Regularization on Spatio-Temporally Smoothed Feature for Action Recognition

Jinhyung Kim[1*,†] Seunghwan Cha[2†] Dongyoon Wee[3] Soonmin Bae[3] Junmo Kim[1]
[1]KAIST [2]Carnegie Mellon University [3]Clova AI, NAVER Corp.

Abstract

Deep neural networks for video action recognition frequently require 3D convolutional filters and often encounter overfitting due to a larger number of parameters. In this paper, we propose Random Mean Scaling (RMS), a simple and effective regularization method, to relieve the overfitting problem in 3D residual networks. The key idea of RMS is to randomly vary the magnitude of low-frequency components of the feature to regularize the model. The low-frequency component can be derived by a spatio-temporal mean on the local patch of a feature. We present that selective regularization on this locally smoothed feature makes a model handle the low-frequency and high-frequency component distinctively, resulting in performance improvement. RMS can enhance a model with little additional computation only during training, similar to other regularization methods. RMS also can be incorporated into typical training process without any bells and whistles. Experimental results show the improvement in generalization performance on a popular action recognition datasets demonstrating the effectiveness of RMS as a regularization technique, compared to other state-of-the-art regularization methods.

1. Introduction

Overfitting is one of the long lasting and practical problems that deep neural networks have been confronted with. The problem can be more fatal in the field of video action recognition where 3D convolutional neural networks (3D ConvNet) [22, 7, 1] have become a popular approach for encoding spatio-temporal representation since they often have immense number of parameters. Moreover, 3D ConvNets often suffer from overfitting even on recent large-scale datasets, *e.g.* Kinetics [12]. There have been several approaches to address this problem by proposing more efficient network architectures [24, 26, 17, 4, 2].

Perturbation based regularization on the input space [30, 3, 29] and on the feature space [10, 5, 27] is another widely studied approach to alleviate the overfitting problem. One advantage of such methods is that they can be incorporated into existing models to enhance their generalization capability with little or no additional computation.

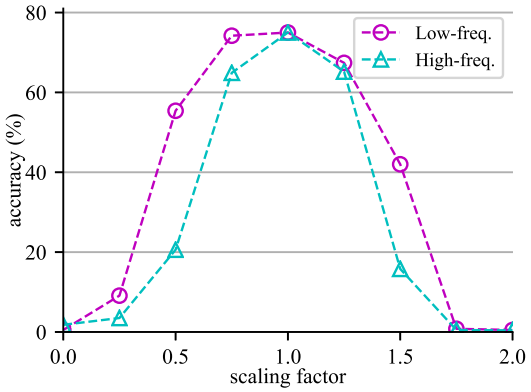

Figure 1: **High-frequency information of the feature is more important for action recognition.** Accuracy changes over the scaling factor which modulates the magnitude of the low-freq. (circle marker) or high-freq. (triangle marker) components in the feature. While one component is scaled, the other component remains unchanged. Mean filter extracts low-freq. components from the feature and the remainder is high-freq. components. The validation accuracy drops more quickly when high-freq. components are perturbed. Using 3D ResNet (SlowOnly-34) trained on Mini-Kinetics validation dataset without freq. scaling.

In particular, [10, 5, 27] have studied the effect of perturbation on feature extensively. A perturbation is given by multiplication of a random scalar to a feature. Although they have given a lot of intuition for regularization, they have overlooked the direction to perturb, where our question starts from.

In order to describe *direction to perturb* more clearly, let us consider a feature vector, $\mathbf{x}_{t,i,j} \in \mathbb{R}^C$, at each spatio-temporal position, t, i, and j, in a feature map, $\mathbf{x} \in \mathbb{R}^{C \times T \times H \times W}$, where C, T, H, and W are channel, time, height, and width respectively. We drop the position index of vectors in the following for the convenience. In this regard, direction to perturb means the direction of a vector to be multiplied by a random scalar, resulting in the

*Correspondence to kkjh0723@kaist.ac.kr
†This work is partly done during an internship at Clova AI.

modulation of the vector's magnitude while its direction is preserved. In order to examine the effect of frequency, we consider 3 options of direction to perturb in this study; direction of original feature vector $\mathbf{x}$, its local mean vector $\bar{\mathbf{x}}$, and residual vector $\mathbf{r}$. $\bar{\mathbf{x}}$ can be obtained by the mean filter (or Gaussian filter) in image processing [16]. $\bar{\mathbf{x}}$ contains low-frequency component of $\mathbf{x}$ so it varies more smoothly within its neighbor. The residual vector, $\mathbf{r} = \mathbf{x} - \bar{\mathbf{x}}$, contains remaining high-frequency component.

Under the assumption that frequency is meaningful criteria to separate features' direction to perturb, we analyze which frequency component is important for action recognition task. Fig. 1 shows the accuracy changes of 3D ConvNet over various scaling factors that modulates the magnitude of $\bar{\mathbf{x}}$ and $\mathbf{r}$. While one of them is scaled, the other remains unchanged. Note that the model trained without frequency scaling. This graph shows that modulation of the high-frequency component affects more than that of low-frequency component on action recognition performance. Our intuition is that $\mathbf{r}$ may possess more essential information for classification while $\bar{\mathbf{x}}$ may contain peripheral information so that small perturbation on it adds diversity of the sample without changing the type of action or object. Based on this observation, we speculate that the selective perturbation on low-frequency component of the feature can be an effective way to regularize a network.

Motivated by this idea, we propose a new regularization method, Random Mean Scaling (RMS), which selects low-frequency direction as direction to perturb. In other words, RMS adds perturbations selectively by multiplying a random scalar to the spatio-temporally smoothed feature $\bar{\mathbf{x}}$. To separate the low-frequency feature, we use a 3D mean filter (3D average pooling operation in most deep learning frameworks) or a Gaussian filter which are the simplest low-pass filters (LPF) in image processing [16]. Similar to other regularization methods, RMS is only required during training, so no additional operation is required during inference.

In order to demonstrate its effectiveness, we experiment RMS on various types of 3D Residual Network (ResNet) [8] for action recognition. As a baseline of this study, we adopt the architecture of the slow branch in SlowFast Network (SlowFast) [4], which is one type of 3D ResNet itself. We will call this model SlowOnly in the remaining sections following the original paper. Experimental results show that adding RMS to the baseline model improves the classification accuracy on Mini-Kinetics and Kinetics-400 dataset. We also compare RMS with several state-of-the-art regularization methods, RandomDrop [10], ShakeDrop [27] and CutMix [29], in Mini-Kinetics dataset and found that RMS shows competitive performance. For rigorous study, we explored the effect of various factors such as positions and sampling distributions of RMS. We also provide empirical evidences that applying perturbation to the low-frequency component is more effective than applying those to the entire feature or the high-frequency component. Additionally, we tested RMS with other models, SlowFast [4] and Channel Sperated Network (CSN) [23], and on another dataset, Something-Something-v2 [6], to show its generality.

In summary, we make the following contributions:

- We propose a simple yet effective regularization method, Random Mean Scaling (RMS), to tackle the overfitting problem of 3D ConvNet.
- We demonstrate the effectiveness of applying RMS to several kinds of 3D ResNets on Mini-Kinetics, Kinetics-400, and Something-Something-v2 dataset.
- We compare RMS with other state-of-the-art regularizations in Mini-Kinetics dataset.
- We validate the design choices of RMS in depth with extensive ablation study.

2. Related Work

2.1. Action Recognition in Videos

Compared to image recognition, action recognition in video requires additional mechanism to deal with temporal information along with spatial information. Accordingly, 3D ConvNets [22, 7, 1] have become a popular approach for action recognition by exploiting the advancement of 2D ConvNets in image recognition [20, 8]. 3D ConvNets can be expanded from 2D ConvNets with an additional dimension to handle spatio-temporal streams. Thanks to the simple expansion of the model design, 3D ConvNets can exploit the learned knowledge from image domain by inflating 2D kernels trained on the large-scale image recognition dataset as in [1]. Although 3D ConvNets show their effectiveness, a large number of parameters has been considered as their shortcoming. To overcome this problem, [24, 26, 17] proposed to decompose 3D kernels into a cascade of 2D and 1D kernels. CoST [15] decomposed a 3D filter into a 2D filter that can be simultaneously applied along H-W, T-H, and T-W. Additionally, models using 3D convolutional filters only in the later stages (top-heavy [26] or reversed mixed convolutions in [24]) often achieve a better performance with fewer number of parameters than full 3D ConvNets. Meanwhile, Channel Separated Convolutional Network (CSN) [23], a lightweight 3D ResNet model, reduced the number of parameters considerably by using depth-wise convolution.

On the other hand, some studies [4, 2] proposed the multi-scale models to use information separately according to its frequency. In [4], authors proposed a two-stream model, SlowFast, composed of the slow branch for static spatial feautres and the fast branch for dynamic motion features respectively. Meanwhile, [2] proposed Octave convo-

lution to process multi-frequency signals in a single-stream model.

Capturing global dependencies is another approach to improve a model for action recognition. [25] first proposed the concept of the non-local module and demonstrated the effectiveness of adding it into 3D ConvNets. Further studies, [28, 9] redefined the non-local module in various ways to reduce the computation with a better performance.

2.2. Regularization

Regularization has been actively studied in image domain to prevent a model from overfitting. For example, in image domain, several regularization techniques, such as data augmentation, Weight decay [14], Dropout [18], Label smoothing [21], and Batch normalization (BN) [11] are frequently used.

Several recent studies proposed data augmentation on input data space by randomly occluding some image regions [3], interpolating two images [30] or transplanting an image patch onto another image [29].

By extension, model's internal feature has become another target for a regularization in recent studies. Shake-Shake regularization [5] proposed to regularize multi-branch ResNet by adding randomly scaled branches in both forward and backward computation, which cannot be applicable to 2-branch ResNet. Stochastic depth [10], also known as RandomDrop, randomly switches between dropping and connecting residual branch. Combining those two previous studies, ShakeDrop [27] adopted the switching mechanism of RandomDrop into Shake-Shake to stabilize training so that it can be also compatible with 2-branch ResNet.

3. Methods

In this section, we introduce our proposed regularization method, Random Mean Scaling (RMS), which adds perturbation on the spatio-temporally smoothed feature. For practical implementation, we further explain the method as a network module.

3.1. Random Mean Scaling

Perturbing the feature with multiplicative noise is a simple regularization method that often used in various ways [10, 5, 27]. In this work, we name a specific perturbation method which modulates the magnitude by multiplying a single random scalar α to a entire feature map as Random Scaling (RS). α can be sampled from a given probability distribution, *e.g.* Gaussian $N(\mu, \sigma)$.

In this work, we propose Random Mean Scaling (RMS), which apply RS to the local mean of features rather than to the feature directly. The local mean is a weighted average within a local window and can be calculated as

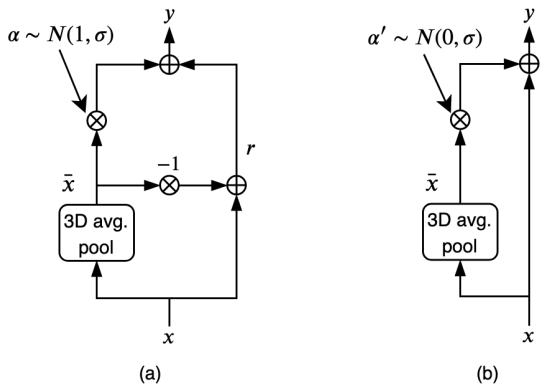

Figure 2: **Random Mean Scaling (RMS) module.** $\mathbf{x}$, $\bar{\mathbf{x}}$, $\mathbf{r}$, and $\mathbf{y}$ are the input, the mean of input, the residual and the output respectively. $\oplus$ and $\otimes$ are the element-wise sum and multiplication. (a) and (b) are the same operation. (a) explicitly separates $\bar{\mathbf{x}}$ and $\mathbf{r}$ for the explanation purpose while (b) is simplified form.

$$\bar{x}_i = \sum_{j \in W_i} w_j x_j, \tag{1}$$

where x is the input feature, W_i is a 3-dimensional local window around current index i, and w_j is a weight of each neighboring position. We simply choose $w_j = 1/\|W_i\|$ for all j, which is a mean filter, as our default setting. One alternative of the mean filter is Gaussian filter whose weight formulation can be found in Appendix C. We decompose the input $\mathbf{x}$ into the mean $\bar{\mathbf{x}}$ and the residual $\mathbf{r}$ as

$$\mathbf{r} = \mathbf{x} - \bar{\mathbf{x}}. \tag{2}$$

The modulated output $\mathbf{y}$ by RMS can be represented as following

$$\mathbf{y} = \begin{cases} \alpha\bar{\mathbf{x}} + \mathbf{r}, & \text{in training} \\ E[\alpha]\bar{\mathbf{x}} + \mathbf{r}, & \text{in test.} \end{cases} \tag{3}$$

The perturbation is applied only during training. If the mean of the probability distribution of alpha is one, $\mathbf{y} = \mathbf{x}$ during inference. The method can be applied to any level in a layer, such as the output of convolution, BN, or nonlinear activation.

In the experiment section, we explore the position of RMS. Additionally, we show that applying RS to $\bar{\mathbf{x}}$ improves the performance more than applying RS to $\mathbf{r}$ or $\mathbf{x}$. $\bar{\mathbf{x}}$ can be interpreted as the low-frequency component of $\mathbf{x}$, while $\mathbf{r}$ represents the remaining high-frequency component.

3.2. Implementation

RMS can be implemented as a network module with several basic operations as illustrated in Fig.2. Mean filter is

identical operation to 3D average pooling provided by most deep learning frameworks. Gaussian filter also can be implemented with basic tensor operations. Therefore, Eq.3 can be represented as a diagram in Fig.2 (a). Note that $\oplus$ and $\otimes$ are the element-wise sum and element-wise multiplication respectively. For a practical implementation, Eq.3 can be modified into a simpler form by substituting Eq.2 as

$$\mathbf{y} = \begin{cases} \mathbf{x} + \alpha'\bar{\mathbf{x}}, & \text{in training} \\ \mathbf{x} + E[\alpha']\bar{\mathbf{x}}, & \text{in test,} \end{cases} \tag{4}$$

where $\alpha' = \alpha - 1$. Fig.2 (b) shows the block diagram of the RMS module which corresponds to Eq.4. Since RMS module requires only simple operations such as an average pooling and a scalar multiplication, it is parameter free and only small amount of additional computation is required during training. Furthermore, no additional computation is required during inference.

4. Experiments

In this section, we experiment our proposed method, RMS, on several action recognition datasets. First, we investigate our method on Mini-Kinetics dataset [26] with various design choices and hyperparameters. Mini-Kinetics is chosen as the main testbed for its relatively small size, which makes a model prone to overfitting so that the regularization effect becomes more significant. We also evaluate RMS with various networks including SlowOnly, CSN [23] and SlowFast [4] on Kinetics-400 dataset [12]. Furthermore, finetuing result of SlowFast with RMS in Something-Something-v2 (Something-v2) dataset [6] will be presented. All experiments were conducted on NAVER Smart Machine Learning (NSML) [19, 13] platform.

4.1. Experimental Setup

Datasets. Kinetics-400 is a large scale action recognition dataset which is a collection of 10-second long trimmed video clips from YouTube video categorized into 400 action classes. The total number of videos in the dataset is around 300K and they are divided into training, validation, and test set with around 240K, 20K, and 40K videos respectively [1]. Mini-Kinetics dataset consists of 200 largest classes from Kinetics-400. The videos of Mini-Kinetics are randomly sampled from those of Kinetics-400 dataset. The set has 80K training samples and 5K validation samples. Something-v2 dataset is another large collection of human-object interaction videos with 174 classes of detailed action descriptions. It contains around 220k videos in total.

[1] About 8% of videos in train set were not available at the time we downloaded

Network. For the experiment, we mainly use SlowOnly, the slow branch detached from SlowFast [4] as our baseline model. SlowOnly maintain temporal resolution without using temporal stride and use 3D convolutions only at the later stages. It can be considered as a typical extension of 2D ResNet to 3D. In order to examine RMS on both the basic block and bottleneck block, we use SlowOnly-34 and SlowOnly-50 in experiments. Details of the model are specified in the Appendix A. All models are trained from scratch to examine the effect of regularization clearly in our study. Only RGB frames are fed into the model.

For RMS module, we choose a $[3, 3, 3]$ kernel with stride of 1 for average pooling and $N(1.0, 0.5)$ for sampling α as our default settings for the experiments. The sampling distribution is chosen based on the results in Table 1 (b). The module is added only in res_4 and res_5 blocks.

Preprocessing. For both training and evaluation, we first set frame-rate of all videos to 25 fps. We drop frames when the original fps is larger and duplicate frames otherwise. Since we utilize only RGB information for the network input, no handcrafted features, *e.g.* optical flow, are used.

Training. For training, N consecutive frames are randomly sampled from a video, then T frames of equidistant intervals from the N frames are fed into the model. We set N and T to 64 and 8 respectively in most of our experiments. For spatial augmentation, we apply random resized crop [20] and random horizontal flip. Random resized crop samples a patch of randomly chosen size (between 25% to 100% of the image area) and aspect ratio (between 3/4 and 4/3) then scale the patch size to 224×224. Other implementation details may be specified in Appendix B.1.

Evaluation. For evaluation, we follow the commonly used 30-crop scheme [4]. Ten clips are uniformly sampled along time from the entire video and three spatial regions of size 256×256 are uniformly sampled along the longer side of the frame. The final prediction is acquired by averaging softmax scores of all clips.

4.2. Mini-Kinetics Experiments

On Mini-Kinetics dataset, we train SlowOnly-34 and SlowOnly-50 to examine the proposed module in both the basic block and the bottleneck block. We first report the performance depending on several design choices. We found that the performance varies depending on the initialization of a model. Thus, we run each experiment three times and report their mean accuracy for reliability. Then we compare our model with some models from previous works.

Method	Top1 Acc.(%)	Top5 Acc.(%)
SlowOnly-34	74.7±0.33	92.1±0.06
+ RMS before the 1st conv.	76.7±0.23	93.2±0.27
+ RMS before the 1st BN	76.9±0.54	93.2±0.30
+ RMS before the 1st ReLU	76.4±0.25	93.2±0.09
+ RMS before the 2nd conv.	76.8±0.38	93.3±0.12
+ RMS before the 2nd BN	76.7±0.23	93.2±0.12
+ RMS before the 2nd ReLU	76.3±0.36	93.0±0.12
+ RMS before all BNs	**77.0**±0.31	**93.5**±0.23
SlowOnly-50	77.5±0.85	93.2±0.42
+ RMS before the 1st BN	78.4±0.39	93.8±0.11
+ RMS before the 2nd BN	77.8±0.55	93.8±0.21
+ RMS before the 3rd BN	**78.6**±0.33	**94.0**±0.12

(a) **Position of RMS.** Performance depending on the position of the RMS module inside a residual branch.

Method	Distribution	Top1 Acc.(%)	Top5 Acc.(%)
SlowOnly-34	-	74.7±0.33	92.1±0.06
SlowOnly-34	$N(1.0, 0.3)$	76.1±0.76	92.9±0.45
+ RMS	$N(1.0, 0.5)$	**77.0**±0.31	**93.5**±0.23
	$N(1.0, 0.7)$	76.9±0.20	93.5±0.37
	$U(0.5, 1.5)$	75.7±0.43	92.6±0.29
	$U(0.0, 2.0)$	76.4±0.36	93.4±0.45
	$U(-0.5, 2.5)$	74.5±0.58	92.6±0.16

(b) **Sampling distribution α of RMS.** Normal and uniform distribution are considered. SlowOnly-34 used.

Method	std. of α	Top1 Acc.(%)	Top5 Acc.(%)
SlowOnly-34	-	74.7±0.33	92.1±0.06
RS on $\bar{\mathbf{x}}$ (RMS)	0.3	76.1±0.76	92.9±0.45
RS on $\bar{\mathbf{x}}$ (RMS)	0.5	**77.0**±0.31	**93.5**±0.23
RS on $\mathbf{x}$	0.3	75.6±0.23	92.9±0.15
RS on $\mathbf{x}$	0.5	75.4±0.40	93.0±0.42
RS on $\mathbf{r}$	0.3	75.0±0.26	92.4±0.13
RS on $\mathbf{r}$	0.5	75.6±0.77	92.5±0.37

(c) **RS on different feature components.** Comparing the effect of RS on mean, residual and feature. RMS outperforms the others

Method	LPF (std.)	Top1 Acc.(%)	Top5 Acc.(%)
SlowOnly-34	-	74.7±0.33	92.1±0.06
+ RMS	MF	77.0±0.31	93.5±0.23
+ RMS	GF (0.4)	**77.5**±0.16	**93.9**±0.03
+ RMS	GF (1.0)	77.2±0.19	93.7±0.23
+ RMS	GF (2.0)	77.2±0.29	93.6±0.08
SlowOnly-50	-	77.5±0.85	93.2±0.42
+ RMS	MF	**78.6**±0.33	94.0±0.12
+ RMS	GF (0.4)	78.3±0.33	**94.1**±0.11

(d) **Type of low-pass filter in RMS.** Performance depending on the type of LPF; mean filter (MF) and Gaussian filter (GF).

Table 1: **Ablation studies on Mini-Kinetics dataset.** All accuracies are reported in mean±std. of multiple runs.

Position of RMS module. As mentioned earlier, RMS module can be applied to any level in a layer, such as 1) before each convolution, 2) before each BN and 3) before each ReLU. Note that RMS before the last ReLU is located before the summation of main branch and residual branch. The results are shown in Table 1 (a). We examined all possible positions on SlowOnly-34. First of all, regardless of the position, the RMS module improves classification accuracy compared to the baseline model. The difference between RMS positions is not significant, but we found that the RMS module before the first BN shows best Top-1 mean accuracy among the cases using a single RMS module. Since RMS module before the second BN also shows reasonable performance, we tested putting RMS module before every BN and it shows the best Top-1 and Top-5 accuracy among all. So we added the RMS module before all BNs in the remaining experiments.

We also tested RMS module in SlowOnly-50 which has the bottleneck structure. Taking the result from the basic block into account, we decided to investigate positions only before the BN. As can be seen in the below part of Table 1 (a), RMS module is beneficial to the network in all three cases. Because RMS module before the last BN shows the best Top-1 and Top-5 accuracy, we choose it as our default setting for the bottleneck block. However, RMS module before the first BN can be an efficient choice without a large performance drop, considering the number of channels in the last BN is four times larger than the others in bottleneck block. For computational efficiency, we choose not to use multiple RMS modules in bottleneck block.

Effect of sampling distribution. Next, we compare several different probability distributions for sampling α: normal distributions with three different σ and uniform distributions with three different ranges. As shown in Table 1 (b), the normal distributions generally perform better than the uniform distribution. Considering the characteristics of each distribution, generating enough number of samples around mean may help RMS improve the performance.

Comparing RS on different feature components. We compare the effect of RS on different feature components with SlowOnly-34 on Mini-Kinetics dataset. Among many possible directions in the feature space, we chose three directions to be examined: $\bar{\mathbf{x}}$, $\mathbf{r}$ and $\mathbf{x}$. The result, shown in Table 1 (c), provides the empirical evidence of our conjecture that applying RS module on mean ($\bar{\mathbf{x}}$) is more effective than $\mathbf{r}$ or $\mathbf{x}$. We found that RS on $\bar{\mathbf{x}}$ gains about

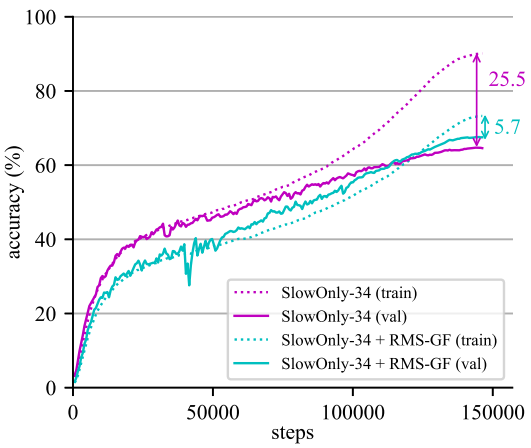

Figure 3: **RMS relieves overfitting problem.** Training accuracy (dotted line) and validation accuracy (solid line) of SlowOnly-34 and SlowOnly-34 + RMS-GF on Mini-Kinetics dataset. The gap between train and val. curve of SlowOnly-34 is more than four times larger than that of SlowOnly-34 + RMS-GF at the last step.

2% on Top-1 accuracy over the baseline. It is more than 1% higher than the others, RS on $\mathbf{r}$ or $\mathbf{x}$. We discuss three methods of applying RS to the model and their effects in the discussion section.

Comparing RMS with different low-pass filters. RMS uses mean filter (MF), also known as box filter, by default to extract low-frequency component from features. Gaussian filter (GF) is another type of LPF which can replace the MF. We investigated 3×3 GF version of RMS, RMS-GF, on Mini-Kinetics dataset as in Table 1 (d). The standard deviation (σ_f) of GF is also presented in the table. We found that GF generally performs similar with or better than MF in both SlowOnly-34 and SlowOnly-50. It is promising that Top-1 and Top-5 accuracies of SlowOnly-34 with RMS-GF reaches or surpasses those of SlowOnly-50 baseline when σ_f is 0.4. We choose $\sigma_f = 0.4$ as a default value of RMS-GF in the following.

Analysis of training curve. As a regularizer, RMS module is expected to reduce the gap between training accuracy and validation accuracy. Fig. 3 illustrates training and validation accuracy of the SlowOnly-34 and the same with RMS-GF module, RMS-GF for short, over training iterations. The training accuracy is a single crop accuracy while the validation accuracy is a three-crop (over time) average accuracy. It is noted that training accuracy of the RMS-GF is always lower than that of the baseline whereas validation accuracy of RMS-GF is lower in the beginning but it overtakes the baseline in the later stage of training.

Method	level	Top1 Acc.(%)
SlowOnly-34	-	74.7±0.33
+ RMS ($\alpha \sim (1, 0.5)$)	Clip	77.0±0.31
+ RMS-GF ($\alpha \sim (1, 0.5)$)	Clip	**77.5**±0.16
+ RandomDrop ($\alpha = 0, \beta = 0$)	Batch	75.6±0.14
+ ShakeDrop ($\alpha = 0, \beta \sim [0, 0.5]$)	Pixel	77.3±0.19
+ Cutmix ($\lambda \sim Beta(1, 1)$)	Batch	76.3±0.28
SlowOnly-50	-	77.5±0.85
+ RMS ($\alpha \sim (1, 0.5)$)	Clip	**78.6**±0.33
+ RMS-GF ($\alpha \sim (1, 0.5)$)	Clip	78.3±0.33
+ RandomDrop ($\alpha = 0, \beta = 0$)	Batch	76.6±0.25
+ ShakeDrop ($\alpha = 0, \beta \sim [0, 0.5]$)	Pixel	78.3±0.01
+ Cutmix ($\lambda \sim Beta(1, 1)$)	Batch	78.2±0.27

Table 2: **Comparing RMS to other regularization methods.** All methods are tested with SlowOnly on Mini-Kinetics dataset. (μ, σ) and $[a, b]$ denote $N(\mu, \sigma)$ and $U(a, b)$ respectively. Accuracies are averaged over multiple runs.

At the end of training, the accuracy gap between training and validation of the baseline (25.5) is much larger than that of RMS (11.7) and RMS-GF (5.7). This indicates that the RMS module is working as a regularizer preventing the model from overfitting.

Comparison with existing regularizations. To our best knowledge, this work is the first attempt to apply regularization for action recognition. Therefore, we compare RMS with other regularization methods, originally examined in image recognition. For comparison, we examine RandomDrop [10], ShakeDrop [27], and Cutmix [29] with SlowOnly on the Mini-Kinetics dataset. Details of hyper-parameters for each method are presented in Appendix D. The major difference of RMS from other regularizations is that RMS extracts low-frequency components from the feature and perturbs it selectively. Table 2 shows Top-1 accuracy with their hyper-parameters. RMS performs comparable to or better than three state-of-the-art methods with both SlowOnly-34 and SlowOnly-50.

Evaluation result. In Table 3, we compare our baseline model and proposed model with several previous works [26, 28, 9] evaluated on Mini-Kinetics dataset. Except for S3D [9] model, I3D model with self-attention operations like Non-Local (NL) [25], Compact Generalized NL (CGNL) [28] and Compact Global Descriptor (CGD) [9] are compared since only these works reported the performance on Mini-Kinetics to our best knowledge. The table shows that our proposed model achieves comparable performance to the other models on Mini-Kinetics dataset. Compared to the baseline model, RMS module substantially increases the performance in both SlowOnly-34 and

Method	Top1 Acc.(%)	Top5 Acc.(%)
S3D [26]	78.4	-
ResNet-50 + CGD [9]	77.56	93.20
ResNet-50 + NL×5 [28]	77.53	94.00
ResNet-50 + CGNL×5 [28]	78.79	94.37
SlowOnly-34	74.7†	92.1†
SlowOnly-34 + RMS	77.0†	93.5†
SlowOnly-34 + RMS-GF	77.5†	93.9†
SlowOnly-50	77.5†	93.2†
SlowOnly-50 + RMS	78.6†	94.0†
SlowOnly-50 + RMS-GF	78.3†	94.1†

Table 3: **Performance on Mini-Kinetics dataset.** † denotes average accuracies over 3 runs.

Method	Input size	Top1 Acc.(%)	Top5 Acc.(%)
SlowOnly	8×224^2	73.0	90.9
+ RMS	8×224^2	74.2	91.5
+ RMS-GF	8×224^2	74.8	91.6
ip-CSN (our imple.)	8×224^2	69.7	88.6
+ RMS	8×224^2	70.5	89.6
SlowFast (our imple.)	32×224^2	75.0	92.1
+ RMS	32×224^2	76.3	92.5

Table 4: **Performance on Kinetics-400 dataset.** RGB input only. all models use ResNet-50 backbone. (* denotes evaluation with 10-crops.)

SlowOnly-50 without any additional parameter. SlowOnly-34 with RMS before all BNs shows an increase in Top-1 accuracy by 2.3% compared to the baseline. RMS-GF improves even more; 2.8% in Top-1 accuracy. In the case of SlowOnly-50, there is a 1.1% increase in the Top-1 accuracy when RMS added before the last BN. RMS-GF improves 0.8% in Top-1 accuracy compared to baseline.

4.3. Kinetics-400 Experiments

In Kinetics-400, we trained not only SlowOnly but also other recently proposed models, SlowFast and CSN, with our training setting. We choose two models as a baseline since SlowFast shows state-of-the-art performance without pretraining and interaction preserved CSN (ip-CSN) is a lightweight 3D model. All models use ResNet-50 as their backbone. In this section, we focus on how our proposed RMS module affects the baseline network in a large scale action recognition dataset.

Evaluation result. In Table 4, we present the performance of SlowOnly, SlowFast and CSN with RMS module. All models are tested with our implementation. The input resolution of each models are also presented in the table. All the works use only RGB frames as input and trained without pretraining. Top-1 and Top-5 accuracies of SlowFast reported in [4] were 77.0 and 92.6. Top-1 accuracy of ip-CSN in [23] with 10-crop evaluation was 70.8. The difference in baseline performance of our implementation from that reported in the original paper might be caused by different training environments[2].

According to the table, we found that RMS module improves performance over the baseline for every case. First, SlowOnly with RMS and RMS-GF shows an increase in Top-1 accuracy by 1.2% and 1.8% from the baseline. RMS also improves performance of two recent models, SlowFast and CSN, by 0.8% and 1.3% in Top-1 accuracy from the baseline respectively. This shows that RMS is also effective to both a more complex model and a lightweight model. So we can conclude that RMS can be generally applicable and is not limited to a certain type of network architecture.

Method	Top1 Acc.(%)	Top5 Acc.(%)
SlowFast-50	59.5	86.6
SlowFast-50 + RMS	61.2	87.6

Table 5: **Performance on Something-Something-v2 dataset.** Models are finetuned from Kinetics pretrained weights.

Dataset	Method	Top1 Acc.(%)	Top5 Acc.(%)
CIFAR-100	ResNet-110	74.49*	93.18*
	ResNet-110+RMS	77.17*	94.15*
ImageNet	ResNet-50	76.81	93.26
	ResNet-50+RMS	77.62	93.91

Table 6: **RMS on 2D Models.** Effect of RMS with 2D models on image recognition datasets. * represents accuracies averaged over multiple runs.

4.4. Something-Something-v2 Experiments

For Something-v2 dataset, we finetune SlowFast pretrained on Kinetics-400. SlowFast is chosen since Something-v2 requires finer temporal details. We set the initial learning rate 0.01 with step-wise decaying schedule for finetuning. Other training and evaluation settings are similar to the Kinetics-400 experiment. Further details may be found in the supplementary material.

Evaluation result. Table 5 shows the evaluation result of the model in Something-v2 dataset. RMS improves both Top-1 and Top-5 accuracy by 1.7% and 1.0% respectively from the baseline. The result shows that effectiveness of RMS is not limited to a certain dataset.

5. Discussion

In this section, we first analyze how RMS changes the model's response to the modulation of low-frequency and

[2]We suspect that 1) different batch size due to GPU limitation, 2) difference in implementation details, and 3) reduced dataset due to blocked videos and lower fps, might cause the discrepancy.

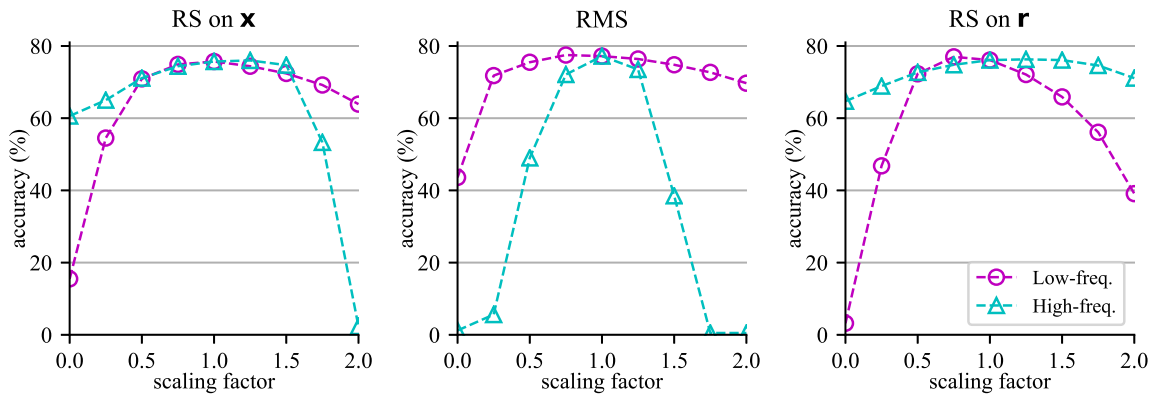

Figure 4: **RMS makes the model more fit to high-frequency components of the feature.** Evaluation of SlowOnly-34 on Mini-Kinetics validation dataset with scaling the magnitude of the low-freq. ($\bar{\mathbf{x}}$, circle marker) and high-freq. ($\mathbf{r}$, triangle marker) components of the feature. While one component is scaled, the other component remains unchanged.

high-frequency components of the feature. We assume that the difference caused by RMS makes our proposed method better than the baseline and RS on alternatives. Second, we evaluate RMS on two image recognition datasets, CIFAR-100 and ImageNet to show that RMS can be utilized in 2D models.

5.1. Analysis on Effect of RMS

Table 1 (c) shows RS on $\bar{\mathbf{x}}$ (RMS) is more effective than the others, RS on $\mathbf{x}$ and RS on $\mathbf{r}$, though all cases still improve accuracy over the baseline. This implies that adding perturbation to features has regularization effect in general but $\bar{\mathbf{x}}$ is a better direction to perturb for the performance. However, it is ambiguous how a model is affected by the direction of perturbation. To investigate its influence, we repeated the same experiment as Fig. 1 on all three cases; $\bar{\mathbf{x}}$, $\mathbf{x}$, and $\mathbf{r}$. Note that we scaled the feature components before all BNs in res_4 and res_5 stages, where the RMS module is applied, during inference. Fig. 4 shows the variation of accuracy depending on the modulation of low-frequency or high-frequency components. It is fairly obvious that RS makes the model robust to the modulation of its targeted components during the training. RMS has a significant distinction from the others in that it makes the model much sensitive to the scaling of the high-frequency component than that of the low-frequency component. We observe that RMS makes the sensitivity gap between two components larger compared to the baseline. In contrast, RS on $\mathbf{x}$ and $\mathbf{r}$ makes the model less sensitive to the scaling of both frequency components, resulting in the sensitivity difference between two components becoming relatively smaller than RMS. One might think that robustness in both directions is advantageous for the generalization but the result in Table 1 (c) is against the intuition. Taking these observations into account, we conjecture that models may have better generalization capability when they utilize more high-frequency information rather than low-frequency information for action recognition. It supports our argument in Fig. 1 that high-frequency components of the feature is more important for the action classification. The ground for the difference between RMS and the others remains for the future work.

5.2. Applying RMS to 2D Model

In this work, we focus on exploring regularization effect in action recognition. However, it is obvious that RMS can be applied to 2D ResNet by using 2D average pooling. So we briefly tested RMS in two popular image recognition datasets, CIFAR-100 and ImageNet. RMS with $\sigma = 0.5$ is tested in both datasets. Please refer to the Appendix B.3 for further details. Table 6 shows that RMS substantially improves classification performance in both datasets. The results show that RMS is also applicable in 2D models.

6. Conclusion

In this work, we propose a new regularization method, RMS for 3D ResNet, which often encounters an overfitting problem. To the best of our knowledge, this study is the first attempt to explore utilizing a regularization for video action recognition. We examine RMS experimentally on Mini-Kinetics dataset with extensive ablations studies and analysis. In Kinetics-400 and Something-Something-v2, we showed that RMS enhances the generalization ability of several baseline models. RMS opens up the possibilities of using regularization methods to improve the generalization of an action recognition model. More comprehensive investigation into various methods, models, and datasets will help further improvement for action recognition.

References

[1] João Carreira and Andrew Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4724–4733, 2017. 1, 2

[2] Yunpeng Chen, Haoqi Fan, Bing Xu, Zhicheng Yan, Yannis Kalantidis, Marcus Rohrbach, Shuicheng Yan, and Jiashi Feng. Drop an octave: Reducing spatial redundancy in convolutional neural networks with octave convolution. *arXiv:1904.05049*, 2019. 1, 2

[3] Terrance DeVries and Graham W Taylor. Improved regularization of convolutional neural networks with cutout. *arXiv:1708.04552*, 2017. 1, 3

[4] Christoph Feichtenhofer, Haoqi Fan, Jitendra Malik, and Kaiming He. Slowfast networks for video recognition. In *The IEEE International Conference on Computer Vision (ICCV)*, October 2019. 1, 2, 4, 7

[5] Xavier Gastaldi. Shake-shake regularization. *arXiv:1705.07485*, 2017. 1, 3

[6] Raghav Goyal, Samira Ebrahimi Kahou, Vincent Michalski, Joanna Materzynska, Susanne Westphal, Heuna Kim, Valentin Haenel, Ingo Fruend, Peter Yianilos, Moritz Mueller-Freitag, Florian Hoppe, Christian Thurau, Ingo Bax, and Roland Memisevic. The "something something" video database for learning and evaluating visual common sense. In *The IEEE International Conference on Computer Vision (ICCV)*, Oct 2017. 2, 4

[7] Kensho Hara, Hirokatsu Kataoka, and Yutaka Satoh. Can spatiotemporal 3d cnns retrace the history of 2d cnns and imagenet? In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6546–6555, 2018. 1, 2

[8] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016. 2

[9] Xiangyu He, Ke Cheng, Qiang Chen, Qinghao Hu, Peisong Wang, and Jian Cheng. Compact global descriptor for neural networks. *arXiv:1907.09665*, 2019. 3, 6, 7

[10] 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. 1, 2, 3, 6

[11] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32Nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, pages 448–456. JMLR.org, 2015. 3

[12] Will Kay, João Carreira, Karen Simonyan, Brian Zhang, Chloe Hillier, Sudheendra Vijayanarasimhan, Fabio Viola, Tim Green, Trevor Back, Paul Natsev, Mustafa Suleyman, and Andrew Zisserman. The kinetics human action video dataset. *arXiv:1705.06950*, 2017. 1, 4

[13] Hanjoo Kim, Minkyu Kim, Dongjoo Seo, Jinwoong Kim, Heungseok Park, Soeun Park, Hyunwoo Jo, KyungHyun Kim, Youngil Yang, Youngkwan Kim, Nako Sung, and Jung-Woo Ha. NSML: meet the mlaas platform with a real-world case study. *arXiv:1810.09957*, 2018. 4

[14] Anders Krogh and John A. Hertz. A simple weight decay can improve generalization. In J. E. Moody, S. J. Hanson, and R. P. Lippmann, editors, *Advances in Neural Information Processing Systems 4*, pages 950–957. Morgan-Kaufmann, 1992. 3

[15] Chao Li, Qiaoyong Zhong, Di Xie, and Shiliang Pu. Collaborative spatiotemporal feature learning for video action recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 2

[16] Mark Nixon and Alberto S. Aguado. *Feature Extraction & Image Processing for Computer Vision, Third Edition*. Academic Press, Inc., Orlando, FL, USA, 3rd edition, 2012. 2

[17] Zhaofan Qiu, Ting Yao, and Tao Mei. Learning spatio-temporal representation with pseudo-3d residual networks. *The IEEE International Conference on Computer Vision (ICCV)*, pages 5534–5542, 2017. 1, 2

[18] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958, 2014. 3

[19] Nako Sung, Minkyu Kim, Hyunwoo Jo, Youngil Yang, Jingwoong Kim, Leonard Lausen, Youngkwan Kim, Gayoung Lee, Dong-Hyun Kwak, Jung-Woo Ha, and Sunghun Kim. NSML: A machine learning platform that enables you to focus on your models. *arXiv:1712.05902*, 2017. 4

[20] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015. 2, 4

[21] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 3

[22] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri. Learning spatiotemporal features with 3d convolutional networks. In *The IEEE International Conference on Computer Vision (ICCV)*, pages 4489–4497, 2015. 1, 2

[23] Du Tran, Heng Wang, Lorenzo Torresani, and Matt Feiszli. Video classification with channel-separated convolutional networks. In *The IEEE International Conference on Computer Vision (ICCV)*, October 2019. 2, 4, 7

[24] Du Tran, Heng Wang, Lorenzo Torresani, Jamie Ray, Yann LeCun, and Manohar Paluri. A closer look at spatiotemporal convolutions for action recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6450–6459, 2018. 1, 2

[25] Xiaolong Wang, Ross B. Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7794–7803, 2018. 3, 6

[26] Saining Xie, Chen Sun, Jonathan Huang, Zhuowen Tu, and Kevin Murphy. Rethinking spatiotemporal feature learning for video understanding. *arXiv:1712.04851*, 2017. 1, 2, 4, 6, 7

[27] Yoshihiro Yamada, Masakazu Iwamura, and Koichi Kise. Shakedrop regularization. *arXiv:1802.02375*, 2018. 1, 2, 3, 6

[28] Kaiyu Yue, Ming Sun, Yuchen Yuan, Feng Zhou, Errui Ding, and Fuxin Xu. Compact generalized non-local network. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 6510–6519. Curran Associates, Inc., 2018. 3, 6, 7

[29] 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, 3, 6

[30] Hongyi Zhang, Moustapha Cissé, Yann N. Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. In *International Conference on Learning Representations, ICLR*, 2018. 1, 3