Learning Self-Similarity in Space and Time as Generalized Motion for Video Action Recognition

Supplementary Material

Heeseung Kwon* Manjin Kim* Suha Kwak Minsu Cho

Pohang University of Science and Technology (POSTECH), South Korea

http://cvlab.postech.ac.kr/research/SELFY/

We present more experimental results that could not be included in the main manuscript due to the lack of space.

A. Implementation details

Architecture details. We use TSN-ResNet and TSM-ResNet as our backbone (see Table 1) and initialize them with ImageNet pre-trained weights. We insert a single SELFY block right after res_3 and use the convolution method as a default feature extraction method. We set the spatio-temporal matching region of SELFY block, (L, U, V), as $(5, 9, 9)$ or $(9, 9, 9)$ when using 8 or 16 input frames, respectively. We stack four $1 \times 3 \times 3$ convolution layers along (l, u, v) dimension for the feature extraction method, and use four 3×3 convolution layers along (x, y) dimension for the feature integration. We reduce a spatial resolution of video feature tensor, $\mathbf{V}$, as 14×14 for computation efficiency before the self-similarity transformation. After the feature integration, we upsample the integrated feature tensor, $\mathbf{G}^{\star}$, as 28×28 for the residual connection.

Training. We sample a clip of 8 or 16 frames from each video by using segment-based sampling [21]. We resize the sampled clips into 240×320 images and apply random scaling and horizontal flipping for data augmentation. When applying the horizontal flipping on SS-V1&V2 [5], we do not flip clips of which class labels include 'left' or 'right' words; the action labels, *e.g.*, 'pushing something from left to right.' We fit the augmented clips into a spatial resolution of 224×224. We adopt the SGD optimizer with a momentum of 0.9. For SS-V1&V2, we set the initial learning rate to 0.01 and the training epochs to 50; the learning rate is decayed by 1/10 after 30^{th} and 40^{th} epochs. The training time of SELFYNet-TSM-R50 using 16 frames on SS-V1&V2 is about 2∼3 days with 8 Titan RTX GPUs. For Diving-48 [11] and FineGym [18], we use a cosine learning rate schedule [15] with the first 10 epochs for gradual warm-up [4]. We set the initial learning rate to 0.01 and the training epochs to 30 and 40, respectively.

Layers	TSN ResNet-50	TSM ResNet-50	Output size
conv_1	1×7×7, 64, stride 1,2,2		T×112×112
pool_1	1×3×3 max pool, stride 1,2,2		T×56×56
res_2	$\begin{bmatrix} 1\times1\times1, 256 \\ 1\times3\times3, 256 \\ 1\times1\times1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} \text{TSM} \\ 1\times1\times1, 256 \\ 1\times3\times3, 256 \\ 1\times1\times1, 256 \end{bmatrix} \times 3$	T×56×56
res_3	$\begin{bmatrix} 1\times1\times1, 512 \\ 1\times3\times3, 512 \\ 1\times1\times1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} \text{TSM} \\ 1\times1\times1, 512 \\ 1\times3\times3, 512 \\ 1\times1\times1, 512 \end{bmatrix} \times 4$	T×28×28
res_4	$\begin{bmatrix} 1\times1\times1, 1024 \\ 1\times3\times3, 1024 \\ 1\times1\times1, 1024 \end{bmatrix} \times 6$	$\begin{bmatrix} \text{TSM} \\ 1\times1\times1, 1024 \\ 1\times3\times3, 1024 \\ 1\times1\times1, 1024 \end{bmatrix} \times 6$	T×14×14
res_5	$\begin{bmatrix} 1\times1\times1, 2048 \\ 1\times3\times3, 2048 \\ 1\times1\times1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} \text{TSM} \\ 1\times1\times1, 2048 \\ 1\times3\times3, 2048 \\ 1\times1\times1, 2048 \end{bmatrix} \times 3$	T×7×7
global average pool, FC			# of classes

Table 1: **TSN & TSM ResNet-50 backbone.**

Testing. Given a video, we sample 1 or 2 clips, resize them into 240×320 images, and crop their centers as 224×224. We evaluate an average prediction of the sampled clips. We report top-1 and top-5 accuracy for SS-V1&V2 and Diving-48, and mean-class accuracy for FineGym.

Frame corruption details. We adopt two corruptions, occlusion and motion blur, to test the robustness of SELFYNet. We only corrupt a single center-frame for every validation clip of SS-V1; we corrupt the 4^{th} frame amongst 8 input frames. For the occlusion, we cut out a rectangle region from the center of the frame. For the motion blur, we adopt ImageNet-C implementation, which is available online[1]. We set 6 levels of severity for each corruption. We set the side length of the occluded region as 40px, 80px, 120px, 160px, 200px and 224px from the level 1 to 6. For the motion blur, we set (*radius*, *sigma*) tuple arguments as $(15, 5)$, $(10, 8)$, $(15, 12)$, $(20, 15)$, $(25, 20)$, and $(30, 25)$.

*Equal contribution.

[1] https://github.com/hendrycks/robustness

model	backbone	#frames	FLOPs×clips	top-1
STM [7]	Res-50	16	67 G×30	73.7
TSM [12]	Res-50	16	65 G×30	74.7
TEINet [14]	Res-50	16	66 G ×30	76.2
TEA [10]	Res-50	16	70 G×30	76.1
MSNet-TSM [9]	Res-50	16	67 G×10	76.4
SlowFast 16 × 8+NL [3]	3D Res-101	16+128	234 G×30	79.8
TimeSformer-L [1]	ViT-L [2]	96	2380 G×3	80.7
SELFYNet-TSM (ours)	Res-50	16	77 G×30	**77.1**

Table 2: **Performance comparison on Kinetics-400 [8].**

B. Performance comparison on Kinetics-400

We also conduct experiments on Kinetics-400 [8], which is the most popular appearance-centric benchmark. Table 2 summarizes the results on Kinetics-400. The first and second compartment of the table shows the results of different models with Res-50 using 16 frames and the results of the state-of-the-art models, respectively. The last row shows our result. The results demonstrate that SELFYNet still shows a clear improvement on the appearance-centric benchmark. SELFYNet obtains the improvement of 2.4%p at top-1 accuracy over the TSM baseline [12], achieving the best accuracy among the models with Res-50 using 16 frames. Although the accuracy of SELFYNet is inferior to that of SlowFast [3] or TimeSformer-L [1], we expect that SELFYNet can achieve the state-of-the-art when using larger backbones (3D Res-101, ViT-L) or a bigger input.

In the following, we provide implementation details for Kinetics-400 experiments. We adopt the dense frame sampling method [22] and sample a clip of 16 frames. For training, we use a cosine learning rate schedule with the first 10 epochs for warm-up. We set the initial learning rate to 0.01 and total epochs to 65. For testing, we sample 10 uniform clips per video and average the softmax scores for the final prediction. We follow the strategy of non-local networks [22] to pre-process the frames and take 3 crops as input. Other experimental details are the same as those in the supplementary material A.

C. Additional experiments

We conduct additional experiments to identify the behaviors of the proposed method. All experiments are performed on SS-V1 by using 8 frames. Unless otherwise specified, we set ImageNet pre-trained TSM ResNet-18 (TSM-R18) with a single SELFY block of which $(L, U, V) = (5, 9, 9)$, as our default SELFYNet.

Spatial matching region. In Table 3a, we compare a single SELFY block with different spatial matching regions, (U, V). As a result, indeed, the larger spatial matching region leads the better accuracy. Considering the accuracy-computation trade-off, we set our spatial matching region, (U, V), as $(9, 9)$ as a default.

Block position. From the 2nd to the 6th row of Table 3b, we identify the effect of different positions of SELFY block in the backbone. We resize the spatial resolution of the video tensor, (X, Y), into 14×14, and fix the matching region, (L, U, V), as $(5, 9, 9)$ for all the cases maintaining the similar computational cost. SELFY after the res_3 shows the best trade-off by achieving the highest accuracy among the cases; early-stage features (pool_1,res_2) lack enough semantics for robust matching while late-stage ones (res_4,res_5) lose appearance details for accurate matching. The last row in Table 3b shows that the multiple SELFY blocks improve accuracy compared to the single block.

Fusing STSS features with visual features. We evaluate SELFYNet purely based on STSS features to see how much the ordinary visual feature $\mathbf{V}$ contributes to the final prediction. That is, we pass the STSS features, $\mathbf{Z} = \text{ReLU}(\mathbf{F}^\star \times_5 \mathbf{W}_\theta)$, into the downstream layers without additive fusion. Table 3c compares the results of using different cases of the output tensor ($\mathbf{V}$, $\mathbf{Z}$, and $\mathbf{Z}+\mathbf{V}$) on SS-V1. Interestingly, SELFYNet using only $\mathbf{Z}$ achieves 45.5% at top-1 accuracy, which is higher as 2.5%p than the baseline. As we add $\mathbf{V}$ to $\mathbf{Z}$, we obtain an additional gain of 2.9%p. It indicates that the STSS features and the visual features are complementary to each other.

Multi-channel $3\times3\times3$ kernel for feature extraction. We investigate the effect of the convolution method for STSS feature extraction when we use multi-channel $3 \times 3 \times 3$ kernels. For the experiment, we stack four $3 \times 3 \times 3$ convolution layers followed by the feature integration step, which are the same as in Section 3.2.2 in our main manuscript. Table 3d summarizes the results. Note that we do not report models of which temporal window $L = 1$, *e.g.*, $\{0\}$ and $\{1\}$. As shown in the table, indeed, the long temporal range gives a higher accuracy. However, the effect of the $3 \times 3 \times 3$ kernel is comparable to that of the $1 \times 3 \times 3$ kernel in Table 4a in our main manuscript. Considering the accuracy-computation trade-off, we choose to fix the kernel size, $L_\kappa \times U_\kappa \times V_\kappa$, as $1 \times 3 \times 3$ for the STSS feature extraction.

Relation with local self-attention mechanisms. The local self-attention [6, 16, 23] and our method have a common denominator of using the self-similarity tensor but use it in a very different way and purpose. The local self-attention mechanism aims to aggregate the local context features using the self-similarity tensor, and it thus uses the self-similarity values as attention weights for feature aggregation. However, our method aims to learn a generalized motion representation from the local STSS, so the final STSS representation is directly fed into the neural network instead of multiplying it to local context features.

For an empirical comparison, we conduct an ablation experiment as follows. We extend the local self atten-

model	$U \times V$	FLOPs	top-1	top-5
TSM-R18	-	14.6 G	43.0	72.3
SELFYNet	5×5	17.1 G	47.8	77.1
	9×9	17.3 G	48.4	77.6
	13×13	18.4 G	48.4	77.8
	17×17	19.8 G	**48.6**	**78.3**

(a) **Spatial matching region**. Performance comparison with different spatial matching-regions, ($U \times V$).

model	position	top-1	top-5
TSM-R18	-	43.0	72.3
SELFYNet	pool_1	45.7	77.6
	res_2	47.2	76.6
	res_3	48.4	77.6
	res_4	46.6	76.0
	res_5	42.8	72.6
	$\text{res}_{2,3,4}$	**48.6**	**77.9**

(b) **Position**. Performance comparison with different positions of SELFY block. For the last row, 3 SELFY blocks are used in total.

model	features	top-1	top-5
TSM-R18	$\mathbf{V}$	43.0	72.3
SELFYNet	$\mathbf{Z}$	45.5	75.9
	$\mathbf{Z} + \mathbf{V}$	**48.4**	**77.6**

(c) **STSS features with visual features.** $\mathbf{V}, \mathbf{Z}$ denotes the visual features and STSS features, respectively.

model	range of l	top-1	top-5
TSM-R18	-	43.0	72.3
SELFYNet	$\{-1, 0, 1\}$	47.4	77.0
	$\{-2, \cdots, 2\}$	48.3	77.2
	$\{-3, \cdots, 3\}$	**48.5**	**77.4**

(d) **Multi-channel** $3 \times 3 \times 3$ **kernel for feature extraction.** Four convolution layers are used for extracting STSS features. $\{\cdot\}$ denotes a set of temporal offsets l.

model	similarity	extraction	top-1	top-5
TSM-R18	-	-	43.0	72.3
SELFYNet	embed. G	mult. w/ $\mathbf{V}$	43.8	72.3
	embed. G	Conv	47.6	76.8
	cosine	Conv	47.8	77.1

(e) **Performance comparison with the local self-attention mechanisms [6, 16].** We implemented the local self-attention by following Ramachandran *et al.* [16].

model	extraction	(L, U, V)	top-1	top-5
TSM-R18	-	-	43.0	72.3
SELFYNet	KS + CM	$(1, 9, 9)$	46.1	75.3
	KS + CM	$(5, 9, 9)$	47.4	76.8
	Conv	$(1, 9, 9)$	47.1	76.3
	Conv	$(5, 9, 9)$	**48.4**	**77.6**

(f) **Performance comparison with MSNet [9].** KS and CM denote the kernel soft-argmax and confidence map, respectively.

model	frames	FLOPs	memory	runtime	top-1	top-5
TSM-R50 [12]	8	33.1 G	8.2 GB	15.6 ms	45.6	74.2
TSM-R50 [12]	16	66.3 G	15.7 GB	30.1 ms	47.3	77.1
TSM-R50 + NL [22]	8	46.5 G	10.3 GB	24.0 ms	49.1	77.2
TSM-R50 + MHSA [19]	8	50.6 G	15.9 GB	26.3 ms	49.2	77.9
TSM-R50 + SELFY	8	36.6 G	9.6 GB	21.1 ms	**52.5**	**80.8**

(g) **Efficiency**. Performance comparison with other attention mechanisms [19, 22]. We insert a single block after res_3 in TSM-R50. We use 8 clips per GPU and measure the runtime by following protocols in [9].

Table 3: **Additional experiments on SS-V1**. Top-1 & 5 accuracy (%) are shown.

tion layer [16] to the temporal dimension and then add the *spatio-temporal* local self-attention layer, which is followed by feature integration layers, after res_3. All experimental details are the same as those in supplementary material A, except that we reduce the channel dimension C of appearance feature $\mathbf{V}$ to 32. Table 3e summarizes the results on SS-V1. The spatio-temporal local self-attention layer is accurate as 43.8% at top-1 accuracy, and both of SELFY blocks using the embedded Gaussian and the cosine similarity outperform the local self-attention by achieving top-1 accuracy as 47.6% and 47.8%, respectively. These results are in alignment with the prior work [13], which reveals that the self-attention mechanism hardly captures motion in the video.

Comparison with correlation-based methods. We also investigate the difference between our method and correlation-based methods [9, 20]. While correlation-based methods extract motion features only from the spatial cross-

similarity tensor between two adjacent frames, and are thus limited to short-term motion, our method effectively captures bi-directional and long-term motion information via learning with the sufficient volume of STSS. Our method can also exploit richer information from the self-similarity values than other methods. MS module [9] only focuses on the maximal similarity value of the (u, v) dimensions to extract flow information, and Correlation block [20] uses an 1×1 convolution layer for extracting motion features from the similarity values. In contrast to the two methods, we introduce a generalized motion learning framework using the self-similarity tensor in Section 3.2 in our main manuscript.

We also conduct experiments to compare our method with MSNet [9], one of the correlation-based methods. For an apple-to-apple comparison, we apply kernel soft-argmax and max pooling operation (*KS + CM* in [9]) to our feature extraction method by following their official codes[2]. Please note that, when we restrict the temporal offset l to $\{1\}$, the SELFY block using KS + CM is equivalent to the MS module of which *feature transformation* layers are the standard 2D convolution layers. Table 3f summarizes the results. KS+CM method achieves 46.1% at top-1 accuracy. As we enlarge the temporal window L to 5, we obtain the additional gain as 1.3%p. The learnable convolution layers improve the top-1 accuracy by 1.0%p in both cases. The results demonstrate the effectiveness of learning geometric patterns within the sufficient volume of STSS tensors for learning motion features.

Efficiency. In Table 3g, we compare the efficiency of SELFYNet with that of other self-attention methods [19,22] in terms of FLOPs, memory footprint, runtime, and accuracy. Compared to TSM-R50 using 16 frames, SELFYNet using 8 frames consumes less memory by 6.1 GB and runs faster by 9.0 ms while improving top-1 accuracy by 5.2 %p. Compared to the self-attention methods [19,22], SELFYNet also achieves the best accuracy with less memory footprint and faster inference speed.

D. Visualizations

In Fig. 1, we visualize some qualitative results of two different SELFYNet-TSM-R18 ($\{1\}$ and $\{-3, \cdots, 3\}$) on SS-V1. We show the different predictions of the two models with 8 input frames. We also overlay Grad-CAMs [17] on the input frames to see whether a larger volume of STSS benefits to capture long-term interactions in videos. We take Grad-CAMs of features which is right before a global average pooling layer. As shown in the figure, the STSS with the sufficient volume helps to learn the more enriched context of temporal dynamics in the video; in Fig. 1a, for example, SELFYNet with the range of ($\{-3, \cdots, 3\}$) focuses on not only regions on which an action occurs but also focuses on the white-stain after the action to verify whether the stain is wiped off or not.

[2] https://github.com/arunos728/MotionSqueeze

References

[1] Bertasius et al. Is space-time attention all you need for video understanding? *arXiv preprint arXiv:2102.05095*, 2021. 2

[2] Dosovitskiy et al. An image is worth 16x16 words: Transformers for image recognition at scale. *Proc. International Conference on Learning Representations (ICLR)*, 2021. 2

[3] Christoph Feichtenhofer, Haoqi Fan, Jitendra Malik, and Kaiming He. Slowfast networks for video recognition. In *Proc. IEEE International Conference on Computer Vision (ICCV)*, 2019. 2

[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 preprint arXiv:1706.02677*, 2017. 1

[5] 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

[6] Han Hu, Zheng Zhang, Zhenda Xie, and Stephen Lin. Local relation networks for image recognition. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3464–3473, 2019. 2, 3

[7] Boyuan Jiang, Mengmeng Wang, Weihao Gan, Wei Wu, and Junjie Yan. Stm: Spatiotemporal and motion encoding for action recognition. In *Proc. IEEE International Conference on Computer Vision (ICCV)*, 2019. 2

[8] 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

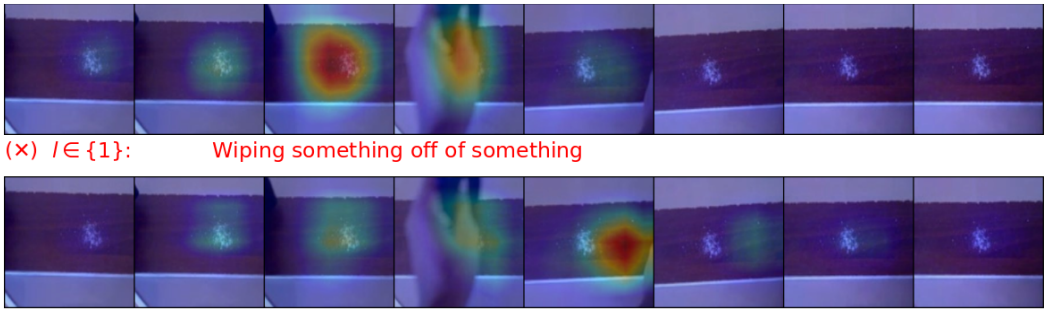
[9] Heeseung Kwon, Manjin Kim, Suha Kwak, and Minsu Cho. Motionsqueeze: Neural motion feature learning for video understanding. *arXiv preprint arXiv:2007.09933*, 2020. 2, 3, 4

[10] Yan Li, Bin Ji, Xintian Shi, Jianguo Zhang, Bin Kang, and Limin Wang. Tea: Temporal excitation and aggregation for action recognition. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 2

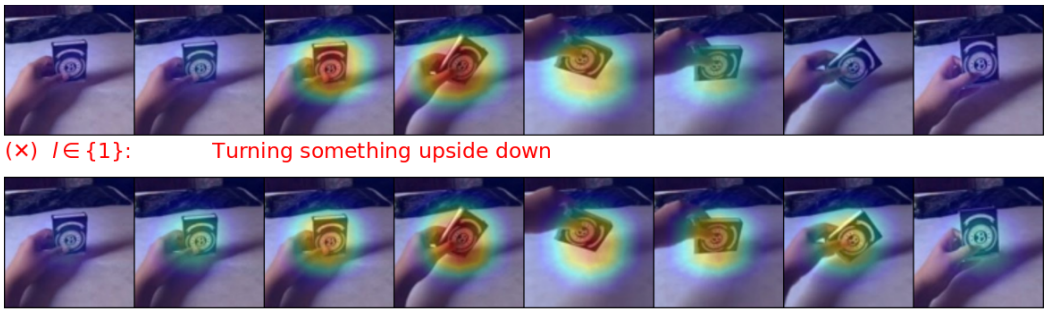
[11] Yingwei Li, Yi Li, and Nuno Vasconcelos. Resound: Towards action recognition without representation bias. In *Proc. European Conference on Computer Vision (ECCV)*, 2018. 1

[12] 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, 3

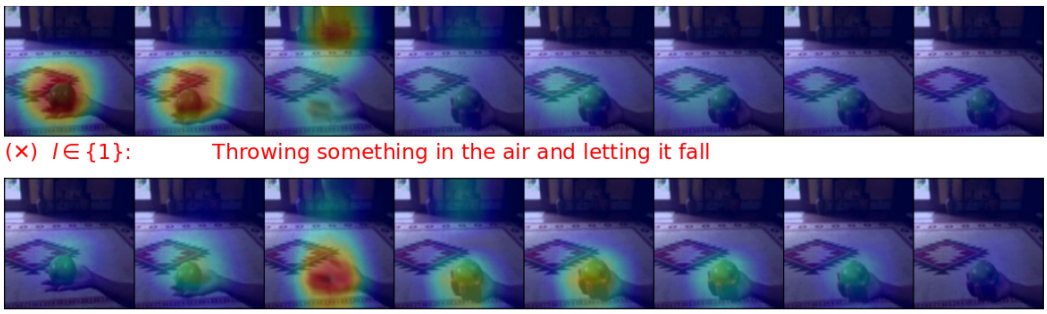
[13] Xingyu Liu, Joon-Young Lee, and Hailin Jin. Learning video representations from correspondence proposals. In *Proc.*



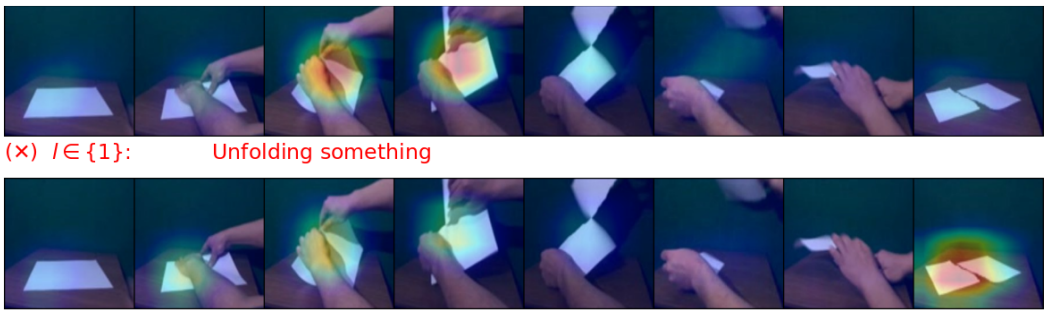
(a)



(b)



(c)



(d)

Figure 1: **Qualitative results** of two SELFYNets on SS-V1. Each subfigure visualizes prediction results of the two models with Grad-CAM-overlaid RGB frames. The correct and wrong predictions are colorized as green and red, respectively.

IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2019. 3

[14] Zhaoyang Liu, Donghao Luo, Yabiao Wang, Limin Wang, Ying Tai, Chengjie Wang, Jilin Li, Feiyue Huang, and Tong Lu. Teinet: Towards an efficient architecture for video recognition. In *Proc. AAAI Conference on Artificial Intelligence (AAAI)*, 2020. 2

[15] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016. 1

[16] Prajit Ramachandran, Niki Parmar, Ashish Vaswani, Irwan Bello, Anselm Levskaya, and Jonathon Shlens. Stand-alone self-attention in vision models. *arXiv preprint arXiv:1906.05909*, 2019. 2, 3

[17] 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 *Proc. IEEE International Conference on Computer Vision (ICCV)*, 2017. 4

[18] Dian Shao, Yue Zhao, Bo Dai, and Dahua Lin. Finegym: A hierarchical video dataset for fine-grained action understanding. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 1

[19] Aravind Srinivas, Tsung-Yi Lin, Niki Parmar, Jonathon Shlens, Pieter Abbeel, and Ashish Vaswani. Bottleneck transformers for visual recognition. *arXiv preprint arXiv:2101.11605*, 2021. 3, 4

[20] Heng Wang, Du Tran, Lorenzo Torresani, and Matt Feiszli. Video modeling with correlation networks. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 3, 4

[21] Limin Wang, Yuanjun Xiong, Zhe Wang, Yu Qiao, Dahua Lin, Xiaoou Tang, and Luc Van Gool. Temporal segment networks: Towards good practices for deep action recognition. In *Proc. European Conference on Computer Vision (ECCV)*, 2016. 1

[22] 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. 2, 3, 4

[23] Hengshuang Zhao, Jiaya Jia, and Vladlen Koltun. Exploring self-attention for image recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10076–10085, 2020. 2