Supplementary material

1 Implementation details

Self-supervised pretrain In self-supervised pretraining stage, we resize each frame to 128×171 and randomly crop images to size of 112×112 in a temporal consistent way. Each input clip has 16 frames with temporal interval of 4. Data augmentations include random color jittering, horizontal flipping and gaussian blurring. We pretrain the model for 200 epochs with an SGD optimizer with an initial learning rate of 0.003, weight decay of 1e-4 and momentum of 0.9. Learning rate is divided by 10 at epoch 120 and 160. Batch size is set to 64. We pretrain the model on 8 Tesla V100 GPUs. Pretrained weights at epoch 190 is selected for evaluation. Size of the MoCo negative queue is set to 16,384. We pretrain on UCF101 training split in all ablation studies and provide pretraining result on large-scale Kinetics400 for performance comparison with previous works. We set τ, τ_{tc}, θ, λ_1, λ_2 to 0.07, 0.5, 0.05, 1.0 and 1.0, respectively.

Supervised finetuning During supervised finetuning, we replace the nonlinear projection head in pretraining stage with a one-layer classification layer. The backbone is initialized with the pretrained weights while the classification head is randomly initialized. Similar as pretraining stage, each clip contains 16 frames sampled at a pace equals to 2. Each frame is first resized to 128×171 and then randomly cropped to size of 112×112 in a temporal consistent way. Data augmentations include color jittering and horizontal flipping. We finetune all layers for 150 epochs using a SGD optimizer with momentum of 0.9. Learning rate is set to 0.05 and divided by 10 at epoch 50 and 100. Batchsize is set to 64.

Finetuning testing After finetuning, we test classification accuracy on the test splits. 10 16-frame clips are temporarily and uniformly sampled in each video with a pace equals to 2. No data augmentation is applied. Each frame is first resized to 128×171 and then center cropped to size of 112×112. Classification probabilities are averaged among the 10 clips for each video.

Video retrieval We compute average features of 10 clips uniformly sampled from each video using the pretrained backbone. Setting is the same as the *Finetuning testing* stage except that we use a pace equals to 4 to be consistent with the pretraining stage. Feature vectors are computed through spatial-temporal average pooling the last layer output of the backbone and then normalized. Cosine similarity is adopted. We conduct video retrieval on UCF101 to retrieve test videos using training videos and calculate the top-k accuracy ($k = 1, 5, 10, 20, 50$).

2 Comparison with shuffling order prediction

We compare our method with shuffling order prediction baseline in Table 1. Improvement on HMDB51 dataset is much larger than that on UCF101, showing our model has a much better transferring ability across different dataset since model is pretrained on UCF101 training split.

Table 1: Comparison between order prediction (OP) and our proposed shuffle-rank (SR). Models are pretrained on UCF101 training set.

	UCF101	HMDB51
SimCLR+OP	75.39	32.38
SimCLR+SR	76.82	40.05

3 More experiment results on inter-intra variance

Continuing our analysis in ablation study section, we provide more experiment results of inter-intra variance on HMDB51 datasets for reference. As can be seen in Table 2, the trend of inter/intra variance changes are consistent on both UCF101 and HMDB51.

Table 2: Comparison of inter-instance variance, intra-instance variance and instance discrimination factor. Results are multiplied by 100 for demonstration. Each cell's increasing(Red) and decreasing(Green) times are compared to the cell above it.

	R3D			R(2+1)D			S3D-G		
	inter-v.	intra-v.	discrim.	inter-v.	intra-v.	discrim.	inter-v.	intra-v.	discrim.
UCF101									
SimCLR	28.8	0.8	34.3	28.2	0.8	33.6	73.1	3.0	24.7
+SR	22.6	$11.3^{\uparrow 13.6\times}$	$2.0^{\downarrow 17.2\times}$	23.0	$11.2^{\uparrow 13.4\times}$	$2.0^{\downarrow 16.8\times}$	51.7	$11.7^{\uparrow 4.0\times}$	$4.4^{\downarrow 5.6\times}$
+SR+TC	42.3	$5.8^{\downarrow 1.9\times}$	$7.3^{\uparrow 3.7\times}$	38.0	$8.0^{\downarrow 1.4\times}$	$4.8^{\uparrow 2.4\times}$	78.7	$4.9^{\downarrow 2.4\times}$	$16.2^{\uparrow 3.7\times}$
HMDB51									
SimCLR	26.6	0.3	85.8	25.8	0.3	83.3	71.1	1.2	58.3
+SR	33.7	$5.8^{\uparrow 18.7\times}$	$5.8^{\downarrow 14.8\times}$	32.2	$5.1^{\uparrow 16.4\times}$	$6.4^{\downarrow 13.0\times}$	61.8	$4.8^{\uparrow 3.9\times}$	$12.9^{\downarrow 4.5\times}$
+SR+TC	43.0	$2.9^{\downarrow 2.0\times}$	$14.8^{\uparrow 2.6\times}$	40.0	$3.8^{\downarrow 1.3\times}$	$10.6^{\uparrow 1.7\times}$	73.3	$2.5^{\downarrow 1.9\times}$	$29.0^{\uparrow 2.2\times}$

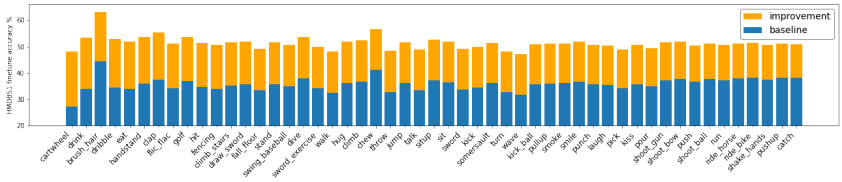


Figure 1: Per-class improvement on HMDB51 downstream supervised finetuning task with SimCLR baseline and R(2+1)D backbone. Model is unsupervisedly pretrained on UCF101 training split and θ is chosen as 0.01. The category labels along the horizontal coordinate follows decreasing order of absolute improvement magnitudes upon the baseline (height of orange bars).

4 Result of different number of segments

We ablated the number of segments in Table 3. Model is equipped with R(2+1)D backbone and pretrained on UCF101 training split. As input clip length is the same, increasing number of segments also increases the similarity between temporarily neighboring segments, which largely increases the difficulty for shuffle-rank task. We find increase segment number from 2 to 4 does not improve the performance much, but 8 segments make the loss not converging, which is caused by the difficulty of the task.

Number of Segment	UCF101	HMDB51
2	79.01	45.37
4	77.32	46.05
8	*null*	*null*

Table 3: Finetune task result on different number of segments with R(2+1)D backbone pretrained on UCF101 training split. *null* means loss can not converge in training time.

5 Per-class improvement result

Per-class Analysis We provide per-class accuracy improvement result of our method upon baseline SimCLR in Figure 1. Remind that in HMDB51 dataset, highly-improved categories such as cartwheel, drink, eat and handstand usually contain decomposable unrepetitive actions with large motion changes, while categories like running, riding horses, shake-hands consist of many repetitive actions thus are less improved by our method. All actions have been improved. These results supported our motivation that representing videos by sub-features are useful.

6 Backbone architecture

We provide architectures of R3D and R(2+1)D in Table 4. As the structure of S3D-G is too complicated to be put in a table, we ask the reader to refer to [3] for more details. We use the same implementation in our code.

Table 4: architecture of R3D and R(2+1)D we used in experiments.

layer name	output size	R3D	R(2+1)D
conv1	$L \times 56 \times 56$	$3 \times 3 \times 3$, 64, stride $1 \times 2 \times 2$	
conv2_x	$L \times 56 \times 56$	$\begin{bmatrix} 3 \times 3 \times 3, 64 \\ 3 \times 3 \times 3, 64 \end{bmatrix} \times 1$	$\begin{bmatrix} 1 \times 3 \times 3, 64 \\ 3 \times 1 \times 1, 64 \end{bmatrix} \times 2$
conv3_x	$\frac{L}{2} \times 28 \times 28$	$\begin{bmatrix} 3 \times 3 \times 3, 128 \\ 3 \times 3 \times 3, 128 \end{bmatrix} \times 1$	$\begin{bmatrix} 1 \times 3 \times 3, 128 \\ 3 \times 1 \times 1, 128 \end{bmatrix} \times 2$
conv4_x	$\frac{L}{4} \times 14 \times 14$	$\begin{bmatrix} 3 \times 3 \times 3, 256 \\ 3 \times 3 \times 3, 256 \end{bmatrix} \times 1$	$\begin{bmatrix} 1 \times 3 \times 3, 256 \\ 3 \times 1 \times 1, 256 \end{bmatrix} \times 2$
conv5_x	$\frac{L}{8} \times 7 \times 7$	$\begin{bmatrix} 3 \times 3 \times 3, 512 \\ 3 \times 3 \times 3, 512 \end{bmatrix} \times 1$	$\begin{bmatrix} 1 \times 3 \times 3, 512 \\ 3 \times 1 \times 1, 512 \end{bmatrix} \times 2$
	$1 \times 1 \times 1$	spatial-temporal global average pooling	

7 Measure inter and intra variances

We provide our mathematical formulations of inter-variance and intra-variance for calculating inter and intra variances. Formally, let μ_k represent the mean embedded feature of the k-th video. We use S_k to represent the hierarchical relation between clips $\{c_i\}_{i=1}^M$ and videos $\{V_i\}_{i=1}^N$, i.e. $S_k = \{i | c_i \in V_k\}$. We define intra-variance as the average variances within clip features sampled from the same video, and inter-variance as the average pairwise distance between videos:

$$\mu_k = \frac{1}{S_k} \sum_{i \in S_k} \boldsymbol{z}_i \tag{1}$$

$$\sigma_{intra} = \frac{1}{N} \sum_{k=1}^{N} \frac{1}{|S_k|} \sum_{i \in S_k} \|\boldsymbol{z}_i - \mu_k\|_2^2 \tag{2}$$

$$\sigma_{inter} = \frac{1}{N(N-1)} \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} \|\mu_i - \mu_j\|_2^2 \tag{3}$$

where $\boldsymbol{z}$ is the encoded feature vector as introduced in the paper. This formulation is adapted from Liu et al. [1].

8 Effect of contrastive loss on inter and intra variance

We provide theoretical analysis to our statement in preliminary section that contrastive loss suppresses the learning of σ_{inter} and encourages learning σ_{inter}. Our analysis follows discussion of Tongzhou and Phillip [2]. Following our definition in preliminary section, clip contrastive loss can be written in the form of $\mathcal{L}_c = -\frac{1}{M} \sum_{i=1}^{M} \log \frac{\exp(\frac{\boldsymbol{z}_i \cdot \boldsymbol{z}_{i^+}}{\tau})}{\sum_{k=1}^{M} \mathbf{1}_{[\mathbf{k} \neq \mathbf{i}]} \exp(\frac{\boldsymbol{z}_i \cdot \boldsymbol{z}_k}{\tau})}$, which can be transformed into:

$$\mathcal{L}_c = -\frac{1}{M} \sum_{i=1}^{M} \frac{\boldsymbol{z}_i \cdot \boldsymbol{z}_{i^+}}{\tau} + \frac{1}{M} \sum_{i=1}^{M} \left[\log \left(\exp(\frac{\boldsymbol{z}_i \cdot \boldsymbol{z}_{i^+}}{\tau}) + \sum_{k=1, k \notin \{i, i^+\}}^{M} \exp(\frac{\boldsymbol{z}_i \cdot \boldsymbol{z}_k}{\tau}) \right) \right] \tag{4}$$

where M is the number of sampled clips, $\boldsymbol{z}$ is normalized encoded clip feature vector and τ is the temperature parameter. As explained in [2], since $\sum_{k=1,k\notin\{i,i^+\}}^{M} \exp(\boldsymbol{z}_i \cdot \boldsymbol{z}_k/\tau)$ is always positive and bounded below, the loss favors smaller $-\frac{1}{M}\sum_{i=1}^{M} \boldsymbol{z}_i \cdot \boldsymbol{z}_{i^+}/\tau$, which is equivalent to decreasing σ_{intra} as the sample size M goes to infinity. On the other hand, when fixing σ_{intra}, minimizing the loss is equivalent to decreasing

$$\frac{1}{M}\sum_{i=1}^{M}\left[\log(t_i + \sum_{k=1,k\notin\{i,i^+\}}^{M} \exp(\boldsymbol{z}_i \cdot \boldsymbol{z}_k/\tau))\right] \tag{5}$$

where $t_i = \exp(\boldsymbol{z}_i \cdot \boldsymbol{z}_{i^+}/\tau)$ is a random variable when σ_{intra} is fixed. Hence, equation 5 measures similarity between features from different instances. Minimizing equation 5 is thus equivalent to minimizing inter-variance σ_{inter}. Therefore, contrastive loss encourages decreasing intra-variance and increasing inter-variance.

9 Training algorithm

9.1 Training SimCLR

Algorithm 1: Training SimCLR

Input: batch size N, backbone f, clip projector f_c, dual projector f_r, videos V;
while *training not converge* **do**
  sample minibatch $\{v_i\}_{i=1}^{N}$ from V;
  for *all* $i \in \{1, 2, ...N\}$ **do**
    sample and augment two clips c_i^1 and c_i^2 from v_i ;
    $\boldsymbol{h}_i^1 = f(c_i^1)$;
    $\boldsymbol{h}_i^2 = f(c_i^2)$;
    `/* clip projection */`
    $\boldsymbol{z}_{2i-1} = f_c(\boldsymbol{h}_i^1)$;
    $\boldsymbol{z}_{2i} = f_c(\boldsymbol{h}_i^2)$;
    `/* dual projection */`
    $\boldsymbol{r}_{2i-1} = f_r(\boldsymbol{h}_i^1)$;
    $\boldsymbol{r}_{2i} = f_r(\boldsymbol{h}_i^2)$;
    `/* shuffling and dual projection */`
    augment c_i^1 into s_i and randomly shuffle s_i into $\hat{s}_i$;
    $\boldsymbol{q}_i = f_r(f(s_i))$;
    $\boldsymbol{p}_i = f_r(f(\hat{s}_i))$;
  end
  calculate $\mathcal{L}_c$ on $\{\boldsymbol{z}\}_{i=1}^{2N}$;
  calculate $\mathcal{L}_{tc}$ on $\{\boldsymbol{r}\}_{i=1}^{2N}$;
  calculate $\mathcal{L}_{rank}^{unaug}$ on $\{\boldsymbol{r}_{2i-1}, \boldsymbol{p}_i\}_{i=1}^{N}$ and $\mathcal{L}_{rank}^{aug}$ on $\{\boldsymbol{q}_i, \boldsymbol{p}_i\}_{i=1}^{N}$;
  update f, f_c and f_r using stochastic gradient descent;
end

9.2 Training MoCo

Unlike SimCLR, negative examples in MoCo come from the dictionary $\{\boldsymbol{k}_i\}_{i=1}^{m}$ while positive example is the sampled augmented clip $\boldsymbol{k}^+$. m is set to 16384. So contrastive loss is denoted as :

$$\mathcal{L}_c = -\frac{1}{N}\sum_{i=1}^{N}\frac{\exp(\boldsymbol{z}_i \cdot \boldsymbol{k}_i^+/\tau)}{\exp(\boldsymbol{z}_i \cdot \boldsymbol{k}_i^+/\tau) + \sum_{j=1}^{m}\exp(\boldsymbol{z}_i \cdot \boldsymbol{k}_j/\tau)} \tag{6}$$

As for temporal coherent contrast, an extra dictionary for dual representations $\{\boldsymbol{d}_i\}_{i=1}^m$ is maintained. Dual representations are denoted as $\boldsymbol{r}$ instead of $\boldsymbol{z}$. Replacing dot product similarity, dictionary $\boldsymbol{k}$, projected feature $\boldsymbol{z}$ with tc-sim, $\boldsymbol{d}$ and $\boldsymbol{r}$ in equation 6 leads us to MoCo's temporal coherent contrast loss:

$$\mathcal{L}_{tc} = -\frac{1}{N}\sum_{i=1}^{N}\frac{\exp(\text{tc-sim}(\boldsymbol{r}_i, \boldsymbol{d}_i^+)/\tau)}{\exp(\text{tc-sim}(\boldsymbol{r}_i, \boldsymbol{d}_i^+)/\tau) + \sum_{j=1}^{m}\exp(\text{tc-sim}(\boldsymbol{r}_i, \boldsymbol{d}_j)/\tau)} \tag{7}$$

We provide training algorithm on MoCo here.

Algorithm 2: Training MoCo

Input: batch size N, backbone f_q f_k, clip projectors g_q g_k, dual projectors h_q h_k, videos V, clip feature dictionary $\{\boldsymbol{k}_i\}_{i=1}^m$ and dual feature dictionary $\{\boldsymbol{d}_i\}_{i=1}^m$;

while *training not converge* **do**

- sample minibatch $\{v_k\}_{k=1}^N$ from V;
- **for** *all* $i \in \{1, 2, ...N\}$ **do**
  - sample and augment two clips c_i^1 and c_i^2 from v_i, c_i^2 is treated as positive pair of c_i^1 ;
  - $\boldsymbol{b}_i^1 = f_q(c_i^1)$;
  - $\boldsymbol{b}_i^2 = f_k(c_i^2)$;
  - `/* clip projection */`
  - $\boldsymbol{z}_i = g_q(\boldsymbol{b}_i^1)$;
  - $\boldsymbol{k}_i^+ = g_k(\boldsymbol{b}_i^2)$;
  - $\boldsymbol{k}_i^+ = \boldsymbol{k}_i^+.detach()$; `// stop gradient`
  - `/* dual projection */`
  - $\boldsymbol{r}_i = h_q(\boldsymbol{b}_i^1)$;
  - $\boldsymbol{d}_i^+ = h_k(\boldsymbol{b}_i^2)$;
  - $\boldsymbol{d}_i^+ = \boldsymbol{d}_i^+.detach()$; `// stop gradient`
  - `/* shuffling and dual projection */`
  - augment c_i^1 into s_i and randomly shuffle s_i into $\hat{s}_i$;
  - $\boldsymbol{q}_i = h_q(f_q(s_i))$;
  - $\boldsymbol{p}_i = h_q(f_q(\hat{s}_i))$;
- **end**
- calculate $\mathcal{L}_c$ on positive pairs $\{\boldsymbol{z}_i, \boldsymbol{k}_i^+\}_{i=1}^N$ and negative queue $\{\boldsymbol{k}_i\}_{i=1}^m$;
- calculate $\mathcal{L}_{tc}$ on positive pairs $\{\boldsymbol{r}_i, \boldsymbol{d}_i^+\}_{i=1}^N$ and negative queue $\{\boldsymbol{d}_i\}_{i=1}^m$;
- calculate $\mathcal{L}_{rank}^{unaug}$ on $\{\boldsymbol{r}_i, \boldsymbol{p}_i\}_{i=1}^N$ and $\mathcal{L}_{rank}^{aug}$ on $\{\boldsymbol{q}_i, \boldsymbol{p}_i\}_{i=1}^N$;
- momentum update f_k, g_k and h_k by f_q, g_q and h_q respectively;
- update f_q, g_q and h_q using stochastic gradient descent;
- $enqueue(\{\boldsymbol{k}_i\}_{i=1}^m, \{\boldsymbol{k}_i^+\}_{i=1}^N)$;
- $dequeue(\{\boldsymbol{k}_i\}_{i=1}^m)$;
- $enqueue(\{\boldsymbol{d}_i\}_{i=1}^m, \{\boldsymbol{d}_i^+\}_{i=1}^N)$;
- $dequeue(\{\boldsymbol{d}_i\}_{i=1}^m)$;

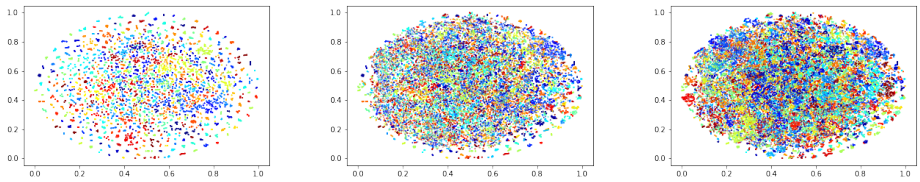
end

10 Feature distribution comparison

We visualize the feature distribution encoded by baseline SimCLR and our pretrained models on UCF101 test set. As shown, feature distribution of baseline SimCLR is very sparse, with large margin between different classes. With our shuffle-rank pretext task, intra-variance becomes larger, resulting in a denser distribution. Further equipping the model with temporal coherent contrastive loss makes the clustering borders clearer. This is consistent with our inter-intra variance motivation.

11 Attention visualization

To qualitatively measure the learned features, we draw heatmaps output from last layer of pretrained models. Specifically, we apply average pooling on last layer feature maps along both feature channel and temporal channel, i.e. from a size of (C, T, H, W) into (H, W), to obtain a heatmap. Such heatmap is then added to each frame of the input clip for visualization. As shown in Figure 3, baseline SimCLR can be easily distracted by surrounding backgrounds, while our proposed method always focus on areas with motion semantics.



(a) SimCLR (b) SimCLR+SR (c) SimCLR+SR+TC

Figure 2: t-SNE visualization of features encoded by pretrained models. Models are pretrained on UCF101 train split and features are computed on UCF101 test split.

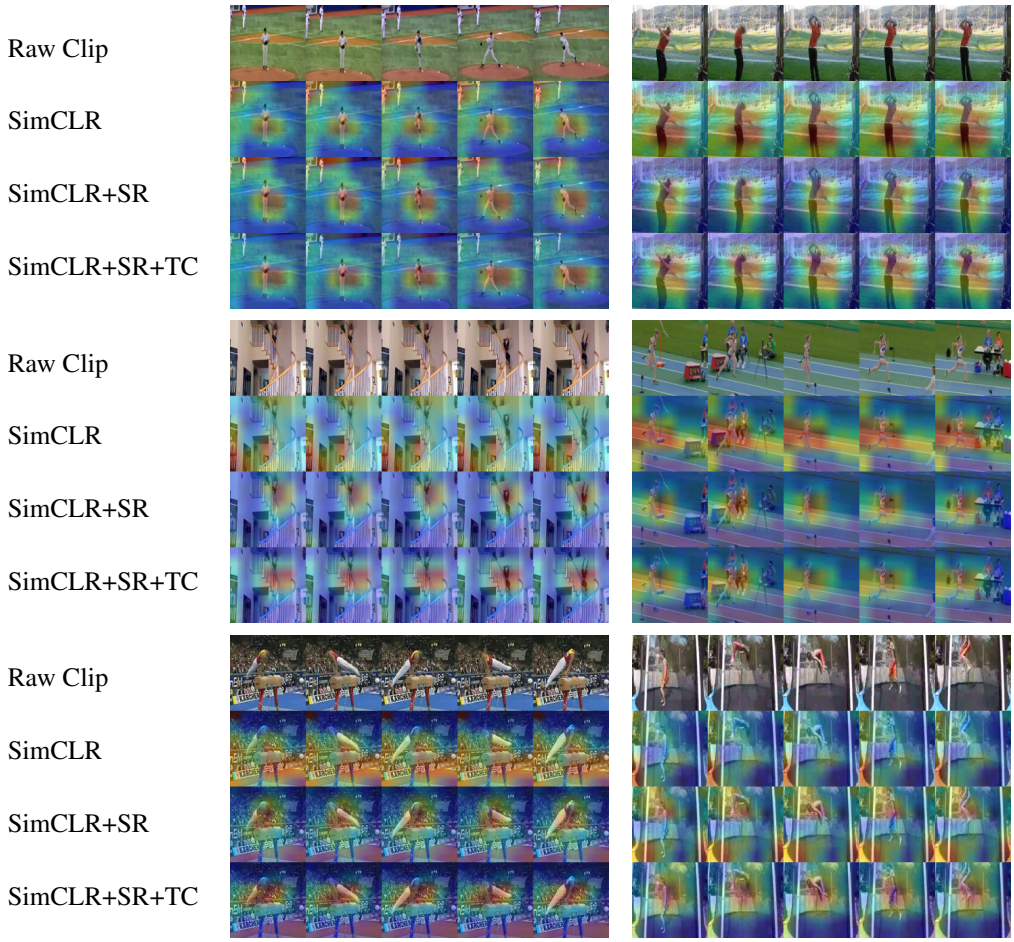


Figure 3: Heatmap visualization. The warmer the color is, the larger the response is. Zoom in for visualization.

12 Video retrieval visualization

In Figure 4, we visualize the video retrieval result of the integrated pretrained model (SimCLR+SR+TC). We pretrain the model on Kinetics-400 and conduct retrieval experiment on UCF101. It can be observed that our retrieval results are not perfect in that videos from different categories can be retrieved if having similar actions and contents, e.g. image 3 (ApplyLipstick) of ApplyEyeMakeup, image 3 (Shotput) of GolfSwing, image 3 (ParallelBars) of BalanceBeam and image 3 (BalanceBeam) of ParallelBars.

References

[1] B. Liu, Y. Cao, Y. Lin, Q. Li, Z. Zhang, M. Long, and H. Hu. Negative margin matters: Understanding margin in few-shot classification. In *ECCV*, 2020.

[2] W. Tongzhou and I. Phillip. Understanding contrastive representation learning through alignment and uniformity on the hypersphere. In *ICML*, 2020.

[3] S. Xie, C. Sun, J. Huang, Z. Tu, and K. Murphy. Rethinking spatiotemporal feature learning: Speed-accuracy trade-offs in video classification. In *ECCV*, 2018.

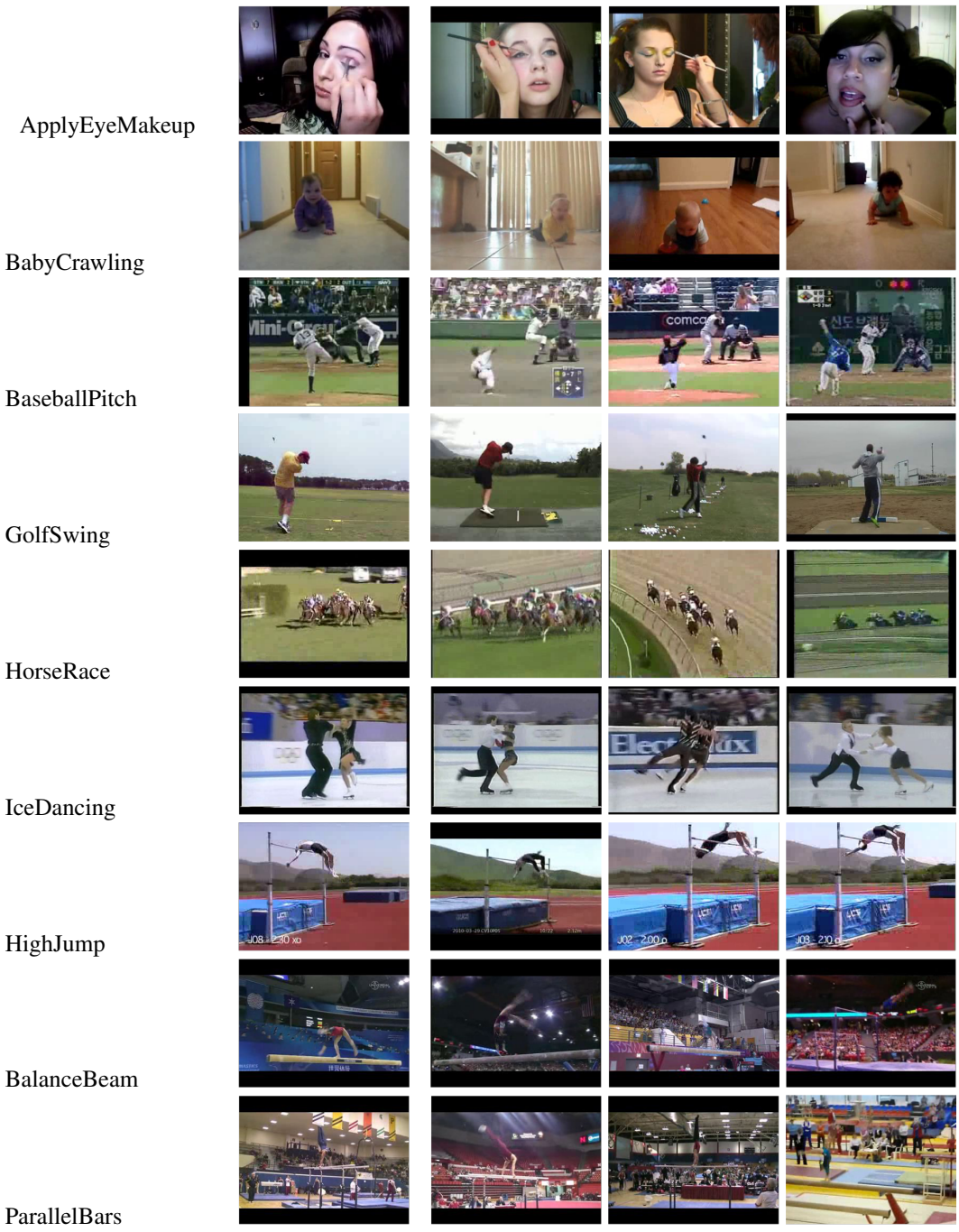


Figure 4: Video retrieval visualization. The first and second column are categories and images of test instance, respectively. The rightmost 3 columns are nearest retrieval results. We select an image from the video for demonstration.