Supplementary Material for Regularization on Spatio-Temporally Smoothed Feature for Action Recognition

Stage	SlowOnly-34	SlowOnly-50	$T \times S^2$
clip	-		8×224^2
conv_1	$1 \times 7^2, 64$ stride $1,2^2$		8×112^2
pool_1	$1 \times 3^2, max$ stride $1,2^2$		8×56^2
res_2	$\begin{bmatrix} 1 \times 3^2, 64 \\ 1 \times 3^2, 64 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1^2, 64 \\ 1 \times 3^2, 64 \\ 1 \times 1^2, 256 \end{bmatrix} \times 3$	8×56^2
res_3	$\begin{bmatrix} 1 \times 3^2, 128 \\ 1 \times 3^2, 128 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1^2, 128 \\ 1 \times 3^2, 128 \\ 1 \times 1^2, 512 \end{bmatrix} \times 4$	8×28^2
res_4	$\begin{bmatrix} 3 \times 3^2, 256 \\ 3 \times 3^2, 256 \end{bmatrix} \times 6$	$\begin{bmatrix} 3 \times 1^2, 256 \\ 1 \times 3^2, 256 \\ 1 \times 1^2, 1024 \end{bmatrix} \times 6$	8×14^2
res_5	$\begin{bmatrix} 3 \times 3^2, 512 \\ 3 \times 3^2, 512 \end{bmatrix} \times 3$	$\begin{bmatrix} 3 \times 1^2, 512 \\ 1 \times 3^2, 512 \\ 1 \times 1^2, 2048 \end{bmatrix} \times 3$	8×7^2
global average pool, fc			-

Table A1. An instantiation of SlowOnly model.

A. Architecture of SlowOnly Model

Architectural details of SlowOnly are specified in Table A1. One can find kernel size, number of residual blocks, number of channels, strides, and feature map size in each stage in the table.

B. Additional Details of Training

B.1. Kinetics-400 and Mini-Kinetics Stochastic gradient descent (SGD) with momentum is used to optimize the model. Cosine annealing [2] with initial learning rate 0.2 is used for the learning rate scheduler. We also apply warm-up scheme that starts from 0.1 during the first 20 epochs. Momentum and weight decay are set to 0.9 and 0.0001 respectively. Dropout is applied before the last fully connected layer with a drop rate 0.5. The model is trained for 250 epochs with 128 batch size using 4 GPUs otherwise specified.

For Mini-Kinetics, we train SlowOnly-50 with batch size 64 due to the limitation of computational resources.

For Kinetics-400, we adopt distributed and mixed precision training in order to train SlowOnly-50 model with large batch size, 128.

Since official code[1] of SlowFast Network (SlowFast) was not released by the time we experimented, we implemented it in PyTorch with reference to unofficial implementation[2]. There were several differences between official code and our implementation which can affect the convergence of the model; 1) BN and ReLU at the lateral connection, 2) initialization of gamma in BN to 0, 3) weight decay of BN parameters to 0.0, and 4) small initial learning rate of warm-up scheme. We use SlowFast model with $T = 8$, $\tau = 8$ and $\alpha = 4$. We set batch size 64 with 4 GPUs and learning rate 0.1. RMS is only applied in Slow branch.

For CSN, we implemented it with reference to an Caffe2 implementation[3]. Since CSN gradually aggregates temporal dimension through the stages, the temporal dimension of the feature map at the latter stages can be less than 3. Therefore, we use RMS with [1,3,3] kernel in CSN.

B.2. Something-Something-v2 For Something-v2, we finetune SlowFast for 40 epochs with initial learning rate 0.01. We use step decay schedule which reduces learning rate by 1/10 at 24 and 30 epochs. According to the training graph, we found that the model easily overfit to the dataset so we enhance the input augmentation, random resized crop, by cropping image patches from the range between 15% to 75% of the image area.

B.3. CIFAR100 and ImageNet For CIFAR-100, ResNet-110 is chosen as the baseline model and trained for 300 epochs. We decay learning rate by 1/10 at 150 and 225 epochs. The initial learning rate set to 0.2 with batch size 1024. We run 3 times with different initialization and report average of the performance. RMS is applied in the last stage among three residual stages.

For ImageNet experiment, we train ResNet-50 for 300 epochs following [4]. We also found that large number of epochs requires for RMS to converge. We adopted cosine

[1] https://github.com/facebookresearch/SlowFast
[2] https://github.com/r1ch88/SlowFastNetworks
[3] https://github.com/facebookresearch/VMZ

annealing learning rate schedule with 5 warm-up epochs. The initial learning rate set to 0.4 with batch size 1024. RMS is applied in res_4 and res_5 stages as 3D models.

C. Details of Gaussian Filter

The Gaussian kernel can be formulated as

$$w_j = \frac{1}{(\sigma_f\sqrt{2\pi})^3} e^{-\frac{x_j^2+y_j^2+t_j^2}{2\sigma_f^2}} \tag{1}$$

where x_j, y_j, and t_j are discrepancy from the center of the kernel along x, y, and t dimension and σ_f is a hyperparamter that determines the standard deviation of the Gaussian kernel.

D. Details of Other Regularizations

Like RMS, RandomDrop [1] and ShakeDrop [3] multiply a random coefficient on feature during a forward computation. Apart from this similarity, RMS has several distinctive characteristics from RadomDrop and ShakeDrop. RMS varies a feature in a residual branch with a continuous random variable, whereas RandomDrop stochastically drops l^{th} residual branch by the discrete random variable b_l from a Bernoulli distribution with survival probability $P(b_l = 1) = p_l$ as represented in the following equation.

$$\mathbf{y} = \begin{cases} b_l\mathbf{x}, & \text{in train} \\ E[b_l]\mathbf{x}, & \text{in test.} \end{cases} \tag{2}$$

On the other hand, ShakeDrop uses two continuous random variables to perturb a residual branch. When $b_l = 0$, ShakeDrop multiplies α during a forward phase and β during a backward phase, respectively:

$$\mathbf{y} = \begin{cases} (b_l + \alpha - b_l\alpha)\mathbf{x}, & \text{in train forward} \\ (b_l + \beta - b_l\beta)\mathbf{x}, & \text{in train backward} \\ E[b_l + \alpha - b_l\alpha]\mathbf{x}, & \text{in test.} \end{cases} \tag{3}$$

In [3], ShakeDrop shows the best performance when $\alpha = 0$ and $\beta \sim U(0, 1)$ sampled for each mini-batch (batch-level) or for each pixel (pixel-level) for ResNet. On the other hand, RMS samples a random variable, α for each video clip (clip-level).

Cutmix [4] is a recently proposed input augmentation method which mixes two randomly chosen images x_1 and x_2 with their corresponding labels y_1 and y_2 as

$$\begin{aligned} \tilde{x} &= \mathbf{M} \odot x_1 + (\mathbf{1} - \mathbf{M}) \odot x_2, \\ \tilde{y} &= \lambda y_1 + (1 - \lambda) y_2, \end{aligned} \tag{4}$$

where binary mask $\mathbf{M} \in \{0, 1\}^{W \times H}$ decides where to cut and paste the image patch and $\odot$ denotes element-wise multiplication. The size of the image patch is determined by the combination ratio λ which is sampled from the beta distribution $Beta(1, 1)$. We recommend readers to refer to the corresponding paper for further details.

For our experiment, We trained all methods with their best hyper-parameter settings reported in each corresponding paper. However, we could not make ShakeDrop converge with their best reported hyper-parameter on Mini-Kinetics dataset. So we decrease the regularization effect by reducing the range of β by half. For Cutmix, we simply extend the method to 3D by adopting identical crop region to every time steps.

References

[1] Gao Huang, Yu Sun, Zhuang Liu, Daniel Sedra, and Kilian Q. Weinberger. Deep networks with stochastic depth. In *The European Conference on Computer Vision*, pages 646–661, 2016. 2

[2] I. Loshchilov and F. Hutter. Sgdr: Stochastic gradient descent with warm restarts. In *International Conference on Learning Representations (ICLR)*, 2017. 1

[3] Yoshihiro Yamada, Masakazu Iwamura, and Koichi Kise. Shakedrop regularization. *arXiv:1802.02375*, 2018. 2

[4] Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. Cutmix: Regularization strategy to train strong classifiers with localizable features. In *The IEEE International Conference on Computer Vision (ICCV)*, October 2019. 1, 2