End-to-End Spatio-Temporal Action Localisation with Video Transformers

Alexey A. Gritsenko Xuehan Xiong Josip Djolonga Mostafa Dehghani
Chen Sun Mario Lučić Cordelia Schmid Anurag Arnab
Google
{agritsenko, aarnab}@google.com

In this supplementary material, we include further details about the discrepancies in the TubeR [22] authors' public code release (Sec. A) and additional experiments and details (Sec. B).

A. Discrepancies in official TubeR code

A.1. Attempt to reproduce TubeR training

In order to make fair comparisons to TubeR, we attempted to train a model following the instructions in the authors' public code release.

Firstly, in order to be able to run the code, we needed to fix some programming bugs as pointed out in a Github issue.

Once these fixes were applied, we were able to run the code. However, the performance on AVA after just a few epochs was poor, as shown below in Tab. A1.

Table A1. Initial attempt at running public TubeR training code on AVA. The results are poor as the public code does not load pretrained weights into the backbone by default.

Epochs	1	3
mAP	1.43	1.98

Looking further into the configuration files provided by the authors, we noticed that, MODEL.PRETRAINED = False, meaning that weights from the backbone were not being loaded, as shown here.

By modifying the configuration file to load the pretrained weights of the backbone, the performance did improve. But after training completed, the results were far from what was expected, as shown in Tab. A2. Concretely, we obtained a final mAP of 19.9, when we expected to achieve 31.1.

We have contacted the authors regarding this issue, and have not received any response. As a result, we were not able to use the public TubeR training code for performing fair comparisons to it, and had to use our reimplementation of it in our training framework.

Finally, note that TubeR achieve their best results with a "Long-Term Context" module adapted from Wu et al. [19]. However, this part is not included in their public code release at all.

We are committed to releasing full training code on acceptance of our paper.

A.2. DETR pretraining

The public release of TubeR initialises the decoder part of the network using DETR pretrained weights, as shown here. However, this fact was not mentioned in the paper at all.

We have contacted the authors to ask what pretraining data was used here, but they have not responded. However, based on Github issues [1] and [2], we have inferred that the DETR was in fact pretrained on COCO.

We are not aware of any other work on AVA that has used object detection pretraining, and this initialisation therefore makes TubeR not fairly comparable to other works in the literature.

Our Tubelet decoder (Section 3.2) is randomly initialised, and does not make use of additional data unlike TubeR.

A.3. Unclear if TubeR actually predicts tubelets on AVA

Reading through the public TubeR code, we noticed that the model is specialised according to the dataset (for example [1, 2, 3]). In particular, when training on AVA, the dimensionality of the queries, $\mathbf{q} \in \mathbb{R}^{S \times d}$ where S is the number of spatial queries, as shown here. On other datasets like UCF101, $\mathbf{q} \in \mathbb{R}^{T \times S \times d}$ where T is the temporal dimension, as shown here. This suggests that TubeR does not predict tubelets on the AVA dataset, but rather just bounding boxes at the centre keyframe. However, Figure 5 of the TubeR paper [22] implies that tubelets are predicted.

We have contacted the authors to clarify the aforementioned details, but they have not responded.

In contrast, we do not instantiate our model differently for each dataset. And our model predicts tubelets even if we have only sparse keyframe supervision. This was demonstrated in Table 4 of the main paper (where we performed

Table A2. Results of running the TubeR authors' public training code on AVA, with the necessary bug-fixes and corrections as described in the text. The final accuracy is still, however, much lower than expected (mAP of 19.9, compared to the expected 31.1). As a result, we could not build upon the public TubeR code-base, and use it to make fair comparisons to TubeR.

Epoch	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
mAP	11.28	15.08	17.77	18.50	18.81	19.59	20.34	20.98	21.00	20.43	20.94	21.77	20.76	20.03	20.35	20.42	21.14	20.79	20.39	19.91
Expected mAP																				31.10

Table A3. Comparison of computational cost to representative methods on the AVA dataset. TubeR [22] and Co-finetuning [3], which use external LTC and person detector modules, do not report FLOPs and parameters of these extra modules, like other two-stage approaches. As shown in the first row, the cost of the person detector (Faster-RCNN with ResNeXt-101 FPN) used in [2, 3, 7, 8, 17–20] is significant, as it can be more than the action localisation model itself.

Model	AVA	Total GFLOPs	Params (10^6)
Person detector	–	756.1	122.2
TubeR [22]	31.1	240	90.1
TubeR with LTC [22]	33.6	240 + LTC	90.1 + LTC
Co-finetuning [3]	36.1	4738 + person detector	431 + detector
InternVideo [18]	41.0	–	–
STAR/B 4 frames	32.3	**168**	126.8
STAR/B 8 frames	33.1	336	126.8
STAR/B 16 frames	34.3	672	126.8
STAR/B 32 frames	36.3	1345	126.8
STAR/L 32 frames	**41.7**	5669	417.1

weakly-supervised tubelet detection), and by visualisations of our results on AVA in the attached supplementary video.

B. Additional experiments

B.1. Computational Cost

Fairly comparing the computational cost of state-of-the-art models is difficult:

TubeR [22] achieve their best results using a "Long-term context" (LTC) module from Wu et al. [19], which first precomputes a "Long-term feature bank" [19] from the entire video clip. The TubeR paper [22] provides no details about how the "Long-term feature bank" is computed (and their public code does not include this module either). However, the original paper [19] on the AVA dataset, precomputed features over the entire 15 minute video, to supplement the 2.5 second clip that was processed by the model. Therefore, one can assume that TubeR's LTC module adds orders of magnitude additional computation.

Moreover, all of the two-stage methods in Table 9 use an additional person detector for proposals. However, they do not report the computation cost of it at all. We analysed the cost of the Faster R-CNN with ResNeXt-101-FPN [12, 15] person-detector used in [2, 3, 7, 8, 17–20], and observed that it is actually more costly than many action detection models itself, as shown in Tab. A3.

In addition, some papers, such as InternVideo [18], do not report compute metrics either.

Table A3 shows that we can trade-off speed and accuracy by varying the number of input frames. Notably, with 4 frames at 320p resolution, STAR with a ViViT-Base backbone uses the least GFLOPs and outperforms TubeR without LTC (the only variant of TubeR for which we know the total GFLOPs). Moreover, STAR/B with 16 frames uses less GFLOPs and parameters than the person detector used by existing two-stage action localisation algorithms. STAR/B with 32 frames also outperforms Arnab et al. [3] with less than 25% of the total GFLOPs.

B.2. Implementation details

We exhaustively list hyperparameter choices for the models used in our state-of-the-art comparisons in Tab. A4 and A5.

Note that our model hyperparameters in Tab. A4 follow the same nomenclature from ViT [6] and ViViT [1] for defining "Base" and "Large" variants.

Our experiments use similar data pre-processing and augmentations as prior work [8, 19, 20], such as horizontal flipping, colour jittering (consistently across all frames of the video) and box jittering. In addition, we used a novel keyframe "decentering" augmentation (Sec. B.5) as our model predicts tubelets, and more aggressive scale augmentation (Sec. B.6).

We train with synchronous SGD and a cosine learning rate decay schedule. As shown in Tab. A5, we typically use the same training hyperparameters across experiments. Note that for the JHMDB dataset, we use $T = 40$ frames as input to our model, as this is sufficient to cover the longest video clips in this dataset. We also do not need to perform "decentering" (Sec. B.5) for datasets with full tube annotations (UCF101-24 and JHMDB51-21). As shown in Tab. A4, we found it beneficial to use a lower learning rate for the vision encoder of our model, as it was already pretrained, in contrast to the decoder which was learned from scratch.

B.3. Further comparison of query parameterisation to TubeR

We extend our experiments from Tab. 1 to UCF and JHMDB in Tab. A6. We observe that our method of binding queries to people, instead of actions (as done in TubeR [22]), still improves here, albeit by a smaller margin.

Table A4. Model architecture hyperparameters. We used the same decoder even when scaling up the vision encoder.

Hyperparameter	Model size	
	Base	Large
Decoder		
Number of layers	6	
Learning rate	10^{-4}	
Hidden size	256	
MLP dimension	2048	
Dropout rate	0.1	
Box head num. layers	3	
Encoder		
Learning rate	5×10^{-6}	2.5×10^{-6}
Learning rate (CLIP init.)	1.25×10^{-6}	
Patch size	$16 \times 16 \times 2$	
Spatial num. layers	12	24
Temporal num. layers	4	8
Attention heads	12	16
Hidden size	768	1024
MLP dimension	3072	4096

Although these datasets have one action per tube, we also need to predict when an action starts and ends within a tubelet. TubeR's approach requires an additional "action switch" [22], which we do not (Sec. 3.4), and so our design may aid model training.

Note that experiments were performed using the same settings as Tab. 1, namely using a ViViT-Base backbone and a frame resolution of 160p.

B.4. Additional analysis of query parameterisation and matching

Tables 2 and 3 in the paper analysed the effect of our query parameterisation, and matching in the loss calculation, on the AVA and UCF101-24 datasets. In Tab. A7 and A8, we perform these experiments on JHMDB too, and find that all our findings are consistent here as well.

B.5. Decentering

The majority of prior work on keyframe-based action localisation datasets (*e.g.* AVA and AVA-Kinetics) predict only at the centre frame of the video clip, as only sparse supervision at this central keyframe is available. As our model predicts *tubelets*, we intuitively would like to supervise it for other frames in the input clip as well.

To this end, we introduce another data augmentation strategy, named "decentering", where we sample video clips during training such that the keyframe with supervision is no longer at the central frame, but may deviate randomly from the central position. We parameterise this by an integer, ρ, which defines the maximum possible deviation, and randomly sample a displacement $\in [-\rho, \rho]$ during training.

We found that this data augmentation strategy results in qualitative improvements in the predicted tubelets (as shown in the supplementary video). However, as shown in Tab. A9, it has minimal effect on the Frame AP which only measures performance on the annotated, central frame of AVA video clips.

Note that for datasets with full tube annotations, *i.e.* UCF101-24 and JHMDB51-21, there is no need to apply decentering, as each frame of the video clip is already annotated. We do, however, use decentering with the $\rho = 8$, when training with weak supervision on UCF101-24 (Tab. 4 of the main paper).

B.6. Scale augmentation

Consistent with object detection in images [5, 9, 13, 16], we found it necessary to perform spatial scale augmentation during training to achieve competitive action localisation performance. As shown in Tab. A10, we found that performing "zoom out" as well as "zoom in" scale augmentation during training significantly boosts action localisation performance. This departs from the choice of performing "zoom in" only scale augmentation in previous work [8, 19, 20].

B.7. Focal and auxiliary loss

Following [14, 21, 23] we use sigmoid focal cross-entropy loss [13] as our classification loss,

$$\begin{aligned}\mathcal{L}_{\text{class}}(a, \hat{a}) = &- \alpha \cdot a \cdot \hat{a}^{\gamma} \log(\hat{a}) \\ &- (1-\alpha)(1-a)(1-\hat{a})^{\gamma} \log(1-\hat{a}),\end{aligned} \tag{A1}$$

where a and $\hat{a}$ are the ground truth and predicted action class probabilities respectively. α and γ are hyperparameters of the focal loss [13]. Furthermore, following Minderer et al. [14] we do not use auxiliary losses [4] (*i.e.* attaching output heads after each decoder layer and summing up the losses from each layer) previously found to be beneficial for matching-based detection models. Both of these choices are motivated by our ablations in Tab. A11: We observe that the focal loss consistently improves performance, and that auxilliary losses are only beneficial when the focal loss is not used.

Table A5. Model training hyperparameters for the four datasets considered in our paper. We train with synchronous SGD and a cosine learning rate decay schedule.

Hyperparameter	Dataset			
	AVA	AVA-K	UCF101-24	JHMDB51-21
Epochs (training steps)	30 (148 050)	30 (246 690)	30 (88 230)	40 (6 800)
Batch size	128			
Optimiser	Adam [11]			
Adam β_1	0.9			
Adam β_2	0.999			
Gradient clipping ℓ_2 norm	1.0			
Focal loss α	0.3			
Focal loss γ	2.0			
Number of spatial queries (S)	64			
Number of frames (T)	32	32	32	40
Center deviation, ρ: per-frame matching	4	4	0	0
Center deviation, ρ: tubelet matching	16	16	0	0
Stochastic depth [10]	0.2	0.2	0.5	0.5

Table A6. Further comparison of query parameterisation, using the ViViT-Base backbone at 160p resolution. Binding queries to actions is done in TubeR [22]. We report the Frame AP in all cases.

	AVA	UCF101-24	JHMDB51-21
Query binds to action	23.6	87.4	84.1
Query binds to person	**26.7**	**88.3**	**84.7**

Table A7. Comparison of independent and factorised queries the JHMDB51-21 dataset. Factorised queries are particularly beneficial for predicting tubelets, as shown by the largest improvement in the Video AP50. Both models use tubelet matching in the loss.

Query	fAP	vAP20	vAP50
Independent	85.0	88.5	85.2
Factorised	**86.9**	**89.5**	**88.2**

Table A8. Comparison of independent and tubelet matching for computing the loss on JHMD51-21. Tubelet matching is particularly beneficial for the Video AP50, showing that it helps to predict more temporally coherent tubelets, as is the case on UCF101-24 as well (in the main paper).

Query	fAP	vAP20	vAP50
Per-frame matching	86.0	88.3	86.0
Tubelet matching	**86.9**	**89.5**	**88.2**

Table A9. Effect of keyframe decentering studied on the AVA dataset (resolution 160p) for a model with IN21K→K400 initialisation, and factorised queries. Mild amounts of keyframe decentering do not hurt performance measured on the center frame while explicitly supervising the models ability to localise and predict actions on other frames. In fact, models trained with small amounts of decentering tend to perform better than models trained without any decentering.

Center deviation, ρ	mAP
0	26.5
1	26.8
2	26.5
4	26.7
8	26.4
16	26.6

Table A10. Comparison of spatial scale augmentation for our models with a ViViT/B backbone on the AVA dataset (resolution of 140p on the shorter side). We find that large range in scale-jittering, in the range of $(0.5, 2.0)$ of the original input frame, as used in [9] works the best. Notably, doing scale augmentation in range of $(\frac{8}{7}, \frac{10}{7})$ as in the open-sourced SlowFast [8] performs significantly worse. Performing no scale augmentation (first row) performs the worst as expected.

Scale (min, max)	mAP
$(1, 1)$ (none)	22.5
$(1.14, 1.43)$ [8]	23.9
$(0.25, 1.0)$	22.7
$(0.5, 1.0)$	23.4
$(0.25, 4.0)$	25.1
$(0.5, 2.0)$	**25.6**

Table A11. Effect of using sigmoid loss and auxiliary losses studied on the AVA dataset (resolution 160p) for a model with IN21K→K400 initialisation. Focal loss ($\alpha = 0.3$ and $\gamma = 2$) clearly performs better than the alternatives. Moreover, the use of auxiliary losses leads to a mild degradation in performance when combined with focal loss, but improves results when the focal loss is not used.

Focal loss	Auxiliary losses	mAP
✗	✗	20.8
✗	✓	21.8
✓	✓	26.4
✓	✗	**26.8**

References

[1] Anurag Arnab, Mostafa Dehghani, Georg Heigold, Chen Sun, Mario Lučić, and Cordelia Schmid. ViViT: A Video Vision Transformer. In *ICCV*, 2021. 2

[2] Anurag Arnab, Chen Sun, and Cordelia Schmid. Unified graph structured models for video understanding. In *ICCV*, 2021. 2

[3] Anurag Arnab, Xuehan Xiong, Alexey Gritsenko, Rob Romijnders, Josip Djolonga, Mostafa Dehghani, Chen Sun, Mario Lučić, and Cordelia Schmid. Beyond transfer learning: Co-finetuning for action localisation. In *arXiv preprint arXiv:2207.03807*, 2022. 2

[4] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *ECCV*, 2020. 3

[5] Ekin D. Cubuk, Barret Zoph, Jonathon Shlens, and Quoc V. Le. Randaugment: Practical automated data augmentation with a reduced search space. In *arXiv preprint arXiv:1909.13719*, 2019. 3

[6] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *ICLR*, 2021. 2

[7] Haoqi Fan, Bo Xiong, Karttikeya Mangalam, Yanghao Li, Zhicheng Yan, Jitendra Malik, and Christoph Feichtenhofer. Multiscale vision transformers. In *ICCV*, 2021. 2

[8] Christoph Feichtenhofer, Haoqi Fan, Jitendra Malik, and Kaiming He. Slowfast networks for video recognition. In *ICCV*, 2019. 2, 3, 4

[9] Golnaz Ghiasi, Yin Cui, Aravind Srinivas, Rui Qian, Tsung-Yi Lin, Ekin D Cubuk, Quoc V Le, and Barret Zoph. Simple copy-paste is a strong data augmentation method for instance segmentation. In *CVPR*, 2021. 3, 4

[10] Gao Huang, Yu Sun, Zhuang Liu, Daniel Sedra, and Kilian Weinberger. Deep networks with stochastic depth. In *ECCV*, 2016. 4

[11] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015. 4

[12] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *CVPR*, 2017. 2

[13] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *ICCV*, 2017. 3

[14] Matthias Minderer, Alexey Gritsenko, Austin Stone, Maxim Neumann, Dirk Weissenborn, Alexey Dosovitskiy, Aravindh Mahendran, Anurag Arnab, Mostafa Dehghani, Zhuoran Shen, et al. Simple Open-Vocabulary Object Detection with Vision Transformers. In *ECCV*, 2022. 3

[15] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *NeurIPS*, 2015. 2

[16] Bharat Singh, Mahyar Najibi, and Larry S Davis. Sniper: Efficient multi-scale training. In *NeurIPS*, 2018. 3

[17] Jiajun Tang, Jin Xia, Xinzhi Mu, Bo Pang, and Cewu Lu. Asynchronous interaction aggregation for action detection. In *ECCV*, 2020. 2

[18] Yi Wang, Kunchang Li, Yizhuo Li, Yinan He, Bingkun Huang, Zhiyu Zhao, Hongjie Zhang, Jilan Xu, Yi Liu, Zun Wang, et al. Internvideo: General video foundation models via generative and discriminative learning. In *arXiv preprint arXiv:2212.03191*, 2022. 2

[19] Chao-Yuan Wu, Christoph Feichtenhofer, Haoqi Fan, Kaiming He, Philipp Krahenbuhl, and Ross Girshick. Long-term feature banks for detailed video understanding. In *CVPR*, 2019. 1, 2, 3

[20] Chao-Yuan Wu, Yanghao Li, Karttikeya Mangalam, Haoqi Fan, Bo Xiong, Jitendra Malik, and Christoph Feichtenhofer. MeMVit: Memory-augmented multiscale vision transformer for efficient long-term video recognition. In *CVPR*, 2022. 2, 3

[21] Hao Zhang, Feng Li, Shilong Liu, Lei Zhang, Hang Su, Jun Zhu, Lionel Ni, and Heung-Yeung Shum. Dino: Detr with improved denoising anchor boxes for end-to-end object detection. In *ICLR*, 2023. 3

[22] Jiaojiao Zhao, Yanyi Zhang, Xinyu Li, Hao Chen, Bing Shuai, Mingze Xu, Chunhui Liu, Kaustav Kundu, Yuanjun Xiong, Davide Modolo, et al. TubeR: Tubelet Transformer for Video Action Detection. In *CVPR*, 2022. 1, 2, 3, 4

[23] Xizhou Zhu, Weijie Su, Lewei Lu, Bin Li, Xiaogang Wang, and Jifeng Dai. Deformable DETR: Deformable transformers for end-to-end object detection. In *ICLR*, 2021. 3