Hierarchical Contrastive Motion Learning for Video Action Recognition

SUPPLEMENTARY MATERIAL

Xitong Yang[1]
xyang35@cs.umd.edu
Xiaodong Yang[2]
https://xiaodongyang.org
Sifei Liu[2]
https://www.sifeiliu.net
Deqing Sun[3]
https://deqings.github.io
Larry Davis[1]
http://users.umiacs.umd.edu/~lsd
Jan Kautz[2]
https://jankautz.com

[1] University of Maryland, College Park
[2] NVIDIA
[3] Google

Section 1 presents additional experiments for comparing the motion features learned at different levels. Section 2 shows qualitative results using Grad-CAM++. Sections 3 and 4 respectively provide more details of our approach and the training on each specific dataset.

1 Motion at different levels

Here we provide more details of the KNN video classification experiment discussed in Section 4.3 of the paper. As shown in Table 7, we compare the classification accuracy for motion features learned at different levels. It is not surprising that the motion features at higher levels achieve higher accuracy than the ones at lower levels as the former possess more useful semantics of motion dynamics for the video action recognition task. More interestingly, we find that the low-level motion features can obtain relatively high accuracy for some action classes with apparent moving patterns (e.g., “turning the camera left"). This indicates that the low-level motion features are capable of extracting elementary movements from raw video frames. On the other hand, the motion features at higher levels can recognize the actions that require finer understanding of high-level motion semantics (e.g., “pulling two ends of something so that it separates into two pieces"). This finding validates the motion hierarchy illustrated in Figure 1 of the paper.

	Acc (%)	Classes with the largest relative gains
Low-level	11.6	“turning the camera left" (73.8) “turning the camera right" (71.7) “turning the camera upwards" (58.3)
Mid-level	15.2	“showing a photo to the camera" (0 → 19.4) “showing smth behind smth" (0 → 12.2) “poking a stack of smth" (0 → 9.5)
High-level	21.7	“pulling two ends of smth" (0 → 14.0) “tipping smth with smth in it over" (0 → 12.5) “sprinkling smth onto smth" (0 → 9.1)

Table 7: Comparison of classification accuracy using motion features learned at different levels (accuracy for random output: 0.6%). For the mid/high levels, we show the top-3 classes with the largest relative gains compared with the lower-level motion features. For the low level, we report the top-3 classes with the highest accuracy instead.

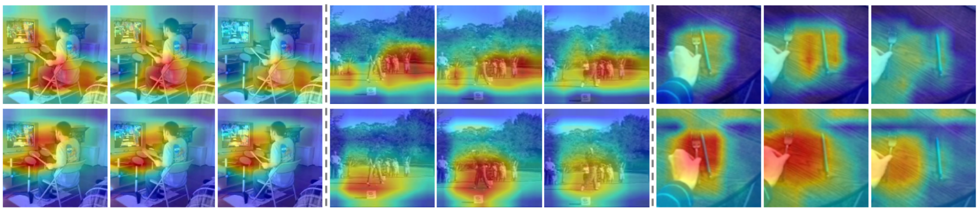
(a) Baseline: PlayingCello Ours: Drumming (b) Baseline: CricketBowling Ours: GolfSwing (c) Baseline: Moving sth closer to something Ours: Moving sth away from something

Figure 6: Visualization of the learned features by Grad-CAM++ [2]. Fonts in green and red indicate correct recognition and misclassification. (a-b) Features learned by our approach (bottom) are more sensitive to the regions with important motion cues. (c) Our motion learning module (bottom) equips the 2D backbone with the ability of reasoning the temporal order of video frames.

2 Qualitative Result

To qualitatively verify the impact of the learned motion features, we utilize Grad-CAM++ [2] to visualize the class activation map of the last convolution layer. Figure 6 shows the comparison between baseline and our model with the backbone R2D-50 on UCF-101 and Something-V1. Our model attentions more on the regions with informative motion, while the baseline tends to be distracted by the static appearance. For instance, in Figure 6(a), our method focuses on the moving hands of the person, while the baseline concentrates on the static human body. Our motion learning module also equips the 2D backbone with effective temporal modeling ability. As shown in Figure 6(c), our model is capable of reasoning the temporal order of the video and predicting the correct action, while the baseline outputs the opposite prediction result as it fails to capture the chronological relationship.

3 More Details

3.1 Backbones

We adopt the standard convolutional networks R2D-50 [15] and R(2+1)D-101 [14] as the backbones used in our experiments. A few changes are made to improve the computation efficiency, as demonstrated in Table 8. Compared with the original network R(2+1)D-101, our backbone supports higher input resolution and applies bottleneck layers with consistent number of channels. We start temporal striding from $\texttt{res}_3$ rather than $\texttt{res}_4$, and employ the top-heavy design as used in [16] for R(2+1)D-101, i.e., only using temporal convolutions in $\texttt{res}_4$ and $\texttt{res}_5$.

Layer	R2D-50 / R(2+1)D-101	Output Size
`input`	–	T × 224 × 224
$\texttt{conv}_1$	$1\times7\times7$, 64 stride: $1\times2\times2$	T × 112 × 112
$\texttt{res}_2$	$\begin{bmatrix} 1\times1\times1, 64 \\ 1\times3\times3, 64 \\ 1\times1\times1, 256 \end{bmatrix}\times3$	T × 56 × 56
$\texttt{res}_3$	$\begin{bmatrix} 1\times1\times1, 128 \\ 1\times3\times3, 128 \\ 1\times1\times1, 512 \end{bmatrix}\times4$	T/2 × 28 × 28
$\texttt{res}_4$	$\begin{bmatrix} 1\times1\times1, 256 \\ (3\times1\times1, 256) \\ 1\times3\times3, 256 \\ 1\times1\times1, 1024 \end{bmatrix}\times6\,/\,23$	T/4 × 14 × 14
$\texttt{res}_5$	$\begin{bmatrix} 1\times1\times1, 512 \\ (3\times1\times1, 512) \\ 1\times3\times3, 512 \\ 1\times1\times1, 2048 \end{bmatrix}\times3$	T/4 × 7 × 7

Table 8: Details of the architectures of the backbone networks R2D-50 / R(2+1)D-101 used in our experiments.

3.2 Prime Motion Block

Here we describe the prime motion block in more details. As illustrated in Figure 2 in the paper, the prime motion block is wrapped as a residual block [7] such that the motion features $\mathcal{P}$ can be inserted into a backbone network seamlessly. For the cost volume layer, we limit the search range with the maximum displacement $d = 6$ and the stride $s = 2$, which is equivalent to covering a region of 13×13 pixels with a stride 2. To combine the complementary information provided by the cost volumes and the convolutional features (after dimension reduction), we concatenate the two features in channels and then perform a 2D convolution. We use batch normalization [8] and ReLU after each convolutional layer and cost volume layer.

3.3 Sampling Strategy

We denote the predicted motion feature at level l as $\hat{P}^l_{t,k}$, where $t \in \{1, ..., T^l\}$ is the temporal index, and $k \in \{(1,1), (1,2), ...(H^l, W^l)\}$ is the spatial index. The only positive pair is $(\hat{P}^l_{t,k}, P^l_{t,k})$, which is the ground-truth feature that corresponds to the same video and locates at the same position in both space and time as the predicted one. Following the terminology used in [6], we define three types of negative samples for all the prediction and ground-truth pairs $(\hat{P}^l_{t,k}, P^l_{\tau,m})$:
Spatial negatives are the ground-truth features that come from the same video of the predicted one but at a different spatial position, i.e., $k \neq m$. Considering the efficiency, we randomly sample N spatial locations for each video within a mini-batch to compute the loss. So the number of spatial negatives is $(N-1)T^l$.
Temporal negatives are the ground-truth features that come from the same video and same spatial position, but from different time steps, i.e., $k = m, t \neq \tau$. They are the hardest negative samples to classify, and the number of temporal negatives are $T^l - 1$.
Easy negatives are the ground-truth features that come from different videos, and the number of easy negatives are $(B-1)NT^l$, where B is the batch size.

4 Experimental Details

4.1 Datasets

We extensively evaluate our proposed approach on the four benchmarks: Kinetics-400 [1], Something-Something (V1&V2) [5] and UCF-101 [12]. **Kinetics-400** is a large-scale video dataset with 400 action categories. As some videos are deleted by their owners over time, our experiments are conducted on a subset of the original dataset with approximately 238K training videos (~96%) and 196K validation videos (~98%). In practice, we notice a bit accuracy drop for the same model using our collected dataset with fewer training samples, suggesting that our results can be further improved with the full original dataset. **Something-Something (V1&V2)** are more sensitive to temporal motion modeling. Something-V1 contains about 100K videos covering 174 classes, and Something-V2 increases videos to 221K and improves video resolution and annotation quality. **UCF-101** includes about 13K videos with 101 classes. As the number of training videos is small, it is often used for evaluating unsupervised representation learning [3, 10] and transfer learning [11, 13].

4.2 Implementation details

We use the synchronized SGD with a cosine learning rate scheduling [9] and a linear warm-up [4] for all model training. The spatial size of the input is 224×224, randomly cropped and horizontally flipped from a scaled video with the shorter side randomly sampled in [256, 320] pixels for R2D-50, and [256, 340] pixels for R(2+1)D-101. We apply temporal jittering when sampling clips from a video. The balancing weights for the joint training in Eq. (5) are set to $\lambda = 15, \gamma^1 = \gamma^2 = 0.25$, respectively. We describe the training details for different benchmarks as follow.
Kinetics-400. We sample a clip of $T = 16$ frames with a temporal stride 2 for the experiments using the backbone R2D-50 and $T = 32$ frames for those with the backbone R(2+1)D-101. We train all models using the distributed SGD on GPU clusters with 8 clips per GPU. We

set the learning rate per GPU to 0.0025, and linearly scale the learning rate according to the number of GPUs. For the self-supervised training phase, all models are trained for 80 epochs with the first 10 epochs for warm-up, and the global batch normalization (BN) [8] is used to avoid trivial solution. As for the joint training phase, the models are trained for 200 epochs with the first 40 epochs for warm-up, and BN statistics is computed within each 8 clips.
Something-V1&V2. Since this dataset has a higher frame rate than Kinetics-400, we sample a clip of $T = 32$ frames with a temporal stride 1 for all experiments. Models are trained for 150 epochs with the first 50 epochs for warm-up and the learning rate per GPU is also 0.0025.
UCF-101. For the experiments described in Table 4 of the paper, the models are initialized with the weights pre-trained on Kinetics-400 for classification, and then are fine-tuned for 30 epochs with a batch size of 32 and a learning rate of 0.002.

References

[1] Joao Carreira and Andrew Zisserman. Quo vadis, action recognition? A new model and the kinetics dataset. In *CVPR*, 2017.

[2] Aditya Chattopadhyay, Anirban Sarkar, Prantik Howlader, and Vineeth N Balasubramanian. Grad-CAM++: Improved visual explanations for deep convolutional networks. *arXiv:1710.11063*, 2017.

[3] Ali Diba, Vivek Sharma, Luc Van Gool, and Rainer Stiefelhagen. DynamoNet: Dynamic action and motion network. In *ICCV*, 2019.

[4] Priya Goyal, Piotr Dollár, Ross Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. Accurate, large minibatch SGD: Training ImageNet in 1 hour. *arXiv:1706.02677*, 2017.

[5] Raghav Goyal, Samira Ebrahimi Kahou, Vincent Michalski, Joanna Materzynska, Susanne Westphal, Heuna Kim, Valentin Haenel, Ingo Fruend, Peter Yianilos, and Moritz Mueller-Freitag. The "something something" video database for learning and evaluating visual common sense. In *ICCV*, 2017.

[6] Tengda Han, Weidi Xie, and Andrew Zisserman. Video representation learning by dense predictive coding. In *ICCV Workshop*, 2019.

[7] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

[8] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, 2015.

[9] Ilya Loshchilov and Frank Hutter. SGDR: Stochastic gradient descent with warm restarts. In *ICLR*, 2017.

[10] Yue-Hei Ng, Jonghyun Choi, Jan Neumann, and Larry Davis. ActionFlowNet: Learning motion representation for action recognition. In *WACV*, 2018.

[11] Zhaofan Qiu, Ting Yao, Chong-Wah Ngo, Xinmei Tian, and Tao Mei. Learning spatio-temporal representation with local and global diffusion. In *CVPR*, 2019.

[12] Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah. UCF101: A dataset of 101 human actions classes from videos in the wild. *arXiv:1212.0402*, 2012.

[13] Jonathan Stroud, David Ross, Chen Sun, Jia Deng, and Rahul Sukthankar. D3D: Distilled 3D networks for video action recognition. In *WACV*, 2020.

[14] Du Tran, Heng Wang, Lorenzo Torresani, Jamie Ray, Yann LeCun, and Manohar Paluri. A closer look at spatiotemporal convolutions for action recognition. In *CVPR*, 2018.

[15] Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. In *CVPR*, 2018.

[16] Saining Xie, Chen Sun, Jonathan Huang, Zhuowen Tu, and Kevin Murphy. Rethinking spatiotemporal feature learning: Speed-accuracy trade-offs in video classification. In *ECCV*, 2018.