Supplementary: Semi-Supervised Temporal Action Detection with Proposal-Free Masking

Sauradip Nag[1,2], Xiatian Zhu[1,3], Yi-Zhe Song[1,2], and Tao Xiang[1,2]

[1] CVSSP, University of Surrey, UK
[2] iFlyTek-Surrey Joint Research Centre on Artificial Intelligence, UK
[3] Surrey Institute for People-Centred Artificial Intelligence, University of Surrey, UK
{s.nag,xiatian.zhu,y.song,t.xiang}@surrey.ac.uk

1 Appendix A : More Implementation Details

Branch Label Assignment To train our SPOT [8], the ground-truth needs to be arranged into a specific format. Concretely, given a training video with temporal intervals and action class labels (Fig. 1(a)), we assign all the snippets (*e.g.*, the orange and green squares of class branch in Fig. 1(a)) lying in each action interval as (positive) action snippets with the shared action class. All the snippets outside of any action interval are labeled as (negative) background samples. For an action snippet in mask branch M, its segmentation mask label is defined as the full binary mask of the associated action instance with the whole video length (the columns with a sequence of blue squares in Fig. 1(a)). When there are multiple action snippets involved in a specific action instance (*e.g.*, 5/5 snippets covered by the first/second action instance from class i/j in Fig. 1(a), each will be assigned with the global segmentation mask label corresponding to that instance.

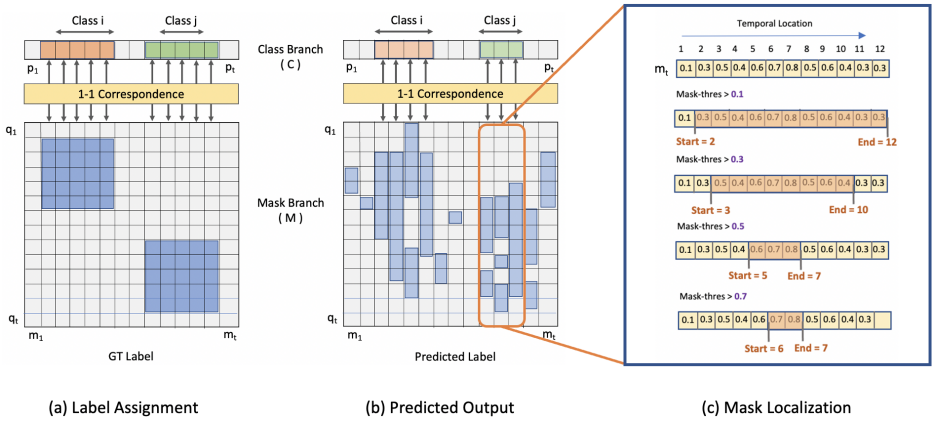

Fig. 1: Illustration of label assignment (see text for details).

Detailed Inference Process Our model inference is similar to existing TAD methods [5, 15, 6, 7, 9, 10]. Given a test video, the action instance predictions are first generated separately based on the classification $\boldsymbol{P}$ and mask $\boldsymbol{M}$ predictions and combined for the following post-processing. From P we first select the top-scoring snippets with class probability greater than a threshold θ_c (*e.g.*, the orange and green squares in Fig 1(b)), and obtain their corresponding segmentation mask predictions by thresholding the corresponding columns of $\boldsymbol{M}$ (Fig 1(c)). To generate sufficient candidates, we apply multiple thresholds $\Theta = \{\theta_i\}$ to yield action candidates with varying lengths and confidences. For each candidate, we compute its confidence score by multiplying the classification score (obtained from the corresponding top-scoring snippet in P) and the segmentation mask score (*i.e.*, the *mean* predicted foreground segment in M). Finally, we apply SoftNMS [1] on top scoring candidates to obtain the final predictions.
More Hyperparameters We first pre-train SPOT (except the classification stream) on the training set including the unlabeled samples for 12 epochs with a learning rate of $5 \times 10^{-3}/10^{-4}$ and weight decay of 4×10^{-3}. We then fine-tune SPOT for 15 epochs using Adam optimizer with a learning rate of $10^{-4}/10^{-5}$, a weight decay of $10^{-3}/10^{-5}$, and a batch size of 200/50.

2 Appendix B : More Results

Setting Apart from the proposed semi-supervised setting using 60% and 10% labeled data, we also experimented with a third setting with 50% labeled data in order to make a comparison with [11].
Competitors Similar to Table 1 in the main paper, we compare our approach with two semi-supervised TAD approaches (SSTAP [13] and SSP [4]) and the very recent SSAD [11]. SSTAP and SSP use UNet [12] for proposal classification, which is trained by full class labels. We use the TSN features as used in [16] for all the competitors for fair comparison.
Results The results are reported in Table 1. It can be observed that all existing methods are drastically outperformed by our SPOT model, due to suffering from localization error propagation. Besides, SSTAP [13] and SSP [4] outperforms SSAD [11]. Two possible reasons include the backbone of SSAD (SSN [16]) is weaker than that of SSTAP (BMN [5]), and the usage of 100% class labels used for training UNet [12].

3 Appendix C : More Analysis

Pre-text Task Design To validate our pre-text task for self-supervised pre-training, we experiment with 3 designs: (a) Without any augmentation, *i.e.*, randomly selecting multiple start and end points and keep the same raw features; (b) We randomly select multiple start and end points as the foreground, and mask out the features by zeroing the features corresponding to the snippets outside these selected foreground segments. This augmentation resembles the mask region modeling in [2] with a different purpose to select the start/end

Table 1: **SS-TAD results on test set of THUMOS14 under** 50% **label setting**. †: Using UntrimmedNet [12] trained with 100% labels for proposal classification.

Method	mAP				
	0.3	0.4	0.5	0.6	0.7
SSP† [4]	51.7	44.5	37.3	26.5	16.4
SSTAP† [13]	54.1	46.8	39.2	27.7	19.4
SSAD [11]	45.6	36.4	26.2	15.5	7.1
SPOT (Ours)	**54.8**	**47.9**	**39.3**	**30.6**	**22.8**

points of the masked segments; (c) We randomly select multiple start and end points, and instead of masking out the features, we add random noise to the features corresponding to the snippets outside these selected segments. This augmentation is similar to Random Erasing [17] but with a motivation of erasing pseudo background content alone, *i.e.*, erasing the features corresponding to the pseudo background snippets. From Fig. 2(a) we observe that the overall pre-training loss is more stable for random mask than other alternatives. Although both the augmented variants incur huge losses in the first few epochs than the no-augmentation variant, the random mask augmentation gradually improves the pre-training. For the variant without any augmentation, the feature reconstruction loss L_{rec} is useless, which is also indicated in the cosine-similarity of features in Fig 2(b). However, this might degrade the pre-training as seen in Table 3 (in main paper).

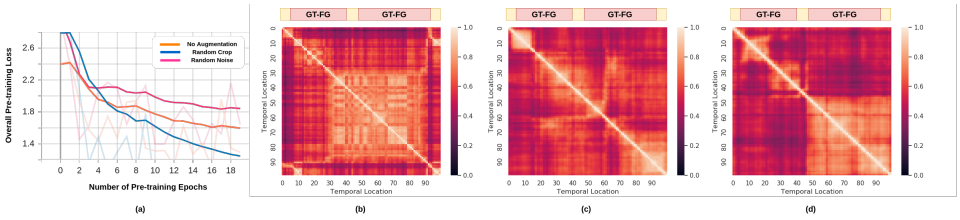

Fig. 2: Illustration of (a) the pre-training loss curve, and the cosine-similarity for (b) no-augmentation pre-training, (c) random noise augmentation, and (d) random mask augmentation.

Impact of Snippet Length We evaluate the impact of video snippet length on ActivityNet. As shown in Table 2, when the snippet length is small (*e.g.*, 50), we observe a performance drop of 2.7% in mAP@0.5. This may be due to that too small snippets are less capable to represent local motion patterns. We find that the length of 100 is the best, confirming the same findings as GTAD [15] and BMN [5].

Table 2: Impact of the snippet length of a video on ActivityNet w/ 10% labeled data.

Snippet Length	mAP	
	0.5	Avg
50	47.2	31.4
100	49.9	**32.1**
150	49.9	32.0
200	**50.0**	31.9

Number of Hyperparameters We compare the number of hyperparameters used for training the existing 2-stage (proposal generation then classification) SSTAP [13] and our single-stage SPOT. For clarity, we subdivide the total number of hyperparameters into 3 stages: (1) Hyperaprameters for semi-supervised learning (SSL); (2) Hyperparameters for Temporal Action Detection (TAD); (3) Hyperparameters for post-processing. From the results in Table 3, we observe that SSTAP has fewer hyperaprameters for semi-supervised learning, but twice the number of parameters in TAD along with more trainable parameters. Overall, our SPOT design cuts down significantly on the number of hyperparameters from 34 to 24, while being more light-weight.

Table 3: Number of trainable parameters and hyperparameters. SSL: Semi-supervised learning.

Method	No of Params (in M)	Number of Hyperparams			
		SSL	TAD	Post-Process	Total
SSTAP	53.2	**7**	12 (1st stage) + 11 (2nd stage)	4	34
SPOT	15.8	8	**12**	**4**	**24**

Snippet Embedding Design We compare the Transformer with CNN for snippet feature learning. To this end, we consider two CNN designs: (1) a 1D CNN with 3 dilation rates (1, 3, 5) each with 2 layers, and (2) a multi-scale Temporal Convolutional Network MS-TCN [3]. Each CNN design substitutes the Transformer respectively while remaining all the others. The results in Table 4 shows that the Transformer is clearly superior to both 1D-CNN and a relatively stronger MS-TCN.

Design Component Analysis Our SPOT primarily consists of a snippet embedding Transformer and 1-D Convolution heads for classification and localization streams. We ablate the number of 1-D CNN layers for both the branch heads in Table 5. As the results suggest, only 1 layer is enough for classification branch. A plausible reason for this is that for classification it needs global information and stacking multiple 1-D CNN may affect global information. For

Table 4: Transformer vs. CNN on ActivityNet under 10% label setting.

Network	**mAP**	
	0.5	**Avg**
1D CNN	40.7	24.5
MS-TCN	45.2	28.6
Transformer	**49.9**	**32.1**

localization branch, it is observed that 3 layers give best performance. This is probably because for predicting the masks the network needs to process local information captured by 1-D CNNs. Additionally, we also ablate the performance of transformer design in head size. Table 6 demonstrates that the performance of SPOT improves significantly with the increase of heads in the Transformer. However, excessive heads will lead to overfitting particularly when there are a small number of annotated samples. The performance peaks with four heads. We set the DropOut regularization to 0.3 for our snippet embedding Transformer.

Table 5: Effect of the number of 1-D CNN Layers for the classification and mask branches on ActivityNet under 10% label setting.

# Layers	Class. branch		Mask branch	
	0.5	Avg	0.5	Avg
1	**49.9**	**32.1**	46.2	31.0
2	49.2	32.0	48.6	31.9
3	48.9	31.9	**49.9**	**32.1**
4	48.0	31.8	49.8	32.1
5	47.7	31.4	49.5	32.0

Table 6: Impact of the head number in the Transformer on ActivityNet under 10% label setting.

Number of heads	mAP	
	0.5	Avg
1	42.5	29.9
2	44.1	30.5
3	47.5	31.1
4	**49.9**	**32.1**
5	48.0	31.8

Mask Design Recall that in Sec 3.2 we introduce a global mask $\in \mathbb{R}^{T\times T}$ for localizing action boundaries in a holistic manner, where T is the snippet length of a video. Alternatively, one can also learn a single 1-D actionness mask $\in \mathbb{R}^{T\times 1}$ per video [5, 14]. We integrate this actionness within our SPOT by reformulating the mask branch to output 1-D mask. From the results in Table 7, we observe a significant performance drop of 5.6%. This validates the efficacy of our mask design in terms of jointly learning multiple masks per snippet and focusing on a single action instance per mask.

Table 7: Effect of mask design on ActivityNet under 10% label setting.

Mask Design	mAP		Avg masks / video
	0.5	Avg	
Actionness	44.3	28.1	8
Our Global Mask	**49.9**	**32.1**	**250**

Kernel Size for Hard Snippet We use a differentiable erosion operation in Inter-Stream Interaction (*cf.* Sec. 3.2 in main paper). We use a kernel parameter e to get the mask boundaries from the localization stream. We ablate the value of e for the selection of mask boundary. From the results in Table 8, we fine the optimal e is 7.

Table 8: Impact of kernel size on ActivityNet in 10% label setting.

Kernel Size	mAP	
	0.5	Avg
3	46.6	30.8
5	47.8	31.2
7	**49.9**	**32.1**
9	49.2	32.0

Pretext Task: Single- vs. Multi-Scale Cropping Recall that we introduce a novel pretext task based on *Random Foreground* in Sec 3.3 of main paper. Given raw input features we create a masked feature by randomly masking out foreground features. In this experiment, we investigate which random cropping strategy (single vs. multi scales) is better. It is evident in Table 9 that pretraining with randomly cropping out segments at multiple different scales is superior by imposing more spatio-temoral dynamics and patterns.

Usefulness of boundary refinement In Sec. 3.2 we introduced a TAD refinement component based on inter-stream interaction with the aim to mitigate

Table 9: **Analysis of Cropping Strategy** during pre-training on ActivityNet under 60% label setting.

Cropping	mAP			
	0.5	0.75	0.95	Avg
Single-scale	51.1	32.2	8.8	32.7
Multi-scale	**52.8**	**35.0**	**8.1**	**35.2**

the boundary ambiguity problem. Here we examine its effectiveness in performance boost on ActivityNet with 10% labeled data. Besides the entire benefit, we also test the respective effects of the background and foreground terms, corresponding to the first and second loss terms of Eq. 3 (of main paper). Table 11 show that: **(1)** Our inter-stream interaction is effective for TAD refinement in semi-supervised learning with 1.5% gain in mAP. This verifies our consideration on the importance of boundary inference and our model design. **(2)** Both foreground and background terms are useful in isolation, and importantly present strong synergy as their combined effect (1.5% mAP gain) is much greater than the sum of their individual gains (0.7%=0.5%+0.2%). This is not surprising as the boundary reasoning requires to model both foreground and background well simultaneously.

Training and inference complexity We compare SPOT with a representative TAD method BMN [5] with our pre-training and a recent SS-TAD method SSTAP [15]. All the methods are tested under the same training setting and the same machine with one Nvidia 2080 Ti GPU. We measure the entire training time (including pre-training) and average inference time per video in testing. We use the two-stream video features used in [13] It can be seen in Table 10 that despite with pre-training and fine-tuning our SPOT is still drastically faster, *e.g.*, 31/35× for training and 2.3× for testing in comparison to SSTAP/BMN, respectively. This validates our motivation of designing a proposal-free SS-TAD model in terms of computational efficiency.

Table 10: Comparison of training and inference time on a single NVIDIA 2080 GTX.

Method	Training (in hrs)		Inference (in secs)
	Pre-train	Fine-tune	
BMN [5]	4.0	6.2	0.21
SSTAP [13]	-	9.4	0.21
SPOT	**0.10**	**0.21**	**0.09**

Table 11: Analysis of inter-stream interaction for TAD refinement on ActivityNet with 10% label supervision.

Refinement loss (Eq. 3)		mAP	
Foreground term	Background term	0.5	Avg
✗	✗	46.8	30.7
✓	✓	**49.9**	**32.1**
✗	✓	47.2	31.1
✓	✗	46.9	30.8

Table 12: Positional Encoding on ActivityNet W/ 10% labels.

# Position Encoding	mAP	
	0.5	Avg
No Encoding	**49.9**	**32.1**
Learnable	46.7	29.4
Non-Learnable	39.8	24.2

Table 13: Ablation of pretraining loss during finetuning W/ 10% labels on ActivityNet.

Loss	Avg
W/O L_{tp}	**32.1**
W/ L_{tp}	31.9

Role of Positional Encoding We evaluate the effect of position encoding on ActivityNet. As shown in Table 12, it is interesting to see that position encoding is not necessary and even harmful to the performance. This indicates that with our current formulation, the snippet level temporal information does not bring extra useful information.

Effect of Pre-training Loss in Finetuning Recall that we do not use the pre-training loss L_{tp} (for temporal ordering pretext task) during the finetuning stage as shown in Sec 3.3. Table 13 shows a slight drop of 0.2% in avg mAP from this pre-text loss which may be due to the incompatibility with the classification loss L_c of TAD. This is not rare in pretraining-finetuning pipeline with the pretext loss dropped during finetuning.

Handling Class-Imbalance Challenge In section 3.3 of main paper, we introduce a new class-balanced loss to handle the class-imbalance problem in SS-TAD. We evaluate on the videos corresponding to top-10 tail classes on ActivityNet in terms of error rate. The imbalance problem arises mainly due to the snippet coverage of a particular class (Fig. 3(b)), and we see a high correlation with the error rate in Fig. 3(a). Importantly, using our class-balanced loss, the error rate of the heavily imbalanced *"Drinking Coffee"* class can be reduced by $\sim 20\%$.

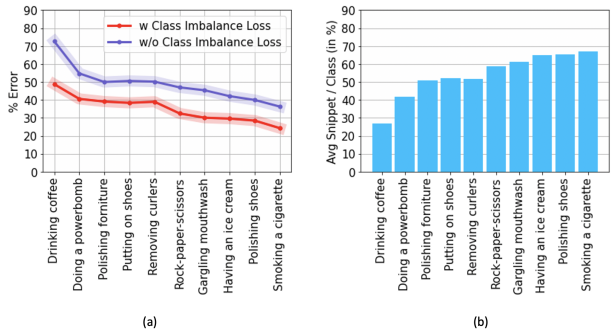

Fig. 3: **Effect of tackling class imbalance** on top-10 tail classes from ActivityNet. (a) shows the effect of dealing with the class imbalance. (b) shows the foreground coverage per class.

References

1. Bodla, N., Singh, B., Chellappa, R., Davis, L.S.: Soft-nms–improving object detection with one line of code. In: Proceedings of the IEEE international conference on computer vision. pp. 5561–5569 (2017)
2. Chen, Y.C., Li, L., Yu, L., El Kholy, A., Ahmed, F., Gan, Z., Cheng, Y., Liu, J.: Uniter: Universal image-text representation learning. In: European conference on computer vision. pp. 104–120. Springer (2020)
3. Farha, Y.A., Gall, J.: Ms-tcn: Multi-stage temporal convolutional network for action segmentation. In: CVPR. pp. 3575–3584 (2019)
4. Ji, J., Cao, K., Niebles, J.C.: Learning temporal action proposals with fewer labels. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 7073–7082 (2019)
5. Lin, T., Liu, X., Li, X., Ding, E., Wen, S.: Bmn: Boundary-matching network for temporal action proposal generation. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 3889–3898 (2019)
6. Nag, S., Zhu, X., Song, Y.Z., Xiang, T.: Temporal action localization with global segmentation mask transformers (2021)
7. Nag, S., Zhu, X., Song, Y.z., Xiang, T.: Proposal-free temporal action detection via global segmentation mask learning. In: ECCV (2022)
8. Nag, S., Zhu, X., Song, Y.z., Xiang, T.: Semi-supervised temporal action detection with proposal-free masking. In: ECCV (2022)
9. Nag, S., Zhu, X., Song, Y.z., Xiang, T.: Zero-shot temporal action detection via vision-language prompting. In: ECCV (2022)
10. Nag, S., Zhu, X., Xiang, T.: Few-shot temporal action localization with query adaptive transformer. arXiv preprint arXiv:2110.10552 (2021)
11. Shi, B., Dai, Q., Hoffman, J., Saenko, K., Darrell, T., Xu, H.: Temporal action detection with multi-level supervision. In: CVPR. pp. 8022–8032 (2021)
12. Wang, L., Xiong, Y., Lin, D., Van Gool, L.: Untrimmednets for weakly supervised action recognition and detection. In: CVPR. pp. 4325–4334 (2017)
13. Wang, X., Zhang, S., Qing, Z., Shao, Y., Gao, C., Sang, N.: Self-supervised learning for semi-supervised temporal action proposal. In: CVPR. pp. 1905–1914 (2021)
14. 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 (2020)
15. Xu, M., Zhao, C., Rojas, D.S., Thabet, A., Ghanem, B.: G-tad: Sub-graph localization for temporal action detection. In: CVPR (2020)
16. Zhao, Y., Xiong, Y., Wang, L., Wu, Z., Tang, X., Lin, D.: Temporal action detection with structured segment networks. In: ICCV (2017)
17. Zhong, Z., Zheng, L., Kang, G., Li, S., Yang, Y.: Random erasing data augmentation. In: Proceedings of the AAAI conference on artificial intelligence. vol. 34, pp. 13001–13008 (2020)