Supplementary Material of "MotionSqueeze: Neural Motion Feature Learning for Video Understanding"

We present additional results and details that are omitted in our main paper due to the lack of space. All our code and data are released online at our project page: http://cvlab.postech.ac.kr/research/MotionSqueeze/

1 Effects of depth-wise separable (DWS) convolutions

We use DWS convolutions rather than standard convolutions to build the feature transformation (FT) layers deeper and wider while saving computational cost. Table 1 shows the results of different forms of FT layers on Something-Something V1 [3] . The accuracy increases as the FT layers become deeper and have wider receptive fields, and the DWS convolutions show the best accuracy-FLOPs tradeoff.

2 Comparison with the CP module [9].

As we mentioned in the main paper, the CP module is the one of the most relevant work to our method in the sense that it leverage correspondences between input video frames. Here we provide more detailed comparisons to it.

Difference in motivation and design. Unlike our MS module, which focuses on extracting effective motion features across consecutive frames, the CP module [9] is designed to capture long-term spatio-temporal relationship within an input video [11] by computing a non-local correlation tensor across all frames. The CP module selects k most likely corresponding features in the correlation tensor with an 'arg top-k' operation, and the operation thus makes the correlation tensor non-differentiable.

Performance comparison. We have already shown in Table 1 of the main paper that the result of our method is better than that of the CP module (from the original paper [9]) on Something-Something V2 [3]. The comparison, however, may not be totally fair in the sense that the backbone and the other experimental settings are not the same. For an apples-to-apples comparison between the MS module and the CP module, we conduct an additional experiment using the same backbone and setup. We re-implement the CP module in Pytorch based on the official Tensorflow code[1]. As a baseline network, we use ImageNet pre-trained TSM ResNet-18 using 8 input frames. Either MS or CP module is inserted after the third stage of the network. Table 2 summarizes the comparative results of the MS module and the CP module on Something-Something V1 [3]. The CP module is effective for improving accuracy while consuming almost 6G FLOPs more than the baseline; the computational cost of the non-local correlation tensor is quadratic to the number of input frames. In contrast, the MS module performs 0.9% points and 0.8% points higher at top-1 and top-5 accuracy, respectively, while consuming 26% less FLOPs, compared to the CP module.

[1] https://github.com/xingyul/cpnet

Table 1: Performance comparison with different forms of feature transformation (FT) layers. $n \times (k, k)$ denotes n standard convolution layers with a kernel size of k. * denotes our FT layers in Fig. 3 of the paper.

model	FT layers	FLOPs	Top-1
TSM-R50	-	33.1G	46.7
MSNet-R50	$1 \times (1, 1)$	33.4G	49.3
MSNet-R50	$1 \times (3, 3)$	33.5G	49.8
MSNet-R50	$4 \times (3, 3)$	35.8G	50.4
MSNet-R50	ours*	33.7G	**50.9**

Table 2: Performance comparison between the CP module [9] and the MS module.

model	FLOPs	Top-1	Top-5
baseline	14.6G	41.5	71.8
CP module [9]	20.4G	44.9	75.6
MS module	15.0G	45.8	76.4

3 Backbone architectures in experiments

In our main paper, we evaluate the effect of the MS module on different backbone architectures: ResNet [4], TSM ResNet [8], MobileNet-V2 [10] and I3D [2]. We provide details of the backbone architectures here.

ResNet & TSM ResNet. Table 3 shows the architecture of ResNet [4] and TSM ResNet [8]. As a default, one MS module is inserted right after res_3.

I3D. Figure 1a, 1b show the architecture of I3D [2] used in our experiment; we reduce the first convolution kernel from 7×7×7 to 1×7×7 as we only use a sampled clip of 8 frames. The MS module is inserted after $Inc(b)$ of Figure 1b.

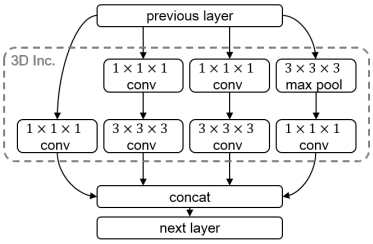
MobileNet-V2. Figure 2 and Table 4 show the architecture of MobileNet-V2 [10]. The MS module is inserted right after $stage_3$ of Table 4. As the feature channel size of the backbone is small enough, we omit the channel reduction layer in the MS module.

4 Additional examples of visualization

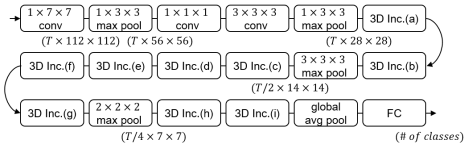
We present more results of visualization on Something-Something V1 [3] in Figure 3 and Kinetics-400 [7] in Figure 4. From the top of each figure, RGB frames, color-coded displacement maps [1], and confidence maps are illustrated. We visualize examples of horizontal, vertical movements (Figure 3a, 3b, 3c, 4a, 4b, 4c), rotations (Figure 3d, 4d), scale changes (Figure 3e, 4e), and deformations (Figure 3f, 4f). We also report some failure cases in the last row of figures (Figure 3g, 3h, 4g, 4h); estimated displacement maps around regions of occlusion or severe deformation are often inaccurate.

Table 3: ResNet & TSM ResNet backbone.

Layers	ResNet-18	TSM ResNet-18	ResNet-50	TSM ResNet-50	Output size
conv_1	1×7×7, 64, stride 1,2,2				T×112×112
	1×3×3 max pool, stride 2				
res_2	$\begin{bmatrix}1\times3\times3,\ 64\\1\times3\times3,\ 64\end{bmatrix}\times2$	$\begin{bmatrix}\text{TSM}\\1\times3\times3,\ 64\\1\times3\times3,\ 64\end{bmatrix}\times2$	$\begin{bmatrix}1\times1\times1,\ 256\\1\times3\times3,\ 256\\1\times1\times1,\ 256\end{bmatrix}\times3$	$\begin{bmatrix}\text{TSM}\\1\times1\times1,\ 256\\1\times3\times3,\ 256\\1\times1\times1,\ 256\end{bmatrix}\times3$	T×56×56
res_3	$\begin{bmatrix}1\times3\times3,\ 128\\1\times3\times3,\ 128\end{bmatrix}\times2$	$\begin{bmatrix}\text{TSM}\\1\times3\times3,\ 128\\1\times3\times3,\ 128\end{bmatrix}\times2$	$\begin{bmatrix}1\times1\times1,\ 512\\1\times3\times3,\ 512\\1\times1\times1,\ 512\end{bmatrix}\times4$	$\begin{bmatrix}\text{TSM}\\1\times1\times1,\ 512\\1\times3\times3,\ 512\\1\times1\times1,\ 512\end{bmatrix}\times4$	T×28×28
res_4	$\begin{bmatrix}1\times3\times3,\ 256\\1\times3\times3,\ 256\end{bmatrix}\times2$	$\begin{bmatrix}\text{TSM}\\1\times3\times3,\ 256\\1\times3\times3,\ 256\end{bmatrix}\times2$	$\begin{bmatrix}1\times1\times1,\ 1024\\1\times3\times3,\ 1024\\1\times1\times1,\ 1024\end{bmatrix}\times6$	$\begin{bmatrix}\text{TSM}\\1\times1\times1,\ 1024\\1\times3\times3,\ 1024\\1\times1\times1,\ 1024\end{bmatrix}\times6$	T×14×14
res_5	$\begin{bmatrix}1\times3\times3,\ 512\\1\times3\times3,\ 512\end{bmatrix}\times2$	$\begin{bmatrix}\text{TSM}\\1\times3\times3,\ 512\\1\times3\times3,\ 512\end{bmatrix}\times2$	$\begin{bmatrix}1\times1\times1,\ 2048\\1\times3\times3,\ 2048\\1\times1\times1,\ 2048\end{bmatrix}\times3$	$\begin{bmatrix}\text{TSM}\\1\times1\times1,\ 2048\\1\times3\times3,\ 2048\\1\times1\times1,\ 2048\end{bmatrix}\times3$	T×7×7
global average pool, FC					# of classes



(a) A 3D Inception module of I3D [2]



(b) I3D [2] architecture.

Fig. 1: I3D (BN-Inception [6]) backbone.

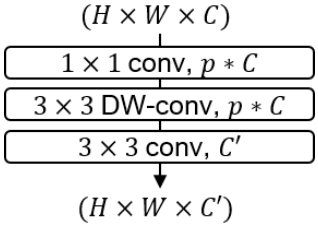
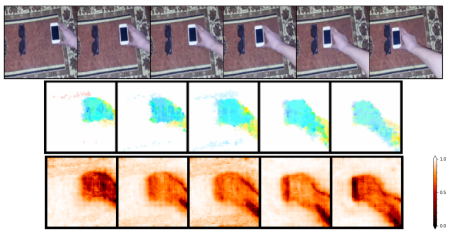


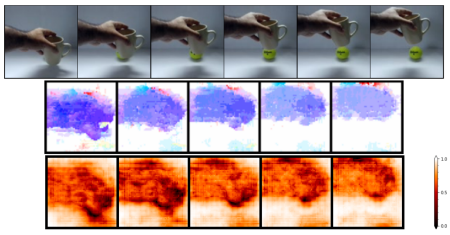
Fig. 2: A bottleneck(p, C') module of MobileNet-V2 [10]. The module transforms C channels to C' channels with an expansion factor p. DW-conv denotes a depth-wise convolution [5].

Table 4: MobileNet-V2 backbone. Bottleneck modules in Figure 2 are applied to the backbone.

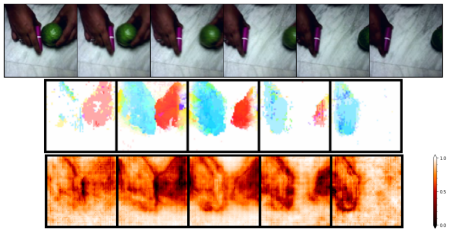
Layers	MobileNet-V2	Output size
$stage_1$	1×7×7, 32, stride 1,2,2	T×112×112
$stage_2$	bottleneck(1,16)	T×56×56
	bottleneck(6,24) × 2	
$stage_3$	bottleneck(6,32) × 3	T×28×28
$stage_4$	bottleneck(6,64) × 4	T×14×14
	bottleneck(6,96) × 3	
$stage_5$	bottleneck(6,160) × 3	T×7×7
	bottleneck(6,320)	
	1×1×1, 1280, stride 1,1,1	
global average pool, FC		# of classes



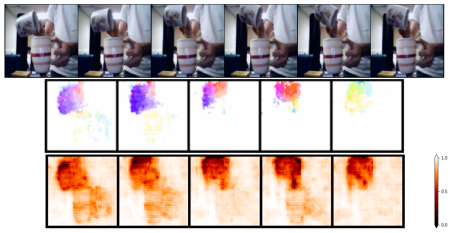
(a) “Moving sth. closer to sth.”.



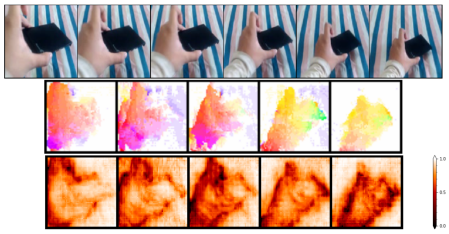
(b) “Removing sth., revealing sth. behind”.



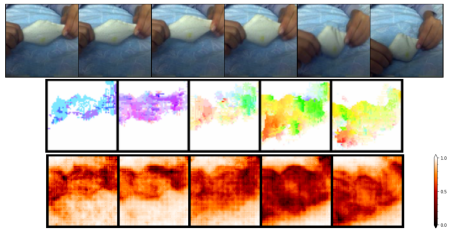
(c) “Moving sth. and sth. away from each other”.



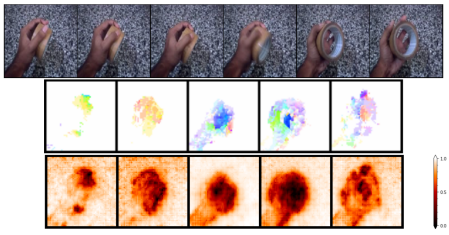
(d) “Pouring sth. into sth.”.



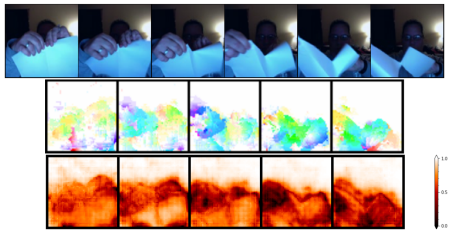
(e) “Pretending to put sth. on a surface”.



(f) “Pulling two ends of sth. so that it gets stretched”.

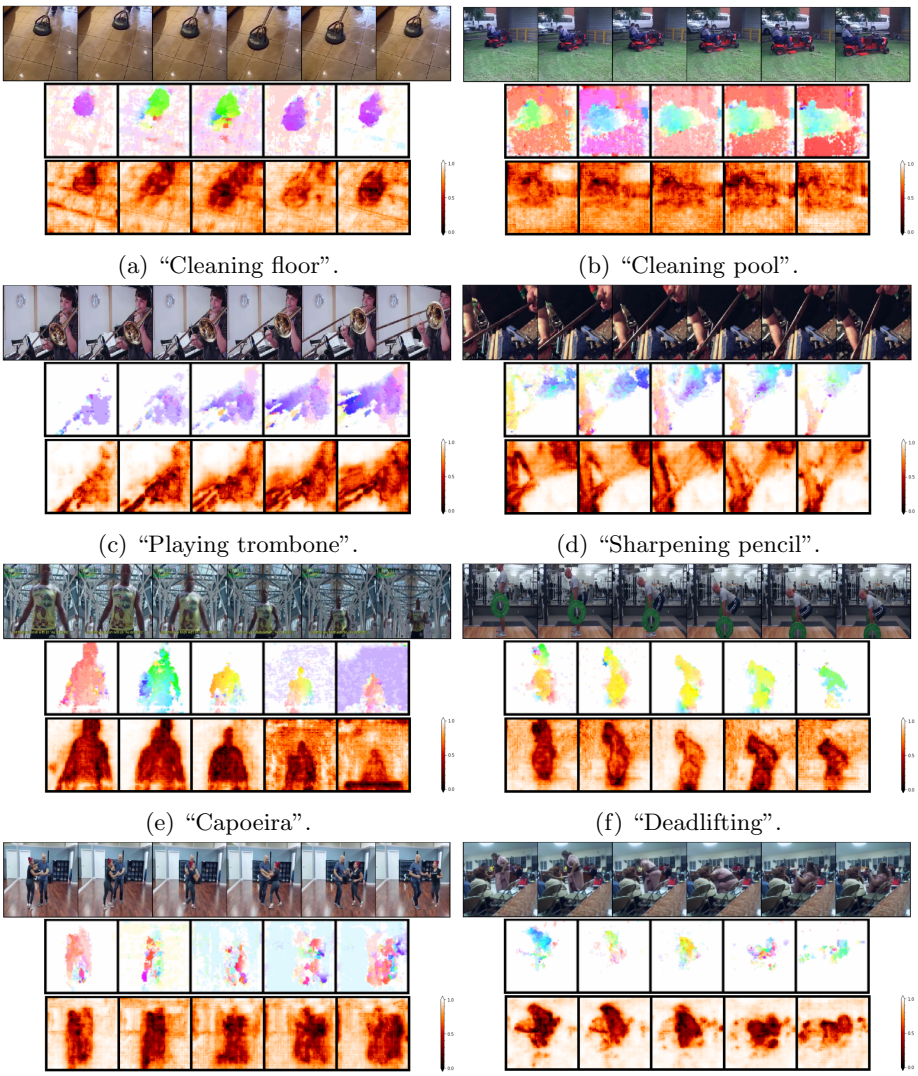


(g) “Pretending to squeeze sth.”.



(h) “Tearing sth. into two pieces”.

Fig. 3: Visualization on Something-Something V1 [3] dataset. Video frames, displacement maps, and confidence maps are shown from the top row in each subfigure.



(a) “Cleaning floor”. (b) “Cleaning pool”.

(c) “Playing trombone”. (d) “Sharpening pencil”.

(e) “Capoeira”. (f) “Deadlifting”.

(g) “Salsa dancing”. (h) “Washing feet”.

Fig. 4: Visualization on Kinetics-400 [7] dataset. Video frames, displacement maps, and confidence maps are shown from the top row in each subfigure.

References

1. Simon Baker, Daniel Scharstein, JP Lewis, Stefan Roth, Michael J Black, and Richard Szeliski. A database and evaluation methodology for optical flow. *International Journal of Computer Vision (IJCV)*, 92(1):1–31, 2011. 2
2. Joao Carreira and Andrew Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 2, 3
3. Raghav Goyal, Samira Ebrahimi Kahou, Vincent Michalski, Joanna Materzynska, Susanne Westphal, Heuna Kim, Valentin Haenel, Ingo Fruend, Peter Yianilos, Moritz Mueller-Freitag, et al. The" something something" video database for learning and evaluating visual common sense. In *Proc. IEEE International Conference on Computer Vision (ICCV)*, 2017. 1, 2, 4
4. Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 2
5. Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017. 3
6. Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015. 3
7. Will Kay, Joao Carreira, Karen Simonyan, Brian Zhang, Chloe Hillier, Sudheendra Vijayanarasimhan, Fabio Viola, Tim Green, Trevor Back, Paul Natsev, et al. The kinetics human action video dataset. *arXiv preprint arXiv:1705.06950*, 2017. 2, 5
8. Ji Lin, Chuang Gan, and Song Han. Tsm: Temporal shift module for efficient video understanding. In *Proc. IEEE International Conference on Computer Vision (ICCV)*, 2019. 2
9. Xingyu Liu, Joon-Young Lee, and Hailin Jin. Learning video representations from correspondence proposals. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 1, 2
10. Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 2, 3
11. Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 1