and randomly selected a subset of consecutive clips from an input video. Position embedding was disabled by default except for ActivityNet. Model EMA [8] and gradient clipping were also implemented to further stabilize the training. Hyperparameters were slightly different across datasets and discussed later in our experiment details.

Inference Details. For fixed length inputs (ActivityNet-1.3), we fed the full sequence into our model. For variable length inputs (THUMOS14 and EPIC-Kitchens 100), we sent the full sequence into the model. When using position embeddings in our ablation study, we adopted the technique from [5]. Specifically, for input sequences shorter than the training sequence length (2304), we fed the full sequence into our model and clipped the position embedding using the actual length of the video. For input sequences longer than the training sequence length, we again fed the full sequence into our model, yet used linear interpolation to upsample the position embeddings.

Score Fusion. For our experiments on THUMOS14 and ActivityNet-1.3, we sometimes consider score fusion using external classification scores. Specifically, given an input video, the top-2 video-level classes given by external classification scores were assigned to all detected action instances in this video, where the action scores from our model were multiplied with the external classification scores. Each detected action instance from our model thus creates two action instances. We refer the readers to [18] (Appendix E) for a more detailed description of the score fusion strategy.

Experiment Details. Our experiment details vary across datasets, as each dataset includes videos of different resolution and frame rate, and considers different types of features. We now describe our experiment details for THUMOS14, ActivityNet-1.3, and EPIC-Kitchens 100.

- **THUMOS14**: We used two-stream I3D [3] pretrained on Kinetics to extract the video features on THUMOS14, following [13,19]. We fed 16 consecutive frames as the input to I3D, used a sliding window with stride 4 and extracted 1024-D features before the last fully connected layer. The two-stream features were further concatenated (2048-D) as the input to our model. mAP@[0.3:0.1:0.7] was used to evaluate our model. Our model was trained for 50 epochs with a linear warmup of 5 epochs. The initial learning rate was 1e-4 and a cosine learning rate decay is used. The mini-batch size was 2, and a weight decay of 1e-4 was used.
- **ActivityNet-1.3**: We used two-stream I3D [3] for feature extraction, yet increased the stride of the sliding window to 16. Following [12,11,16], the extracted features were downsampled into a fixed length of 128 using linear interpolation. For evaluation, we used mAP@[0.5:0.05:0.95] and also reported the average mAP. Our model was trained for 15 epochs with a linear warmup of 5 epochs. The learning rate was 1e-3, the mini-batch size was 16, and the weight decay was 1e-4. For ActivityNet, we find it is helpful to train our model to generate proposals by considering all actions from a single category, and then use external classification scores for the recognition. This strategy was also used in previous single-stage TAL methods [10].
- **EPIC-Kitchens 100**: We used a SlowFast network [6] pre-trained on EPIC-Kitchens for feature extraction. This model is provided by [4]. We fed 32 frame window with a stride of 16 to extract 2304-D features. Our model was trained on the training set and evaluated on the validation set. A window size of 9 was used for local self-attention. For evaluation, we used mAP@[0.1:0.1:0.5] and report the average mAP following [4]. Our model was trained for 30 epochs with learning rate 1e-4, mini-batch size 2, and weight decay of 1e-4.

Reproducibility of Our Results. All results reported in the paper were obtained with the same random seed using PyTorch 1.10, CUDA 10.2 and CUDNN 7.6.5 on an NVIDIA Titan Xp GPU, using deterministic GPU computing routines. On the same machine, our code will always produce the same results when using the same random seed. Across machines/GPUs and computing environments, we have observed minor variation of average mAP scores (up to 0.5%

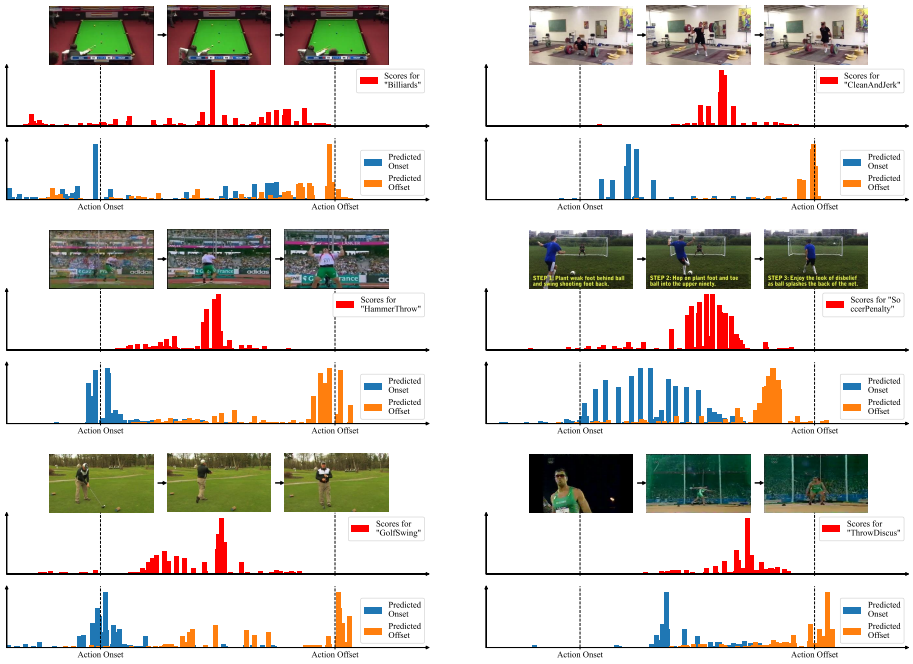

Fig. D. More visualization of our outputs. For each item from *top* to *bottom*: (1) the input video frames; (2) action scores at each time step; (3) histogram of action onsets and offsets computed by weighting the regression outputs using the action scores. *Left*: successful cases; *Right*: failure cases. This figure is best viewed in color and when zoomed in.

average mAP on THUMOS, less than 0.2% average mAP on ActivityNet, and under 0.8% average mAP on EPIC Kitchens), yet those minor variations do not erode the clear performance gains of our method.

Code. We have included our source code in the supplement. *The ECCV author guideline strongly discourages authors to include links to websites created for the purpose of their submission to ECCV'22 (e.g., to offer code or data, share demos or videos), even if steps were taken to anonymize them.* For this reason, we are unable to share all video features that are necessary to reproduce all our results. Nonetheless, we included scripts to download and process the publicly available TSP features [2] for ActivityNet 1.3 (with the best average mAP score on this dataset). The processed features can be used with our code to reproduce our best results on ActivityNet. Further, we also included a pre-trained model with its training logs in the supplement. We refer the readers to the instructions provided in our code to use our code and reproduce our results.

D Additional Visualizations

Further, we present more visualizations of our results in Fig. D, extending Fig. 3 of the main paper. Our model is able to detect the occurrence of actions and estimate their temporal boundaries for the most of the cases (see the first column of Fig. D). The major failure modes of our model, as demonstrated in the second column of Fig. D, include (1) incorrect classification of action centers, *i.e* background confusion (classification errors); (2) inaccurate regression of the action's onset and offset (localization errors). We plan to address these issues in our future work.

E Limitations

A main limitation of our method is the use of pre-extracted video features, also faced by many previous approaches. Another limitation is the need for many human labeled videos for training and the constraint of a pre-defined vocabulary of actions. Interesting future directions include pre-training for action localization [2,15], and learning from videos and text corpus [14,9] without human labels.

References

1. Alwassel, H., Caba Heilbron, F., Escorcia, V., Ghanem, B.: Diagnosing error in temporal action detectors. In: Eur. Conf. Comput. Vis. LNCS, vol. 11207, pp. 256–272 (2018)
2. Alwassel, H., Giancola, S., Ghanem, B.: TSP: Temporally-sensitive pretraining of video encoders for localization tasks. In: Int. Conf. Comput. Vis. Workshops. pp. 1–11 (2021)
3. Carreira, J., Zisserman, A.: Quo vadis, action recognition? a new model and the Kinetics dataset. In: IEEE Conf. Comput. Vis. Pattern Recog. pp. 4724–4733 (2017)
4. 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)
5. Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al.: An image is worth 16x16 words: Transformers for image recognition at scale. In: Int. Conf. Learn. Represent. (2021)
6. Feichtenhofer, C., Fan, H., Malik, J., He, K.: SlowFast networks for video recognition. In: Int. Conf. Comput. Vis. pp. 6202–6211 (2019)
7. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: IEEE Conf. Comput. Vis. Pattern Recog. pp. 770–778 (2016)
8. Huang, G., Li, Y., Pleiss, G., Liu, Z., Hopcroft, J.E., Weinberger, K.Q.: Snapshot ensembles: Train 1, get m for free. In: Int. Conf. Learn. Represent. (2017)
9. Jia, C., Yang, Y., Xia, Y., Chen, Y.T., Parekh, Z., Pham, H., Le, Q.V., Sung, Y., Li, Z., Duerig, T.: Scaling up visual and vision-language representation learning with noisy text supervision. In: Int. Conf. Mach. Learn. (2021)
10. Lin, C., Xu, C., Luo, D., Wang, Y., Tai, Y., Wang, C., Li, J., Huang, F., Fu, Y.: Learning salient boundary feature for anchor-free temporal action localization. In: IEEE Conf. Comput. Vis. Pattern Recog. pp. 3320–3329 (2021)

11. Lin, T., Liu, X., Li, X., Ding, E., Wen, S.: BMN: Boundary-matching network for temporal action proposal generation. In: Int. Conf. Comput. Vis. pp. 3889–3898 (2019)
12. Lin, T., Zhao, X., Su, H., Wang, C., Yang, M.: BSN: Boundary sensitive network for temporal action proposal generation. In: Eur. Conf. Comput. Vis. LNCS, vol. 11208, pp. 3–19 (2018)
13. Liu, D., Jiang, T., Wang, Y.: Completeness modeling and context separation for weakly supervised temporal action localization. In: IEEE Conf. Comput. Vis. Pattern Recog. pp. 1298–1307 (2019)
14. Radford, A., Kim, J.W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., et al.: Learning transferable visual models from natural language supervision. In: Int. Conf. Mach. Learn. (2021)
15. Xu, M., Pérez-Rúa, J.M., Escorcia, V., Martinez, B., Zhu, X., Zhang, L., Ghanem, B., Xiang, T.: Boundary-sensitive pre-training for temporal localization in videos. In: Int. Conf. Comput. Vis. pp. 7220–7230 (2021)
16. Xu, M., Zhao, C., Rojas, D.S., Thabet, A., Ghanem, B.: G-TAD: Sub-graph localization for temporal action detection. In: IEEE Conf. Comput. Vis. Pattern Recog. pp. 10156–10165 (2020)
17. Yang, L., Peng, H., Zhang, D., Fu, J., Han, J.: Revisiting anchor mechanisms for temporal action localization. IEEE Trans. Image Process. **29**, 8535–8548 (2020)
18. Zeng, R., Huang, W., Tan, M., Rong, Y., Zhao, P., Huang, J., Gan, C.: Graph convolutional networks for temporal action localization. In: Int. Conf. Comput. Vis. pp. 7094–7103 (2019)
19. Zhao, P., Xie, L., Ju, C., Zhang, Y., Wang, Y., Tian, Q.: Bottom-up temporal action localization with mutual regularization. In: Eur. Conf. Comput. Vis. LNCS, vol. 12353, pp. 539–555 (2020)
20. Zhu, Z., Tang, W., Wang, L., Zheng, N., Hua, G.: Enriching local and global contexts for temporal action localization. In: Int. Conf. Comput. Vis. pp. 13516–13525 (2021)