Convolutional Neural Networks with Dynamic Regularization

Yi Wang*, *Student Member, IEEE*, Zhen-Peng Bian*, Junhui Hou, *Member, IEEE*, and Lap-Pui Chau, *Fellow, IEEE*

Abstract—Regularization is commonly used in machine learning for alleviating overfitting. In convolutional neural networks, regularization methods, such as Dropout and Shake-Shake, have been proposed to improve the generalization performance. However, these methods are lack of self-adaption throughout training, i.e., the regularization strength is fixed to a predefined schedule, and manual adjustment has to be performed to adapt to various network architectures. In this paper, we propose a dynamic regularization method which can dynamically adjust the regularization strength in the training procedure. Specifically, we model the regularization strength as a backward difference of the training loss, which can be directly extracted in each training iteration. With dynamic regularization, the large model is regularized by the strong perturbation and vice versa. Experimental results show that the proposed method can improve the generalization capability of off-the-shelf network architectures and outperforms state-of-the-art regularization methods.

Index Terms—CNN, image classification, dynamic regularization, overfitting, generalization.

I. INTRODUCTION

CONVOLUTIONAL neural networks (CNNs), which use a stack of convolution operations followed by non-linear activation (e.g., Rectified Linear Unit, ReLU) to extract high-level discriminative features, have achieved considerable improvements for visual tasks [1], [2], [3]. Via layer-by-layer connectivity, extracted features can reach outstanding representational power. Recent advances of the CNN architectures, such as ResNet [2], DenseNet [4], ResNeXt [5], and PyramidNet [6], ease the problems of vanishing gradients and boost the performance. However, overfitting, which reduces the generalization capability of CNNs, is still a big problem.

A wide variety of regularization strategies were exploited to alleviate overfitting and decrease the generalization error. Data augmentation [1] is a simple yet effective manner to make models adapt to the diversity of data. Batch Normalization [7] standardizes the mean and variance of features for each mini-batch, which makes the optimization landscape smoother [8]. Drop-based methods [9], [10] aim to train an ensemble of sub-networks, which weakens the effect of "co-adaptions" on training data. Recently, Shake-Shake regularization [11] was proposed to randomly interpolate two complementary features

Yi Wang and Lap-Pui Chau are with School of Electrical and Electronics Engineering, Nanyang Technological University, Singapore, 639798 (e-mail: {wang1241@e.ntu.edu.sg, elpchau@ntu.edu.sg).

Zhen-Peng Bian is with Singapore Telecommunications Limited, Singapore, 239732 (e-mail: zbian1@ntu.edu.sg).

Junhui Hou is with Department of Computer Science, City University of Hong Kong (e-mail: jh.hou@cityu.edu.hk).
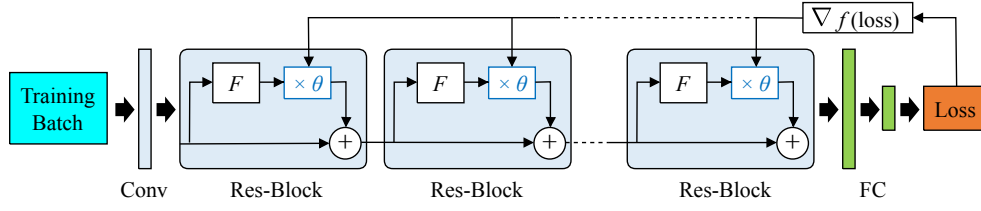
*Authors contributed equally.

in the two branches of ResNeXt, achieving state-of-the-art classification performance. ShakeDrop [12] incorporated the idea of Stochastic Depth [13] with Shake-Shake regularization to stabilize the training process in the residual branch of ResNet or PyramidNet. Despite the impressive improvement by Shake-based regularization methods, there are two main drawbacks with this type of methods.

1) ShakeDrop regularization was designed for deep networks and not suitable for shadow network architectures. It may not improve generalization performance, and even make the performance worse for shadow networks (see the TABLE I).
2) The regularization strength (or amplitude) is unchangeable over the whole training process. The fixed strong regularization is beneficial to reduce overfitting, but it causes difficulties to fit data at the beginning of training. From the perspective of curriculum learning [14], the learner needs to begin with easy examples.

In view of these issues, we propose a dynamic regularization method for CNNs, in which the regularization strength is adaptable to the dynamics of the training loss. During training, the dynamic regularization strength can be gradually increased with respect to the training status. Analogous to human education, the regularizer is regarded as an instructor who gradually increases the difficulty of training examples by way of feature perturbation. Moreover, dynamic regularization can adapt to models with different sizes. It provides a strong regularization for large models and vice versa. (See Fig. 4 (b)). That is, the regularization strength grows faster and achieves the higher value for the large model than that of the light model.)

Fig. 1 shows the proposed dynamic regularization in the ResNet structure. The training loss is not only used to perform backpropagation but also exploited to update the amplitude of the regularization. The features are multiplied by the regularizer in the residual branch. The regularizer works as a perturbation which introduces an augmentation in feature space, so CNNs are trained by the diversity of augmented features. Additionally, the regularization amplitude is changeable with respect to the dynamics of the training loss. We conduct experiments on the image classification task to evaluate our regularization strategy. Experimental results show that the proposed dynamic regularization outperforms state-of-the-art regularization methods, i.e., PyramidNet and ResNeXt equipped with our dynamic regularization improve the classification accuracy in various model settings, when compared with the same network with ShakeDrop [12] and

Fig. 1. The proposed dynamic regularization in the ResNet structure. Conv denotes the convolutional layer. FC denotes the fully connected layer. F denotes the residual function. $\nabla f(loss)$ denotes a backward difference of the training loss. The dynamic regularization aims to make a self-adaptive schedule throughout training for various network sizes by adjusting the strength of the random perturbation θ. As a manner of feature augmentation, the θ introduces noises for the residual branch in the forward and backward process.

Shake-Shake [11] regularization.

The rest of this paper is organized as follows. We first briefly introduce the related work on deep CNNs and regularization methods in Section II. Then, the proposed dynamic regularization is presented in Section III. Experimental results and discussion are given in Section IV. Finally, Section V concludes this paper.

II. RELATED WORK

A. Deep CNNs

CNNs have become deeper and wider with a more powerful capacity [2], [4], [6], [15], [16]. As our proposed regularization is based on ResNet and its variants, we briefly review the basic structure of ResNet, i.e., residual block.

Residual block. The residual block (Res-Block, shown in Fig. 1) is formulated as

$$x_{l+1} = x_l + F(x_l, \mathcal{W}_l), \tag{1}$$

where an identity branch x_l is the input features of the lth Res-Block, which is added with a residual branch F that is non-linear transformations of the input x_l by a set of parameters $\mathcal{W}_l$ ($\mathcal{W}_l$ will be omitted for simplicity). F consists of two Conv-BN-ReLU or Bottleneck Architectures in the original ResNet structure [2]. In recent improvement, F can also be other forms, e.g. Wide-ResNet [17], Inception module [18], PyramidNet [6], and ResNeXt [5]. PyramidNet gradually increases the number of channels in the Res-Blocks as the layers go deep. ResNeXt has multiple aggregated residual branches expressed as

$$x_{l+1} = x_l + F_1(x_l) + F_2(x_l), \tag{2}$$

where F_1 and F_2 are two residual branches. The number of branches (namely cardinality) is not limited.

B. Regularization

In addition to the advances of network architectures, many regularization techniques, i.e., data augmentation [1], [19], stochastic drooping [10], [13], [20], and Shake-based regularization methods [11], [12], have been successfully applied to avoid overfitting of CNNs.

Data augmentation (e.g., random cropping, flipping, and color adjusting [1]) is a simple yet effective strategy to increase the diversity of data. DeVries and Taylor [19] introduced an image augmentation technique, in which augmented images are generated by randomly cutting out square regions from input images (called Cutout). Dropout [10] is a widely used technique which stochastically drops out the hidden nodes from the networks during the training process. Following this idea, Maxout [21], Continuous Dropout [22], DropPath [20], and Stochastic Depth [13] were proposed. Based on ResNet, Stochastic Depth randomly drops a certain number of residual branches so that the network is shrunk in training. It performs inference using the whole network without dropping. Shake-based regularization approaches [11], [12] was recently proposed to augment features inside CNNs, which achieves outstanding classification performance.

Shake-based regularization approaches. Gastaldi [11] proposed a Shake-Shake regularization method, as shown in Fig. 2 (a). A random variable α is used to control the interpolation of the two residual branches (i.e., $F_1(x)$ and $F_2(x)$ in 3-branch ResNeXt). It is given by:

$$x_{l+1} = x_l + \alpha F_1(x_l) + (1 - \alpha)F_2(x_l), \tag{3}$$

where $\alpha \in [0, 1]$ follows the uniform distribution in the forward pass. For the backward pass, α is replaced by another uniform random variable $\beta \in [0, 1]$ to disturb the learning process. The regularization amplitude of each branch is fixed to 1.

To extend the use of Shake-Shake regularization, Yamada *et al.* [12] introduced a single Shake in 2-branch architectures (e.g., ResNet or PyramidNet) as shown in Fig. 2 (b) in which they adopted Stochastic Depth [13] to stabilize the learning:

$$x_{l+1} = x_l + (b_l + \alpha - b_l\alpha)F(x_l), \tag{4}$$

where $\alpha \in [-1, 1]$ is an uniform random variable and $b_l \in \{0, 1\}$ is a Bernoulli random variable which decides to performs the original network (i.e., $x_{l+1} = x_l + F(x_l)$, if $b_l = 1$) or the perturbated one (i.e., $x_{l+1} = x_l + \alpha F(x_l)$, if $b_l = 0$). In backward pass, α is replaced by $\beta \in [0, 1]$. The regularization amplitude of the branch is also fixed to 1. In [12], Yamada *et al.* also presented a structure of *Single-branch Shake* without the original network: $x_{l+1} = x_l + \alpha F(x_l)$, in which the perturbation $\alpha \in [-1, 1]$ is applied in the feature space. They showed that this structure gets bad results in some cases. For instance, the 110-layer PyramidNet with *Single-branch Shake* drops the error rate to 77.99% on CIFAR-100. This fixed large regularization overemphasizes the overfitting. We argue that the fixed regularization amplitude cannot fit the dynamics of the training process and different model sizes well.
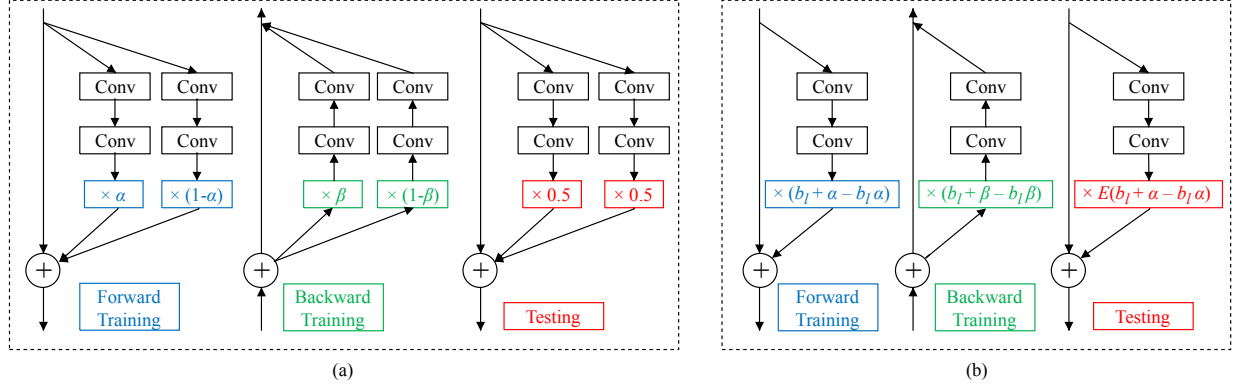
Fig. 2. Shake-based regularization methods in the Res-Block. Some layers (e.g., batch normalization and ReLU) in the residual branch is omitted for simplicity. (a) 3-branch architecture with Shake-Shake regularization [11]. (b) 2-branch architecture with ShakeDrop [12].

III. THE PROPOSED METHOD

As aforementioned, the fixed regularization strength in the existing regularization methods, such as DropPath [20], Stochastic Depth [13], Shake-Shake [11], and Shakedrop [12], departs from the human learning paradigm (e.g., the curriculum learning [14] or self-paced learning [23]). A naive way is to predefine the schedule for updating the regularization strength, such as the linear increment scheme in [24], which linearly increases the learning difficulty from low to high. We argue that the predefined schedule is not flexible enough to reveal the learning process. Inspired by the fact that the feedback of the learning itself can provide useful information, we propose a dynamic regularization, which is capable of adjusting the regularization strength adaptively.

Our dynamic regularization for CNNs is based on the dynamics of the training loss. Specifically, at the beginning of the training process, both the training and testing loss keeps decreasing, which means the network is learning to recognize the images. However, through a certain number of iterations, the network may overfit training data, resulting in that the training loss decreases more rapidly than the testing loss. The design of the regularization method needs to follow this dynamics. If the training loss drops in an iteration, the regularization strength should increase against overfitting in the next iteration; otherwise, the regularization strength should decrease against underfitting. In what follows, we first introduce the network architectures with dynamic regularization and then deliberate the update of the regularization strength in each iteration of the training process.

A. Network Architectures with Dynamic Regularization

We apply the dynamic regularization method on the two residual network architectures: the 2-branch architecture (e.g., PyramidNet [6]) and the 3-branch architecture (e.g., ResNeXt [5]).

1) The 2-branch architecture with dynamic regularization: **Training phase.** During training, dynamic regularization is adopted in Res-Block, as shown in Figs. 3 (a) and (b). Specifically, a dynamic regularization unit (called random
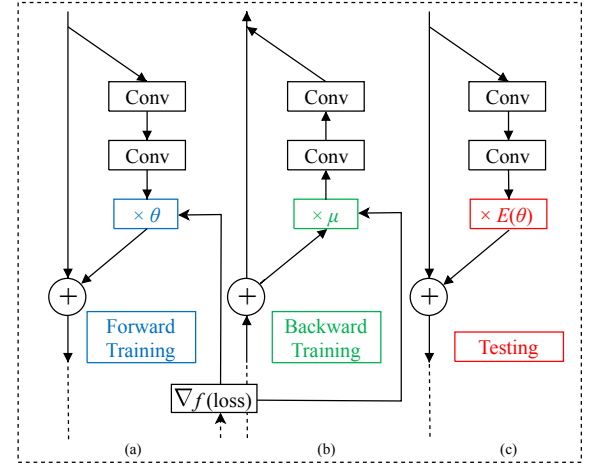

Fig. 3. The 2-branch Res-Block with dynamic regularization.

perturbation) is introduced into the residual branch of Res-Block. The random perturbation θ is achieved by

$$\theta = A + s_i \cdot r, \qquad (5)$$

where A is the basic constant amplitude, s_i is the dynamic factor at the ith iteration, and $r \in [-R, R]$ is the uniform random noise with the expected value $E(r) = 0$. The value of s_i is updated via the backward difference of the training loss (See Section III.B). The regularization amplitude is proportional to $A + s_i \cdot R$. In the forward pass, the output of the $(l+1)^{th}$ Res-Block can be expressed as:

$$x_{l+1} = x_l + (A + s_i \cdot r)F(x_l). \qquad (6)$$

In the backward pass, θ has a different value (represented by μ in Fig. 3 (b)) due to the random noise r.

Random noise. The range of r, i.e., R, is a hyper-parameter in the training phase. A straightforward way is to set R to be uniform inside all Res-Blocks. According to [13], the features of the earlier Res-Blocks should remain more than those of the later Res-Blocks. Hence, we propose a linear enhancement rule to configure this range inside Res-Blocks. For the lth Res-Block, the range denoted as R_l is given by

$$R_l = l/L, \qquad (7)$$

where L is the total number of Res-Blocks. With the increasing trend of the range R, the regularization strength is gradually raised from the bottom layer to the top layer. We conduct a comparison between different settings of R inside Res-Blocks in Section IV.

Inference phase. As shown in Fig. 3 (c), we calculate the expected value of θ:

$$E(\theta) = E(A + s_i \cdot r) = A, \tag{8}$$

and obtain forward pass for inference:

$$x_{l+1} = x_l + A \cdot F(x_l). \tag{9}$$

Since A is a constant, Eq. (9) is equivalent to the standard Res-Block.

2) The 3-branch architecture with dynamic regularization: We apply the dynamic regularization on a 3-branch architecture (See Fig. 2 (a)). Shake-Shake regularization is given by Eq. (3), in which $\alpha \in [0, 1]$ is a uniform random variable. We introduce the random perturbation θ in Eq. (5) to replace α in Eq. (3). Res-Block with dynamic regularization can be defined as

$$x_{l+1} = x_l + (A + s_i \cdot r)F_1(x_l) + (1 - A - s_i \cdot r)F_2(x_l), \tag{10}$$

If we set $A = 0.5$ and $r \in [-0.5, 0.5]$ and limit s_i equal to 1, θ ranges from 0 and 1, which is consistent with α in Eq. (3). The Shake-Shake regularization can be thought of as a special case of our dynamic regularization with a fixed dynamic factor.

B. Update of the Regularization Strength

The proposed updating solution for the dynamic regularization strength is achieved by the dynamics of the training loss. Specifically, the dynamic characteristic of the training loss can be model as the difference of the training loss between successive iterations. We define the backward difference between the training loss at two successive iterations as

$$\nabla loss_i = loss_i - loss_{i-1}, \tag{11}$$

where $loss_i$ denotes the training loss at the ith iteration. Although the training loss shows a downtrend in overall, there are huge fluctuations when feeding sequential mini-batches. To eliminate the noise and find out the trend of the loss, we apply a Gaussian filter to smooth it. Hence, the filtered backward difference can be rewritten as

$$\nabla f(loss_i) = f(loss_i) - f(loss_{i-1}), \tag{12}$$

where $f(\cdot)$ is the filtering operation defined as

$$f(loss_i) = \sum_{n=0}^{N} w[n] \cdot loss_{i-n}, \tag{13}$$

where the filter length is $N + 1$. We use the normalized Gaussian window $w[n]$ defined by

$$w[n] = \frac{1}{\sqrt{2\pi}(\sigma N/2)} e^{-\frac{1}{2}\left(\frac{n - N/2}{\sigma N/2}\right)^2}, \tag{14}$$

where $\sigma = 0.4$, and $0 \le n \le N$. The standard deviation is determined by $\sigma \cdot N/2$. We will discuss the Gaussian filter in

the experiment. The dynamic factor in Eqs. (6) and (10) with respect to $\nabla f(loss_i)$, i.e.,

$$s_{i+1} = \begin{cases} s_i + \Delta s, & \nabla f(loss_i) \le 0 \\ s_i - \Delta s, & \nabla f(loss_i) > 0 \end{cases} \tag{15}$$

where Δs is a small constant step for changing the regularization amplitude. From Eq. (15), it can be observed that if the training loss decreases ($\nabla f(loss_i) \le 0$), the regularization amplitude increases to avoid overfitting; otherwise, it decreases to prevent underfitting. The dynamic factor keeps updating to follow the dynamics of the training loss in each iteration of the training procedure.

Remark. There are some existing methods to change the regularization strength. For instance, Zoph *et al.* [24] introduced a ScheduledDropPath to regularize NASNets, which is a linear increment scheme of the regularization strength. The probability of dropping out a path is increased linearly throughout the training. However, the constant or linear scheme is a predefined rule, which cannot adapt to the training procedure and different model size. Different from them, our proposed dynamic scheduling exploits the dynamics of the training loss, which is applicable to the training procedure in different network architectures. In Section IV, we conduct comparisons between them.

IV. EXPERIMENTAL RESULTS

In this section, we evaluate the proposed dynamic regularization on the classification benchmark: CIFAR100 [25], in comparison with two state-of-the-art regularization approaches: Shake-Shake [11] and ShakeDrop [12]. Then we conduct ablation studies to compare with the fixed or linear-increment scheme of the regularization strength, and discuss the effectiveness of the Gaussian filter and the random noise.

A. Implementation Details

The following settings are used throughout the experiments. We set the training epoch to 300 and the batch size to 128. The learning rate was initialized to 0.1 for the 2-branch architecture as [12] and 0.2 for the 3-branch architecture as [11], and we used the cosine learning schedule to gradually reduce the learning rate to 0 at the end of training. For the dynamic regularization, we set the initial dynamic factor $s_0 = 0$, $A = 0.5$, and $\Delta s = 0.0003$ for the 2-branch architecture and $\Delta s = 0.00025$ for the 3-branch architecture. The length of the Gaussian filter was 501. PyramidNet [6] and ResNeXt [5] were used as baselines. We employed the standard translation, flipping [1] and Cutout [19] as the data augmentation scheme. Therefore, the Shake-based regularizer is the only one variable to affect experiments. All experimental results are presented by the average of 3 runs at the 300-th epoch.

B. Comparison with State-of-the-Art Regularization Methods

We first compare the proposed dynamic regularization with ShakeDrop in the 2-branch architecture on CIFAR100. Following the ShakeDrop, we used PyramidNet [6] as our baseline (namely Baseline) and chose different architectures including:

TABLE I
COMPARISON OF REGULARIZATION METHODS IN THE 2-BRANCH
ARCHITECTURE (I.E., PYRAMIDNET). TOP-1 ERROR RATES (%) ARE
SHOWN. DYNAMIC DENOTES THE PROPOSED REGULARIZATION METHOD.

Network Architecture	Params	Regularization	Top-1 Error (%)
PyramidNet-110-a48	1.8M	Baseline [6]	23.40
		ShakeDrop [12]	21.60
		Dynamic (ours)	**21.32**
PyramidNet-26-a84	0.9M	Baseline [6]	26.30
		ShakeDrop [12]	31.83
		Dynamic (ours)	**23.83**
PyramidNet-26-a200	3.8M	Baseline [6]	22.53
		ShakeDrop [12]	26.11
		Dynamic (ours)	**20.34**

1) PyramidNet-110-a48 (i.e., the network has a depth of 110 and a widening factor of 48) which is a deep and narrow network, 2) PyramidNet-26-a84 which is a light network, and 3) PyramidNet-26-a200 which is a shallow and wide network.

Table I shows the experimental results. From Table I, it can be observed that our dynamic regularization outperforms the counterparts of ShakeDrop in various architectures. The error rates of ShakeDrop are even worse than those of Baseline in the shallow architectures, i.e., PyramidNet-26-a84 and PyramidNet-26-a200, which means ShakeDrop with fixed regularization strength fails in this case. This issue comes from Stochastic Depth [13] in ShakeDrop where Stochastic Depth works well for deep networks. Regardless of the depth of networks, PyramidNet with dynamic regularization obtains a consistent improvement. Networks with the dynamic regularization are comparable with the baseline networks which has the double number of parameters (e.g., 23.83% of 26-a84-Dynamic v.s. 23.40% of 110-a48-Baseline; and 21.32% of 110-a48-Dynamic v.s. 22.53% of 26-a200-Baseline).

For the 3-branch architecture, we compare the dynamic regularization with Shake-Shake [11] in ResNeXt-26-2x32d (i.e., the network has the depth of 26 and the residual branch of 2, and the first residual block has the width of 32) and ResNeXt-26-2x64d as shown in Table II. We can see that the error rates of dynamic regularization are lower than those of Shake-Shake. The results from Tables I and II shows that our dynamic regularization can adapt to various network architectures.

Fig. 4 shows the training loss, dynamic factor, and Top-1 error with respect to the epoch in the two networks, i.e., PyramidNet-26-a84 and PyramidNet-110-a48. For networks with dynamic regularization, the downward trend of the training loss is slowed down, unlike Baseline in which the loss goes down towards zero (See Fig. 4 (a)). Dynamic regularization can prevent networks from rote learning the training data. As shown in Fig. 4 (b), the dynamic factor of the two network architectures gradually increases throughout the training process. Instead of using a predefined scheduling function in [24], our dynamic scheduling is self-adaptive according to the backward difference of training loss. Another important property of the dynamic scheduling is that a small regularization strength is generated for a light model (i.e., 26-a84), and a large strength is for a large model (i.e., 110-a48). Fig. 4 (c) illustrates networks with dynamic regularization can narrow the gap
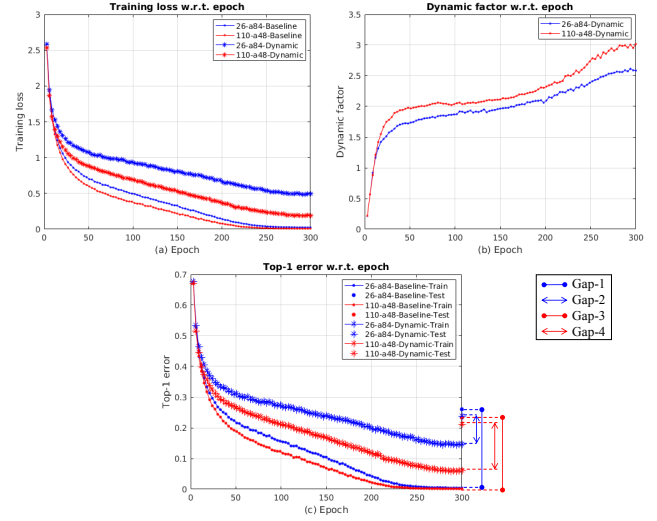

Fig. 4. The training loss, dynamic factor, and Top-1 error with respect to epoch. Gap stands for the difference between training and testing errors.

TABLE II
COMPARISON OF REGULARIZATION METHODS IN THE 3-BRANCH
ARCHITECTURE (I.E., RESNEXT). TOP-1 ERROR RATES (%) ARE SHOWN.

Network Architecture	Params	Regularization	Top-1 Error (%)
ResNeXt-26-2x32d	2.9M	Baseline [6]	22.95
		Shake-Shake [11]	21.45
		Dynamic (ours)	**20.91**
ResNeXt-26-2x64d	11.7M	Baseline [6]	20.59
		Shake-Shake [11]	19.19
		Dynamic (ours)	**18.76**

between the training and testing errors (from Gap-1 to Gap-2, and from Gap-3 to Gap-4) and achieve lower testing error when compared with Baseline.

C. Ablation Study and Discussion

1) Schedules of the regularization strength: Apart from the proposed dynamic schedule, the regularization strength can be adjusted by a linear-increment schedule as [24], where ScheduledDropPath is proposed to linearly increase the probability of dropped path (that can also be considered as the regularization strength) in training. Besides, the fixed regularization schedule is commonly used in many previous methods [20], [13], [11], [12]. We compared our dynamic method with such fixed or linear increment schedules. We used PyramidNet-26-a84 as a backbone to compare different regularization schedules.

Table III illustrates six different configurations of the regularization strength. 'Fix-x' means the dynamic factor is fixed to x and 'Linear-x' means the dynamic factor is linearly scheduled from 0 to x over the course of training steps. 'Fix-2' and 'Linear-3' achieve the best results in fixed and linear schedules, respectively. Compared with them, the dynamic setting with 23.83% error rate achieves the best performance, which shows the effectiveness of our dynamic regularization schedules.

2) Random noise: As mentioned in Section III, the range of the random noise involved in our dynamic regularization,

TABLE III
COMPARISON OF REGULARIZATION SCHEDULES.

PyramidNet-26-a84	Top-1 Error(%)	PyramidNet-26-a84	Top-1 Error(%)
Fix-1	25.45	Linear-1	25.76
Fix-2	24.75	Linear-2	25.09
Fix-3	25.52	Linear-3	24.28
Fix-4	30.52	Linear-4	25.80
		Dynamic	**23.83**

TABLE IV
EFFECTIVENESS OF LINEARLY GROWING R AND GAUSSIAN FILTERING.

PyramidNet-26-a84	Top-1 Error (%)
Baseline	26.30
Dynamic-Uniform R	25.28
Dynamic-Linear growth R	**23.83**
Dynamic-No filter	25.21
Dynamic-Gaussian filter	**23.83**

i.e., R, is designed to grow from bottom Res-Blocks to top Res-Blocks linearly. To evaluate this setting, we performed the dynamic regularization with uniform R and linearly growing R in PyramidNet-26-a84. From the third and fourth row of Table IV, we can see the model with uniform R is inferior to the model with linearly growing R inside Res-Blocks (25.83% v.s. 23.83%).

3) Gaussian Filtering: In the process of updating the dynamic factor, we employed a Gaussian filter to remove the instant change of the training loss in a mini-batch mode. That is, we refer to the Eq. (11) instead of Eq. (12) to update the dynamic factor. To study the effectiveness of Gaussian filter, we conducted comparative experiments between the Eq. (11) and Eq. (12). The last two rows of Table IV shows that if we remove the Gaussian filter, the error rate increases by 1.38%. This shows that the Gaussian filter also plays an important role in dynamic regularization.

V. CONCLUSION

In this paper, we have presented a dynamic schedule to adjust the regularization strength to fit various network architectures and the training process. Our dynamic regularization is self-adaptive in accordance with the change of the training loss. It produces a low regularization strength for light network architectures and high regularization strength for large ones. Furthermore, the strength is self-paced grown to avoid overfitting. Experimental results demonstrate that the proposed dynamic regularization outperforms state-of-the-art ShakeDrop and Shake-Shake regularization in the feature augmentation field. We consider that the dynamic regularization highly encourages to be exploited in data augmentation and Dropout-based methods in the future.

REFERENCES

[1] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems*, 2012, pp. 1097–1105.

[2] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 770–778.

[3] Zhong-Qiu Zhao, Peng Zheng, Shou-tao Xu, and Xindong Wu, "Object detection with deep learning: a review," *IEEE Transactions on Neural Networks and Learning Systems*, 2019.

[4] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger, "Densely connected convolutional networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 4700–4708.

[5] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He, "Aggregated residual transformations for deep neural networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 1492–1500.

[6] Dongyoon Han, Jiwhan Kim, and Junmo Kim, "Deep pyramidal residual networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 5927–5935.

[7] Sergey Ioffe and Christian Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International Conference on Machine Learning*, 2015, pp. 448–456.

[8] Shibani Santurkar, Dimitris Tsipras, Andrew Ilyas, and Aleksander Madry, "How does batch normalization help optimization?," in *Advances in Neural Information Processing Systems*, 2018, pp. 2483–2493.

[9] Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," *CoRR*, vol. abs/1207.0580, 2012.

[10] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[11] Xavier Gastaldi, "Shake-shake regularization," *CoRR*, vol. abs/1705.07485, 2017.

[12] Yoshihiro Yamada, Masakazu Iwamura, Takuya Akiba, and Koichi Kise, "Shakedrop regularization for deep residual learning," *CoRR*, vol. abs/1802.02375, 2018.

[13] Gao Huang, Yu Sun, Zhuang Liu, Daniel Sedra, and Kilian Q Weinberger, "Deep networks with stochastic depth," in *European Conference on Computer Vision*. Springer, 2016, pp. 646–661.

[14] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston, "Curriculum learning," in *Proceedings of the Annual International Conference on Machine Learning*. ACM, 2009, pp. 41–48.

[15] Karen Simonyan and Andrew Zisserman, "Very deep convolutional networks for large-scale image recognition," in *International Conference on Learning Representations*, 2014.

[16] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 1–9.

[17] Sergey Zagoruyko and Nikos Komodakis, "Wide residual networks," in *British Machine Vision Conference*, 2016.

[18] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alexander A Alemi, "Inception-v4, inception-resnet and the impact of residual connections on learning," in *AAAI Conference on Artificial Intelligence*, 2017.

[19] Terrance DeVries and Graham W Taylor, "Improved regularization of convolutional neural networks with cutout," *CoRR*, vol. abs/1708.04552, 2017.

[20] Gustav Larsson, Michael Maire, and Gregory Shakhnarovich, "Fractalnet: Ultra-deep neural networks without residuals," in *International Conference on Learning Representations*, 2017.

[21] Ian J Goodfellow, David Warde-Farley, Mehdi Mirza, Aaron Courville, and Yoshua Bengio, "Maxout networks," in *International Conference on Machine Learning*, 2013.

[22] Xu Shen, Xinmei Tian, Tongliang Liu, Fang Xu, and Dacheng Tao, "Continuous dropout," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 29, no. 9, pp. 3926–3937, 2017.

[23] M Pawan Kumar, Benjamin Packer, and Daphne Koller, "Self-paced learning for latent variable models," in *Advances in Neural Information Processing Systems*, 2010, pp. 1189–1197.

[24] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V Le, "Learning transferable architectures for scalable image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 8697–8710.

[25] Alex Krizhevsky, Geoffrey Hinton, et al., "Learning multiple layers of features from tiny images," Tech. Rep., Citeseer, 2009.