Unsupervised Visual Representation Learning by Tracking Patches in Video Supplementary Material

A. Evaluation Pipeline

In this section, we introduce the evaluation details when transferring the pretrained models to the downstream tasks.

Action recognition We follow the common practice [4] to construct an action recognition model. Specifically, we adopt a CNN encoder to extract the spatial-temporal feature from the raw video clip. The CNN encoder can be initialized from the pretrained models of various proxy tasks, e.g., VCOP [3], 3DRot [1] or our CtP. The feature extracted by the CNN encoder is averaged in both spatial and temporal dimensions, which leads to a feature vector of size 512 (for R3D-18 and R(2+1)D-18 in our work). We append a one-layer linear classifier based on the averaged feature.

The entire action recognition model is finetuned on the target datasets. The input video clip has a temporal coverage of 32 frames. For UCF-101 and HMDB-51 datasets, we sample 16 frames with a temporal stride of 2. For Something-Something dataset, we successively sample 32 frames since this dataset emphasizes more fine-grained temporal relationships. The standard data augmentation strategy is applied in the training period, including random cropping, horizontal flip and color jitters. During inference time, the video frames are resized to a spatial resolution of 171×128, and we center crop the regions of shape 112×112. For each video, we temporal-uniformly pick 10 clips. The final classification decision of the video is averaged by the prediction results of these clips.

The optimization lasts for 150 epochs with standard SGD algorithms. The learning rate is started from 0.01 and it is gradually decayed by a factor of 0.1 at 60th epoch and 120th epoch. The weight decay and the momentum value is set to 5×10^{-4} and 0.9, respectively. After the optimization, we adopt the model produced in the last epoch for evaluation.

Video clip retrieval In the video clip retrieval task, the pretrained CNN encoder is directly used without further finetuning. We follow the same implementation as in VCOP [3]. The output feature of the CNN encoder is transferred to a fixed shape of $2 \times 3 \times 3 \times 512$ by an adaptive pooling operation, where each dimension denotes the temporal size, height, width and channels, respectively. For each video, we uniformly sample 10 clips and use the averaged feature of these 10 clips to represent the video-level feature. The cosine similarity between two different videos is considered as the distance metric. If a test video shares the same class label with one of its k-nearest training videos, it will be viewed as a correct retrieval.

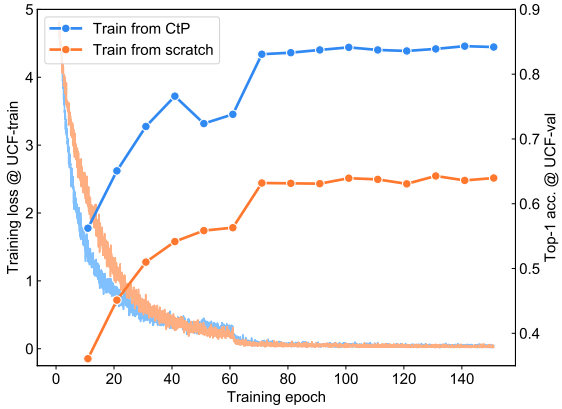

Figure 1: Training loss curves and validation accuracy curves on UCF-101 dataset. The training process of action recognition task lasts for 150 epochs. We test the classifier in every 10 epochs.

B. Analysis of Action Recognition

In this section, we have a closer look at the training process of action recognition tasks. Generally, we consider two configurations when building a classifier on the UCF-101 dataset: (1) initialization from the CtP-learned model or (2) training from scratch. The training loss curve and test accuracy of both settings are shown in Figure 1. Compared with the baseline, our CtP pretraining enables a faster convergence speed in the early epochs. More importantly, even if both settings achieve close values of training losses in the final epoch, our CtP-pretrained model is significantly better than the baseline model in terms of validation accuracy. This is because our CtP game introduces motion priors about object movements, which helps the model quickly capture the moving targets and prevent overfitting.

Table 1: Evaluation on the validation set of Kinetics-400.

Backbone	Pretraining	K400-val Acc. (%) Top-1	Top-5
R3D-18	None	64.1	85.3
R3D-18	CtP	**65.0**	**85.7**

Table 2: Ablation analysis on the backbone stride in the pretraining stage. The models are pretrained on the UCF-101 dataset and transfer to the action recognition task.

Backbone	Stride	Top-1 Acc. (%) UCF-101	HMDB-51
R3D-18	16	83.5	52.5
R3D-18	8	**83.9**	**53.6**

Table 3: Ablation analysis on the temporal squeezing operation. The models are pretrained on the UCF-101 dataset and transfer to the action recognition task.

Backbone	Squeeze?	Top-1 Acc. (%) UCF-101	HMDB-51
R3D-18		82.9	52.3
R3D-18	✓	**83.9**	**53.6**

C. Evaluation on Kinetics Dataset

As suggested by the common practice, most of our experiments are conducted on UCF-101 and HMDB-51 datasets. However, these two datasets are relatively small (less than 10k videos). One may raise a natural concern, whether the CtP pretraining can still improve the performance when the downstream task has plenty of data. In order to address the concern, in this section, we transfer the CtP-pretrained model to the action recognition task on Kinetics-400 dataset. It has about 220k labeled videos for training and 18k videos for validation.

The experimental results are presented in Table 1. Our CtP-pretrained model outperforms the baseline model by a considerable margin. It proves that the downstream task on the large-scale dataset can still benefit from the CtP pretraining.

D. Analysis of Backbone Stride

In the pretraining stage, we introduce a slight modification to the backbone CNN encoder. The spatial stride of the last residual block is set to 1 instead of 2, i.e., the total spatial stride is decreased to 8. This strategy will lead to a finer feature map in the spatial dimensions, which has proven effective in object tracking literature. Note that the architecture is only modified in the pretraining stage. We reset the backbone is to the standard configuration for fair evaluations in the downstream tasks. The ablation results are presented in Table 2. It clearly shows that this modification is helpful to video representation learning.

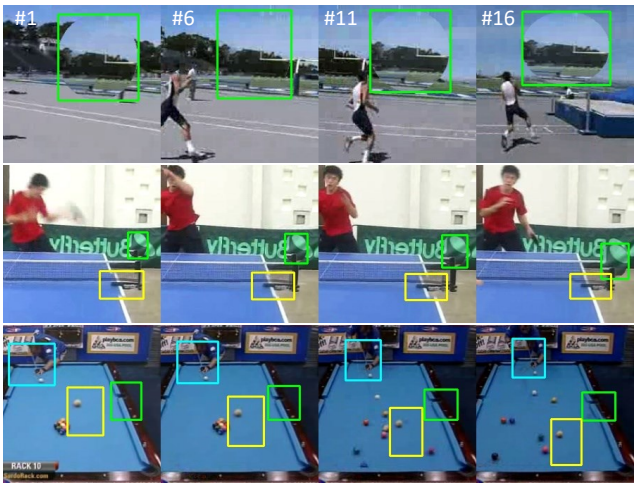

Figure 2: Generated training examples.

E. Analysis of Scaling Factors

When calculating the loss function, we apply four constant scaling factors σ to the prediction targets. It is a widely-used strategy in the area of object tracking and helps to balance the penalties of different regression terms. We provide the additional analysis on the choice of scaling factors, as shown in Table 4. The performance is rather stable when the scaling factors are drawn from a reasonable range.

F. Analysis of Temporal Squeezing

In our framework, we squeeze the temporal dimension of the extracted features before they are fed into the RoI align module. This operation can aggregate the information of the entire video and encourage the model to learn a more compact feature representation. As shown in Table 3, the action recognition results slightly degrade without this operation.

G. Details of Trajectory Synthesis

In this section, we present the details of trajectory synthesis. Firstly, we determinate the bounding box location in the starting frame, which is randomly picked within a size range of $[16, 64]$ and an aspect ration range of $[1/2, 2]$. Secondly, we choose several key frames (3 to 5 in our experiments), and decide the bounding boxes in these key frames. To avoid the vast change of the synthetic trajectory, we add two constraints, namely speed constraint and scale constraint. For the former one, we force the position differences between the boxes in two neighbour key frames to be less than $3\Delta T$ pixels, where ΔT is the difference of two

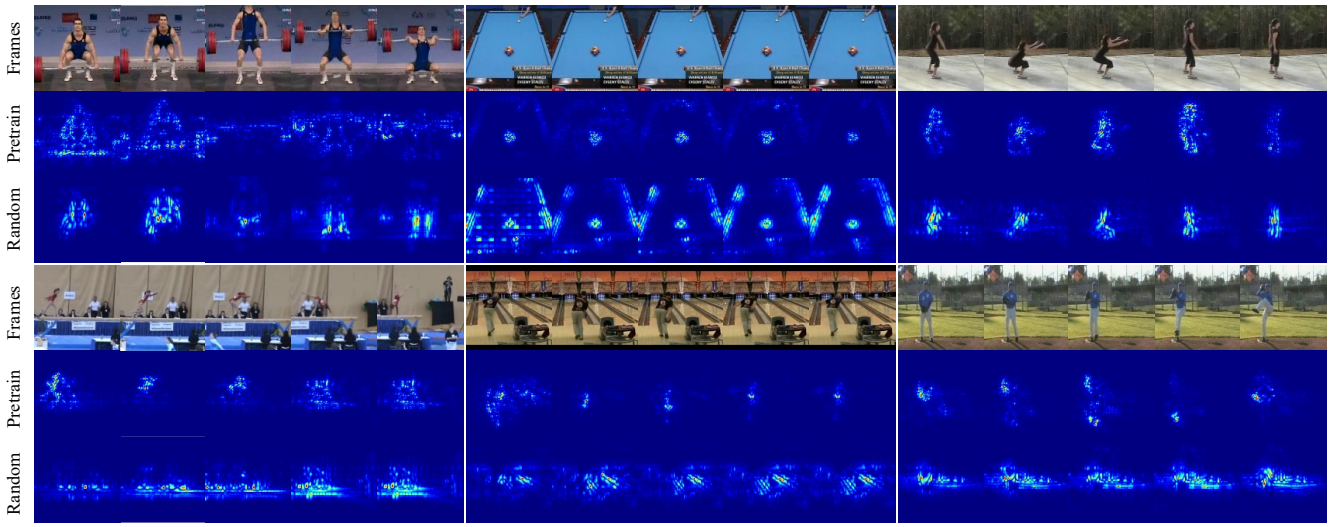

Figure 3: More visualization examples. The important pixels are highlighted by Guided GradCAM algorithms [2].

scaling factors		(0.8/0.04)	(0.8/0.08)	(0.8/0.02)	(1.0/0.05)
Cls. Acc. (%)	U101	83.9	83.3	83.6	83.2
	H51	53.6	52.9	55.0	54.8

Table 4: Ablation analysis on the impact of scaling factors.

Speed	1	3	5	15
U101	80.6	83.9	83.4	80.6
H51	48.6	53.6	54.0	49.3

Scale	0.010	0.025	0.040	0.100
U101	83.3	83.9	83.5	84.3
H51	54.3	53.6	53.2	54.2

Table 5: Ablation analysis on the patch speed and scaling ratio (described in L543.). The blue columns denote the default setting.

frame indexes. While for the scale constraint, the size ratio should lie in a range of $[\exp(-0.025\Delta T), \exp(0.025\Delta T)]$. Finally, the bounding boxes in the rest of frames are linearly interpolated between two key frames. Once we have the trajectory of bounding boxes, we can copy and paste the image patch to fill this trajectory. In Fig. 2, we provide some generated examples, with one to three synthetic trajectories.

We also analyzes how the hyper-parameters of two constraints affect the pretraining performance. As shown in Table 5, the performances are harmed if the speed is too fast or too slow. Besides, CtP pretraining is not sensitive to the scale changes.

H. More Visualizations

Due to the space limitation, we only show two sequences in Section 5.7. Here, we further present more visualization examples in Figure 3. The classifier trained from the CtP model can well capture the salient targets and avoid overfitting to the background regions.

References

[1] Longlong Jing, Xiaodong Yang, Jingen Liu, and Yingli Tian. Self-supervised spatiotemporal feature learning via video rotation prediction. *arXiv preprint arXiv:1811.11387*, 2018. 1

[2] Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *ICCV*, pages 618–626, 2017. 3

[3] Dejing Xu, Jun Xiao, Zhou Zhao, Jian Shao, Di Xie, and Yueting Zhuang. Self-supervised spatiotemporal learning via video clip order prediction. In *CVPR*, pages 10334–10343, 2019. 1

[4] Dahua Lin Yue Zhao, Yuanjun Xiong. Mmaction. https://github.com/open-mmlab/mmaction, 2019. 1