Supplementary material
COMEDIAN: Self-Supervised Learning and Knowledge Distillation for Action Spotting using Transformers

Julien Denize [*†] Mykola Liashuha [*] Jaonary Rabarisoa [*] Astrid Orcesi [*]
Romain Hérault [‡]
[*] Université Paris-Saclay, CEA, List, F-91120, Palaiseau, France
firstname.lastname@cea.fr
[†] Normandie Univ., INSA Rouen, LITIS, 76801, Saint Etienne du Rouvray, France
firstname.lastname@insa-rouen.fr
[‡] Normandie Univ., UNICAEN, ENSICAEN, CNRS, GREYC, 14000 Caen, France
firstname.lastname@unicaen.fr

A. Implementation details

A.1. Architectures

Encoders. For the spatial encoder, two different transformer architectures are used: ViT [3] and Swin [5]. By default, the temporal encoder is a stack of 4 attention layers as in ViT architecture. For ViT, the global architecture corresponds to the ViViT model 2 [1]. We keep this name and refer to the Swin based-architecture as ViSwin. In Tab. 1, we provide the input dimension of tokens for the spatial and temporal encoders, and their number of parameters and GFLOPs for ViViT Tiny, ViViT Small, and ViSwin Tiny.

For all models, the majority of the computations are performed in the spatial encoder which sees a lot of tokens, and the temporal encoder computational cost is negligible. However, the number of parameters does not scale well with the output dimension of the spatial encoder, due to the self-attention mechanism, which is reflected in ViSwin Tiny. It has 4 times more temporal parameters than ViViT small but only 1.25 times more spatial parameters. However, as Swin has less reduced computational usage in comparison with ViT by design [5] it scales better to deeper spatial architectures.

A.2. Optimizers

We use the optimizer ADAMW for pretraining and finetuning with a weight decay of 0.05. The initial learning rate depends on the training step as well as the backbone as detailed below. However, the steps follow different linear scaling rules for an initial learning rate η:

- Step 1: $\eta_{scaled} = \eta \times \frac{batch\_size}{256}$
- Step 2 and 3: $\eta_{scaled} = \eta \times \frac{batch\_size}{256} \times \frac{T_g}{64}$ with T_g the number of global frames per video.

Step 1. The initial learning rate is $\eta = 5 \times 10^{-4}$ with 10 epochs of warmup and a cosine annealing scheduler is applied throughout training.

Step 2. The initial learning rate is $\eta = 0.002$ with 10 epochs warmup and cosine annealing scheduler that ends at $0.01 \times \eta$.

Step 3. The initial learning rate is $\eta = 5 \times 10^{-4}$ for ViViT and $\eta = 3 \times 10^{-4}$ for ViSwin that ends at $0.01 \times \eta$.

A.3. Spatial Pretraining.

To perform the pretraining of the spatial encoder, we follow practices introduced by ρMoCo [4] and SCE [2]. More specifically we use a 3 layers Multi-Layer Perceptron (MLP) on top of the online and target encoders of hidden size 1024 and output size 256 that is discarded after this step. The online predictor is a 2 layers MLP with the same hidden and output size as the projectors. The data augmentation distributions are the standard contrastive ones used on images [2] and the temperature applied is $\tau = 0.1$. The momentum buffer size is $65,536$. For data sampling, all sub-videos of 1 second, or 2 frames, are used. The model is trained for 100 epochs with a batch size of 1024.

A.4. Spatio-temporal pretraining.

To perform the pretraining of the spatio-temporal encoder, we follow practices introduced by SCE [2]. More specifically we use a 3-layer MLP on top of the temporal encoder of hidden size 1024 and output size 512 to match the dimension of the Baidu [8] features. The projector is later discarded. For the SCE loss parameters we use $\tau = 0.1$,

	ViViT Tiny	ViViT Small	ViSwin Tiny
Input dim	$C \times T_g \times H \times W$	$C \times T_g \times H \times W$	$C \times T_g \times H \times W$
Spatial encoder			
Input tokens dim	$\left(\frac{H}{16} \times \frac{W}{16} + 1\right) \times \frac{T_g}{2} \times 192$	$\left(\frac{H}{16} \times \frac{W}{16} + 1\right) \times \frac{T_g}{2} \times 384$	$\frac{H}{4} \times \frac{W}{4} \times \frac{T_g}{2} \times 96$
Num parameters	5.7M	22.0M	27.5M
GFLOPs	41.19	149.30	144.68
Temporal encoder			
Input tokens dim	$\frac{T_g}{2} \times 192$	$\frac{T_g}{2} \times 384$	$\frac{T_g}{2} \times 768$
Num parameters	1.8M	7.1M	28.4M
GFLOPs	0.06	0.24	0.96
Global model			
Num parameters	7.5M	29.1M	55.9M
GFLOPs	41.25	149.54	145.64

Table 1. Comparison of the ViViT Tiny, ViViT small, and ViSwin Tiny spatial and temporal encoders and global model in terms of computational usage.

$\tau_m = 0.07$, $\lambda = 0.5$. The data augmentation used is the $strong_\gamma$ without cropping reported in the SCE paper. For data sampling, we randomly extract 150 videos of 32 seconds or 64 frames per game at each epoch. The batch size for 32-second videos at 2 FPS is 64. For longer clips, the batch size is inversely proportional to the length and number of windows. For example, for 64 seconds, the batch size is 32 and the number of windows sampled per match is 75.

A.5. Finetuning.

A linear classifier is applied to each output temporal token to perform fine-tuning. Each video sampled is augmented by using color jittering with probability 0.8 and of strength ± 0.4 on brightness, contrast, and saturation and 0 for hue to avoid changing the color of cards. Random Gaussian blur is also applied with probability 0.5 and a kernel size of 23 with $\sigma \in [0.1, 2.]$. A horizontal flip of probability 0.5 is also applied followed by a mixup [7] whose mixing coefficient is sampled by a Beta law $\mathcal{B}(0.1, 0.1)$.

The classifier is first trained during 30 epochs for its initialization and then the whole architecture is fine-tuned for 20 epochs for ViViT and 10 for ViSwin. The learning rate is reset for the second part.

For data sampling, 100 videos per match are uniformly sampled whilst enforcing that the beginning and end of each half are selected to avoid missing kickoffs and last-second actions. The batch size for 32-second videos at 2 FPS is 128. For longer clips, the batch size is inversely proportional to the length and number of windows. For example, for 64 seconds, the batch size is 64 and the number of windows sampled is 50 per match.

Seconds	t-AmAP (%)
0	66.4
2	66.6
4	66.7
6	66.6
8	66.6
10	**66.8**
12	**66.8**
14	**66.8**
16	66.7

Table 2. Influence of the number of seconds ignored at the start and end of each window prediction on the t-AmAP.

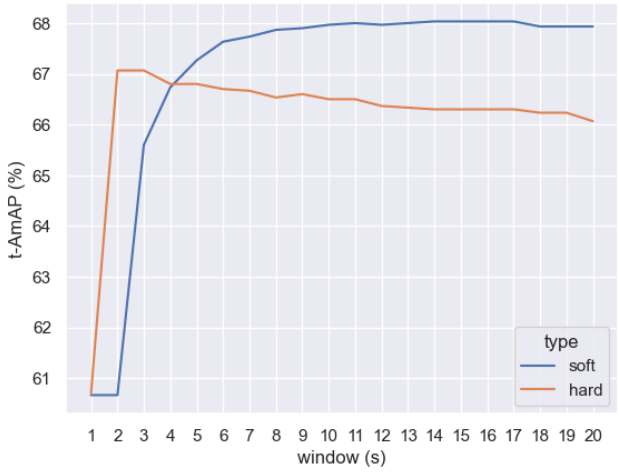

Figure 1. Influence of the soft and hard NMS and its window size in second on the t-AmAP.

B. Inference hyper-parameters search

During inference, a sliding window with half overlap is applied on all videos. For multiple timestamp classifications, the maximum of predictions per action is kept. No data augmentation is applied. By default, a hard Non-Maximum Suppression (NMS) of a 5-second window is applied. The 6 first and last seconds of each window prediction are ignored to keep predictions that have past and future context.

In Tab. 2, we study the effect of varying the number of seconds to ignore. Taking all predictions has the worst result of 66.4% t-AmAP showing that it is interesting to remove predictions on edge that do not have access to the context from the past or the future. The results increase up to 66.8% at 10 seconds and are stable for further seconds ignored. The increase in performance is relatively low and can be explained by the fact that the inference sliding window allows for some undetected predictions on edges to be retrieved by past or future windows.

In Fig. 1, we study the effect of using Hard or Soft NMS. As for [6], we see an increase in using soft NMS over hard NMS. Depending on the NMS type the optimal temporal window size for NMS is not the same. The best results are achieved for a hard NMS with a 4-5 seconds window at 66.8% t-AmAP and 68.0% for a soft NMS with an 11-17 seconds window. The results show that not only does soft NMS perform better than hard NMS but is also more stable.

Acknowledgement

This publication was made possible by the use of the Factory-AI supercomputer, financially supported by the Ile-de-France Regional Council and the HPC resources of IDRIS under the allocation 2023-AD011014382 made by GENCI.

References

[1] Anurag Arnab, Mostafa Dehghani, Georg Heigold, Chen Sun, Mario Lucic, and Cordelia Schmid. Vivit: A video vision transformer. In *International Conference on Computer Vision*, pages 6816–6826, 2021. 1

[2] Julien Denize, Jaonary Rabarisoa, Astrid Orcesi, and Romain Hérault. Similarity contrastive estimation for image and video soft contrastive self-supervised learning. *Machine Vision and Applications*, 34(6), 2023. 1

[3] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021. 1

[4] Christoph Feichtenhofer, Haoqi Fan, Bo Xiong, Ross B. Girshick, and Kaiming He. A large-scale study on unsupervised spatiotemporal representation learning. In *Conference on Computer Vision and Pattern Recognition*, pages 3299–3309, 2021. 1

[5] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *International Conference on Computer Vision*, pages 9992–10002, 2021. 1

[6] João V. B. Soares, Avijit Shah, and Topojoy Biswas. Temporally precise action spotting in soccer videos using dense detection anchors. In *International Conference on Image Processing*, pages 2796–2800, 2022. 3

[7] Hongyi Zhang, Moustapha Cissé, Yann N. Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. In *6th International Conference on Learning Representations*, 2018. 2

[8] Xin Zhou, Le Kang, Zhiyu Cheng, Bo He, and Jingyu Xin. Feature combination meets attention: Baidu soccer embeddings and transformer based temporal detection. *arXiv*, abs/2106.14447, 2021. 1