Addi-Reg: A Better Generalization-Optimization Tradeoff Regularization Method for Convolutional Neural Networks

Yao Lu, Zheng Zhang, *Senior Member, IEEE*, Guangming Lu, *Member, IEEE*, Yicong Zhou, *Senior Member, IEEE*, Jinxing Li, and David Zhang, *Life Fellow, IEEE*

***Abstract*—In convolutional neural networks (CNNs), generating noise for the intermediate feature is a hot research topic in improving generalization. The existing methods usually regularize the CNNs by producing multiplicative noise (regularization weights), called multiplicative regularization (Multi-Reg). However, Multi-Reg methods usually focus on improving generalization but fail to jointly consider optimization, leading to unstable learning with slow convergence. Moreover, Multi-Reg methods are not flexible enough since the regularization weights are generated from a definite manual-design distribution. Besides, most popular methods are not universal enough, because these methods are only designed for the residual networks. In this article, we, for the first time, experimentally and theoretically explore the nature of generating noise in the intermediate features for popular CNNs. We demonstrate that injecting noise in the feature space can be transformed to generating noise in the input space, and these methods regularize the networks in a Mini-batch in Mini-batch (MiM) sampling manner. Based on these observations, this article further discovers that generating multiplicative noise can easily degenerate the optimization due to its high dependence on the intermediate feature. Based on these studies, we propose a novel additional regularization (Addi-Reg) method, which can adaptively produce additional noise with low dependence on intermediate feature in CNNs by employing a series of mechanisms. Particularly, these well-designed mechanisms can stabilize the learning process in training, and our Addi-Reg method can pertinently learn the noise distributions for every layer in CNNs. Extensive experiments demonstrate that the proposed Addi-Reg method is more flexible and universal, and meanwhile achieves better generalization performance with faster convergence against the state-of-the-art Multi-Reg methods.**

***Index Terms*—Additional regularization (Addi-Reg), convolutional neural networks (CNNs), deep learning, multiplicative regularization (Multi-Reg), regularization.**

Manuscript received 26 December 2020; accepted 17 February 2021. Date of publication 22 March 2021; date of current version 16 September 2022. This work was supported in part by the National Key Research and Development Program of China under Project 2018AAA0100102; in part by the Shenzhen Fundamental Research Fund under Grant JCYJ20180306172023949; in part by the Guangdong Basic and Applied Basic Research Foundation under Grant 2019B1515120055 and Grant 2019A1515110475; in part by the Open Project Fund from Shenzhen Institute of Artificial Intelligence and Robotics for Society under Grant AC01202005018 and Grant AC01202005017; in part by the National Natural Science Foundation of China under Grant 62002085; in part by the Key Project of Shenzhen Municipal Technology Research under Grant JSGG20200103103401723; in part by the China Postdoctoral Science Foundation under Grant 2019TQ0316 and Grant 2019M662198; in part by the Medical Biometrics Perception and Analysis Engineering Laboratory, Shenzhen, China; in part by the Science and Technology Development Fund, Macau SAR under Grant 189/2017/A3; and in part by University of Macau under Grant MYRG2018-00136-FST. This article was recommended by Associate Editor B. Ribeiro. *(Corresponding authors: Zheng Zhang; Guangming Lu.)*

Yao Lu is with the Department of Computer Science and Technology, Harbin Institute of Technology, Shenzhen 518055, China (e-mail: yaolu@um.edu.mo).

Zheng Zhang, Guangming Lu, and Jinxing Li are with the Department of Computer Science and Technology, Harbin Institute of Technology, Shenzhen 518055, China (e-mail: darrenzz219@gmail.com; luguangm@hit.edu.cn; lijinxing158@gmail.com).

Yicong Zhou is with the Department of Computer and Information Science, University of Macau, Macau, China (e-mail: yicongzhou@um.edu.mo).

David Zhang is with the School of Data Science, Chinese University of Hong Kong, Shenzhen 518172, China, also with the Shenzhen Research Institute of Big Data, Chinese University of Hong Kong, Shenzhen 518172, China, and also with the Shenzhen Institute of Artificial Intelligence and Robotics for Society, Chinese University of Hong Kong, Shenzhen 518172, China (e-mail: davidzhang@cuhk.edu.cn).

This article has supplementary material provided by the authors and color versions of one or more figures available at https://doi.org/10.1109/TCYB.2021.3062881.

Digital Object Identifier 10.1109/TCYB.2021.3062881

I. Introduction

CONVOLUTIONAL neural networks (CNNs) have achieved tremendous breakthrough in the area of object recognition. Technically, the generalization capability is an important criteria to evaluate the performance of CNNs. However, generalization gap [1], a discrepancy of mean losses between the training and test datasets, has been recognized as a serious problem in CNNs. Notably, most of the current popular CNNs, such as [2]–[11], are usually overparameterized for the less abundant training datasets, which leads to the overfitting problem in networks. This is a main cause of generalization gap in CNNs.

To avoid overfitting, many regularization techniques have been proposed. e.g., the widely used L_1 norm and L_2 norm regularization in the loss function [1]; stochastic gradient descent (SGD) [12], [13]; batch normalization [3]; weight decay [14]; data augmentation [15]; and early stopping [16]. Especially, L_1 norm and L_2 norm can also be utilized in the traditional machine learning methods to improve the robustness of representations, such as [17]. There are also some compression methods of CNNs. Pruning methods [18], low-rank decomposition [19]–[22], and knowledge distillation [23] are introduced. These methods can also avoid the overfitting problem to some extent. To further improve generalization, researchers resort to generate multiplicative noise for the intermediate feature in CNNs, and these methods are called multiplicative regularization (Multi-Reg) methods. Different from traditional

methods, Multi-Reg produces regularization weights in feature space to directly regularize the hidden layers in CNNs. The typical Multi-Reg methods include Dropout [24], [25]; RandomDrop [26]; DropBlock [27]; Shake-Shake [28]; and ShakeDrop [29]. For example, Dropout randomly drops the neural nodes in each layer, which can be seen as generating regularization weights from Bernoulli distribution. RandomDrop is designed for ResNets [6] to drop layers with a linear decay probability. DropBlock selects several block areas in the feature maps according to the importance of the feature and drop them with a specific probability. RandomDrop and DropBlock can be seen as two special cases of Dropout. The effectiveness of the well-known Dropout will be weakened when it meets batch normalization [3], [9]. In contrast, Shake-Shake [28] can produce random regularization weights for multiple branch networks in Residual networks with identity-mapping [6], [7], [9], [10]. Since single branch networks are generally more efficient than multiple branch networks, ShakeDrop [29], a generalized version of Shake-Shake, was proposed for Residual networks. Due to the hierarchical structure of CNNs, regularizing CNNs actually is to regularize each layer, which implies the success of Multi-Reg is due to the direct regularization in hidden layers. Note that in real-world applications, the inherent noise in the training phase will corrupt the learned feature, which can be addressed by the robust machine-learning methods [30]–[34]. Therefore, the noise generated for hidden layers in Multi-Reg methods should require an elaborate search to produce suitable regularization noise rather than corrupt noise.

Although Multi-Reg methods are promising in generalization, there are three disadvantages to be considered.

1) They usually sacrifice the optimization. According to the relevant experiments in [28] and [29], the selection of the regularization weights has a great effect on the stability, leading to the slow convergence in training. For instance, Shake-Shake and ShakeDrop achieve a significant gain in accuracy with 1800 training epochs than those using 300 training epochs. However, in machine learning, the target of a generalization-optimization algorithm is to minimize the expected excess risk in a shortest training term. Based on [35], the expected excess risk of an algorithm can be upper bounded by the summation of the expected optimization and generalization errors, which confirms that generalization and optimization are both important. This suggests Multi-Reg is too weak to balance the generalization and optimization.
2) The flexibility of these existing Multi-Reg methods is limited, because the regularization weights are generated from manual-design distributions that rely on prior knowledge or preliminary searches by hand.
3) Although shake family regularization methods achieve outstanding performance in the Multi-Reg methods, they are not universal enough. For example, Shake-Shake and ShakeDrop methods can only be utilized in the residual networks, because the identity structure in this kind of residual network can alleviate the strong perturbations of the calculations caused by the weighing regularization.

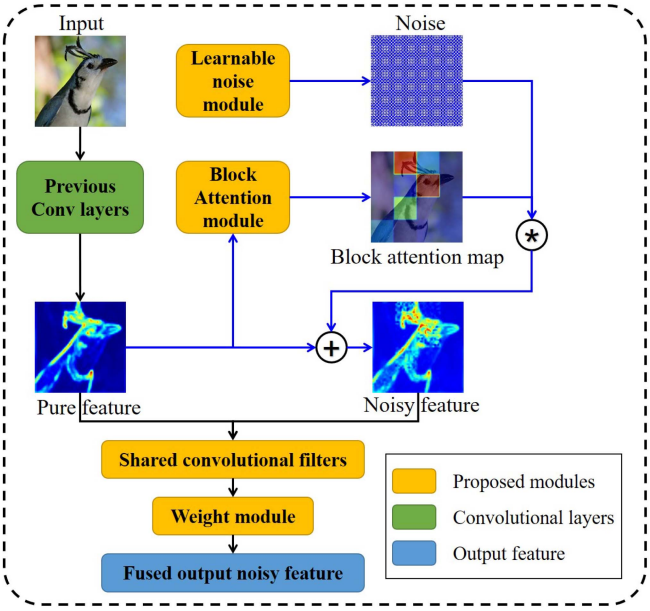


Fig. 1. Framework of the Addi-Reg method.

In this work, we conduct in-depth experimental and theoretical analyses to uncover the fundamental and functional mechanism of the existing Multi-Reg methods, and then a novel additional regularization (Addi-Reg) method is proposed to further improve the flexibility, university, and generalization capabilities of the Multi-Reg methods. Specifically, we study these methods from a more general aspect, including generating multiplicative noise and additional noise in the feature space. We give three crucially important observations to explore the underlying intrinsic nature of Multi-Reg methods. We give a first insight on the generated noise relationship between feature space and input space. As such, a new fundamental mechanism observation is discovered on generating noise in the feature space for CNNs. Based on the above two observations, we further offer an insightful observation that generating noise in feature space can lead to optimization decay in learning. Moreover, we reveal that the prior manual-design distribution of the weights in Multi-Reg methods is the main cause of the optimization decay.

To this end, we propose an Addi-Reg method. The framework of the Addi-Reg method is illustrated in Fig. 1. An Addi-Reg method utilizes learnable noise distributions to generate additional noise instead of multiplicative noise sampled from manual-designed noise distributions. Compared to the traditional Multi-Reg methods, the additional noise predicted by the Addi-Reg method is more flexible and can decrease the dependence on intermediate feature, leading to a flexible and stable regularization process. Furthermore, a block attention (BA) mechanism is employed to maintain the small change of features' distributions. Particularly, it shares filters in the pure and noisy feature, and utilizes a weight module to fuse the corresponding outputs. These two mechanisms can further stabilize calculations in learning. In this way, the

proposed Addi-Reg method can effectively address the deficiencies from the Multi-Reg methods. Our main contributions are summarized as follows.

1) This article explores and studies the fundamental mechanism of the regularization methods of generating the noise in the feature space, and further proves their optimization decay in learning. In the reasoning process, the main causes of the problems resulting from the Multi-Reg method have been determined.
2) Based on the understandings and causes of the problems, this article proposes a series of mechanisms and formulates the Addi-Reg method. Different from the Multi-Reg method, our Addi-Reg can adaptively generate more suitable additional noise from learnable noise distributions rather than multiplicative noise from fixed noise distributions.
3) Our method is applied to three state-of-the-art networks and evaluated on five benchmark databases. Compared to the popular Multi-Reg methods, the proposed Addi-Reg method is more flexible, universal, and obtains a better generalization-optimization tradeoff.

The remainder of this article is organized as follows. Section II introduces the related works of Multi-Reg methods. Section III explores the methods of generating noise in feature space from a more general aspect. Section IV will demonstrate the details of the proposed Addi-Reg method. Section V presents the experimental results on benchmark datasets and comparisons with Multi-Reg methods. Section VI concludes this article with future works.

II. Related Works

In this section, we will briefly introduce some related works on regularization methods of generating noise in feature space. Technically, the state-of-the-art regularization methods are usually Multi-Reg methods, which regularize the networks by producing multiplicative noise for hidden layers. There are five popular and typical methods, that is: 1) the well-known Dropout; 2) RandomDrop; 3) DropBlock; 4) Shake-Shake; and 5) ShakeDrop methods. Since Dropout and ShakeDrop are the general versions of drop family methods and Shake-Shake method, respectively, we only introduce Dropout and ShakeDrop in this section.

Dropout [24], [25] was proposed to turn off the neurons with a fixed possibility in the training process. The regularization process of Dropout can be seen as generating the regularization weights 0 or 1 from Bernoulli distribution for the intermediate feature in hidden layers. Suppose the input of the specific layer is denoted by x, and the convolutional transformation is f, the Dropout process is formulated as

$$y = \begin{cases} \omega_d f(x), & \text{in train}^{\text{fwd}} \\ \omega_d f(x), & \text{in train}^{\text{bwd}} \\ E(\omega_d) f(x), & \text{in test} \\ \omega_d \sim \text{Bernoulli} \end{cases} \tag{1}$$

where ω_d is the regularization weight tensor, which is randomly generated from Bernoulli distribution.

ShakeDrop exploits two random uniform variables to generate the regularization weights for each sample in the feature space. In addition, ShakeDrop uses a gate to alleviate the strong perturbations in training, which is inspired by RandomDrop. ShakeDrop is used in the networks with the identity-mapping structure. According to [29], the regularization process of ShakeDrop can be represented as

$$y = \begin{cases} x + (g + \omega_a - g\omega_a) f(x), & \text{in train}^{\text{fwd}} \\ x + (g + \omega_b - g\omega_b) f(x), & \text{in train}^{\text{bwd}} \\ x + E(g + \omega_a - g\omega_a) f(x), & \text{in test} \\ \omega_a, \omega_b \sim \text{Uniform} \\ g \sim \text{Bernoulli} \end{cases} \tag{2}$$

where ω_a and ω_b are from Uniform distribution. g is the gate from Bernoulli distribution. In (2), when $g = 0$, $f(x)$ will be, respectively, weighed by ω_a and ω_b in the forward and backward processes in training, and be weighed by the expectation of ω_a in the test process; otherwise, it will remain constant.

It should be noted that our Addi-Reg method is totally different from the existing Multi-Reg methods from the following three perspectives.

1) Addi-Reg method generates additional noise rather than multiplicative noise in the feature space, which can decrease the noise dependence on the sample feature, leading to more stable learning in training.
2) Addi-Reg method can learn the noise distributions for each layer in CNNs through the learning mechanism, resulting in improvements of flexibility and pertinence in the regularization process.
3) Addi-Reg method can be used in the residual networks as well as the other networks with different structures due to a series of stable learning mechanisms, yielding prominent universality.

III. Explorations of Generating Noise in Feature Space

In this section, we will explore the fundamental mechanisms of Multi-Reg methods by experimentally analyzing the nature of generating noise in the feature space for CNNs. Then, three observations are uncovered and demonstrated to prove the optimization decay in this kind of regularization method.

A. Observation on the Generated Noise Relationship Between Feature Space and Input Space

First, it is well known that the encoder–decoder methods [36] are widely used in many computer vision tasks. For example, the encoder can transform images to an embedded feature, and the decoder can project the embedded feature to images or other data in a specific domain. Furthermore, the regularization methods with generating noise in input space and feature space both can produce regularization effects to the networks. According to these two factors, we intuitively hypothesize that the generating noise for the intermediate feature can be seen as augmenting training images. That is to say, there may exist augmented training images whose regularization effect is equivalent to augmenting data in the feature space. Therefore, the key problem is to find the augmented training images.

Suppose n is the number of layers of networks, $f^{(i)}(\cdot; \theta^i)$, $\delta^{(i)}(\cdot)$, and $\varepsilon^{(i)}(\cdot)$, respectively, represent transformation function, activation function (ReLU), and noisy function at the ith

layer in networks, where θ^i is the parameters at the ith layer. Given the input image x, the output of the ith layer under the regularization with generating noise in feature space can be formulated as follows:

$$\begin{aligned}\tilde{F}^{(i)}\left(x;\theta^{1\sim i}\right) &= \varepsilon^{(i)}\left(\delta^{(i)}\left(f^{(i)}\left(F^{(i-1)}\left(x;\theta^{1\sim(i-1)}\right);\theta^i\right)\right)\right)\\ &= \left(\varepsilon^{(i)}\circ\delta^{(i)}\circ f^{(i)}\right)\left(F^{(i-1)}\left(x;\theta^{1\sim(i-1)}\right);\theta^i\right)\end{aligned} \tag{3}$$

and for the augmented image x', the output of the ith layer in the same networks is

$$\begin{aligned}F^{(i)}\left(x';\theta^{1\sim i}\right) &= \delta^{(i)}\left(f^{(i)}\left(F^{(i-1)}\left(x';\theta^{1\sim(i-1)}\right);\theta^i\right)\right)\\ &= \left(\delta^{(i)}\circ f^{(i)}\right)\left(F^{(i-1)}\left(x';\theta^{1\sim(i-1)}\right);\theta^i\right).\end{aligned} \tag{4}$$

Thus, we can obtain the augmented image x' through minimizing the following error function:

$$E\left(x,x'\right) = \left\|\tilde{F}^{(n)}(x;\Theta) - F^{(n)}\left(x';\Theta\right)\right\|^2 \tag{5}$$

where Θ indicates the parameters from all the layers in networks. Since the depth of networks is usually very large, it is difficult to directly minimize (5) because of the vanishing gradients. Therefore, the error function can be minimized by utilizing outputs from all the layers. The error function can be reformulated as following:

$$\begin{aligned}&E\left(x,x^{(1)'},x^{(2)'},\ldots,x^{(n)'}\right)\\ &\quad= \sum_{i}^{n}\lambda_i\left\|\tilde{F}^{(i)}\left(x;\theta^{1\sim i}\right) - F^{(i)}\left(x^{(i)'};\theta^{1\sim i}\right)\right\|^2\end{aligned} \tag{6}$$

where λ_i is the weight for the ith layer's error. In the ideal case, we can obtain the only one optimal augmented image with $x' = x^{(1)'} = x^{(2)'} = \cdots = x^{(n)'}$. However, it is hard to achieve this goal through our proof in the Appendix in the supplementary material. This implies that it is impossible to find the optimal augmented images based on the minimization of (5). However, we can resolve this problem through the following two strategies.

First, the decoder–encoder networks [36] can project source data into embedded feature by the encoder networks and the reconstruct data into the specific domain by decoder networks. Therefore, we can apply the encoder–decoder to address our problem. Specifically, the encoder is used to project raw images with intermediate noise, whose output of every layer is the same as (3). Then, the decoder is used to reconstruct augmented images from the embedded feature produced from the encoder. In order to train this encoder–decoder effectively, the decoder has the same depth to that of the encoder. Thus, the supervised information from every pair of encoder–decoder layers can be used in training. The relevant output of the decoder corresponding to the ith layer of the encoder is formulated as follows:

$$\begin{aligned}&\tilde{F}_*^{(i)}\left(\tilde{F}^{(n)};\theta_*^{n\sim(n-i+1)}\right)\\ &\quad= \left(\delta_*^{(i)}\circ f_*^{(i)}\right)\left(\tilde{F}_*^{(i+1)}\left(\tilde{F}^{(n)}(x,\Theta);\theta_*^{n\sim(n-i)}\right);\theta_*^{i+1}\right)\end{aligned} \tag{7}$$

where θ_*^i represents the parameters at the ith layer in the decoder.

Second, since deep CNNs usually attach nonliner layers at the end of every liner layer, the nonlinear layers can throw away information [37], resulting in noise loss and further leading to regularization decay of the augmented input data. This indicates that although the augmented images are obtained, the final performance achieved by networks cannot be the same as that from the augmentation in feature space with the same parameters Θ in networks [see (5)]. Therefore, considering the practical requirement, the restriction of two networks with the same parameters is relaxed to the same structure in (5). This relaxed restriction implies that the networks can learn parameters for their own augmented input images to keep performance the same as possible as that from the augmentation in feature space.

Based on the above two strategies, the error function can be formulated from two parts. The first part is the error between the noisy output at the ith $(i = 1, 2, \ldots, n-1)$ layer of the encoder and the corresponding reconstructed feature from the decoder. The second part is the final outputs' error between the encoder and the networks for the reconstructed augmented images with the same structure as that of the encoder. Therefore, the final error function is shown as

$$\begin{aligned}E(x,x') &= \sum_{i=1}^{n-1}\lambda_i\left\|\tilde{F}^{(i)}\left(x;\theta^{1\sim i}\right) - \tilde{F}_*^{(i)}\left(z;\theta_*^{n\sim(n-i+1)}\right)\right\|^2\\ &\quad+ \lambda_n\left\|\tilde{F}^{(n)}(x;\Theta) - F^{(n)}\left(x';\Theta'\right)\right\|^2\\ &\text{where}\begin{cases} z = \tilde{F}^{(n)}(x;\Theta)\ \text{(the embedded vector)}\\ x' = \tilde{F}_*^{(n)}(z;\Theta_*)\ \text{(the reconstructed images)}.\end{cases}\end{aligned} \tag{8}$$

Through minimizing (8), we can obtain the final augmented images, whose regularization effects are almost the same as that of the generating noise in feature space. According to the above analyses, the following observation can be obtained.

Observation 1: Generating the noise in the feature space can be seen as generating the noise in the input space.

The detailed experiments are illustrated as follows. Initially, the encoder networks $\tilde{F}$ include three convolutional layers (conv) and one fully connected layer (fully connected)

$$\begin{pmatrix}\text{conv } (2\times 2,\ \ 128)\\ \text{conv } (2\times 2,\ \ 256)\\ \text{conv } (2\times 2,\ \ 512)\\ \text{fully connected } (512\times 16,\ \ 10)\end{pmatrix}.$$

The decoder networks $\tilde{F}_*$ are constructed with the deconvolutional layers to generate the reconstructed feature with the same resolution to the corresponding convolutional layers in the encoder networks. The networks F for reconstructed augmented images are constructed with the same structure to encoder networks. We conduct experiments on the well-known MNIST [38] dataset. Particular, the encoder networks are trained through, respectively, injecting multiplicative noise and additional noise in the hidden layers. The noise distributions are adjusted and determined to ensure approximately the same classification performances of these two networks, obtaining accuracies with 93.86% and 93.92%. At the next step, the decoders are employed to inversely project the noisy

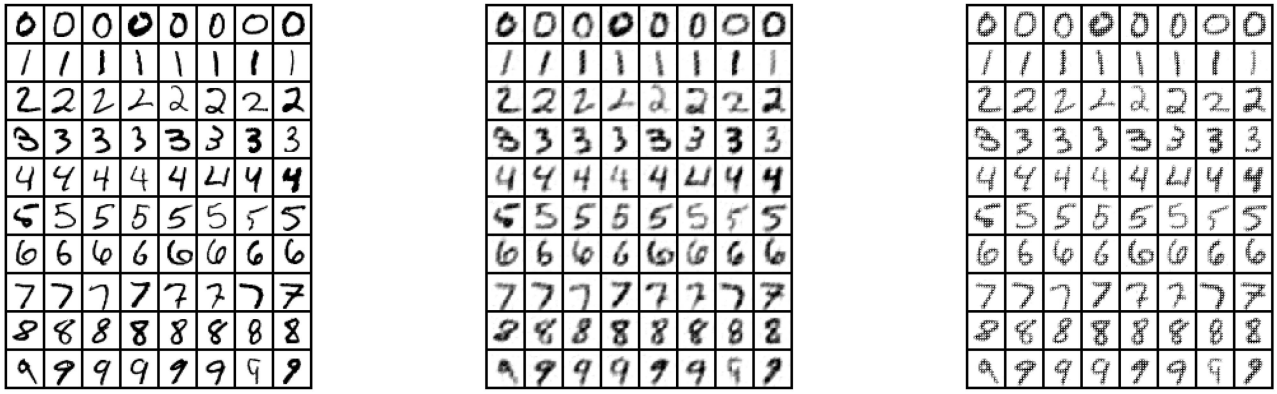


Fig. 2. Visualizations of the original input images and the reconstructed images, respectively, from multiplicative noise and additional noise under approximately the same accuracy on the MNIST dataset. (a) Original input images. (b) Reconstructed images from multiplicative noise. (c) Reconstructed images from additional noise.

intermediate feature to the input space and obtain the final reconstructed noisy images. These reconstructed augmented images are fed to the networks F. Then, the reconstructed augmented images are obtained through training $\tilde{F}$, $\tilde{F}_*$, and F together by minimizing the error according to (8).

The sampled reconstructed training images from multiplicative and additional noise are visualized in Fig. 2. Eventually, we use the noisy training images reconstructed from multiplicative noise and additional noise to compute the final classification performance. The accuracies achieved on these two sets of the reconstructed noisy images are 93.79% and 93.86%, respectively. This demonstrates that the regularization effects from reconstructed noisy images are almost the same as that from generating the noise for hidden layers. This observation indicates there may exist an augmented noisy dataset, whose regularization effect is equivalent to that of generating the noise in feature space.

B. Observation on the Fundamental Mechanism of Generating Noise in the Feature Space

Based on Observation 1, we can further obtain the following.

Observation 2: Generating noise in the feature space regularize the networks by a Mini-Batch in Mini-Batch (MiM) sampling manner.

Since in each epoch of the training process, the noise is injected to each hidden layer, this process can be seen as implicitly augmenting the training images once according to the previous observation. The main difficulty in the reasoning process of this kind of method is that the training data are dynamically augmented in each epoch. However, we can analyze the augmenting procedure in the reverse process. By this way, the training process will be very easy to understand. Suppose the training set is $\mathcal{D} = \{x_1, x_2, \ldots, x_N\}$ with N samples, the reconstructed noise in the kth epoch is ε_k and the number of training epochs is K. Since $\mathcal{D}$ is augmented with ε, then all possible augment samples from all the epochs can form a union, which can be regarded as a super augmented dataset denoted by $\mathcal{D}^+$. In this way, we have

$$\mathcal{D}^+ = \bigcup_{k=1}^{K} \mathcal{D}_{\varepsilon_k}. \tag{9}$$

It is clear that in each epoch, a batch of training samples $\mathcal{D}_{\varepsilon_k}$ is selected from $\mathcal{D}^+$ to train the networks. This mechanism is similar to the mini-batch SGD (MSGD) [39]. The difference is that, in different epochs, there may exist same training samples, while not with MSGD. This is because in practice, MSGD has been used in every epoch because of its regularization effect. In the training process of these methods of generating noise in feature space, the augmented dataset has been sampled twice with different levels, which can be named with MiM.

In MiM, if the noise is generated from a discrete distribution, the number of the samples in the super augmented dataset is finite, otherwise, it is infinite. Since the regularization weights of Shake-Shake and ShakeDrop are generated from continuous distributions, their super augmented datasets are infinite. This implies the networks need much more epochs to cover all the possible samples in training, leading to a slow convergence. The experimental results in [29] and Table VI indeed validate these methods have a significant improvement in performance when the models are trained for a long term.

C. Observation on the Optimization of Generating Noise in the Feature Space

Through the previous two observations and the theory of expected excess risk, we can prove the following observation.

Observation 3: Generating noise in the feature space can cause an optimization decay in learning.

Proof: The expected excess risk can be upper bounded by the expected generalization error and optimization error [35]. $\mathcal{G}$ is the unseen data with unknown joint probability distribution. The training set $\mathcal{D} = \{x_1, x_2, \ldots, x_N\}$ with N samples drawn independent identically distributed from $\mathcal{G}$. The goal is to find output f in the function family $\mathcal{F}$ to minimize the loss function $\ell(f, x)$. Then, the expected risk and empirical risk are

$$L_{\mathcal{G}}(f) := \mathbb{E}_{x\sim\mathcal{G}}\big[\ell(f, x)\big];\ L_{\mathcal{D}}(f) := \frac{1}{N}\sum_{i=1}^{N} \ell(f, x_i). \tag{10}$$

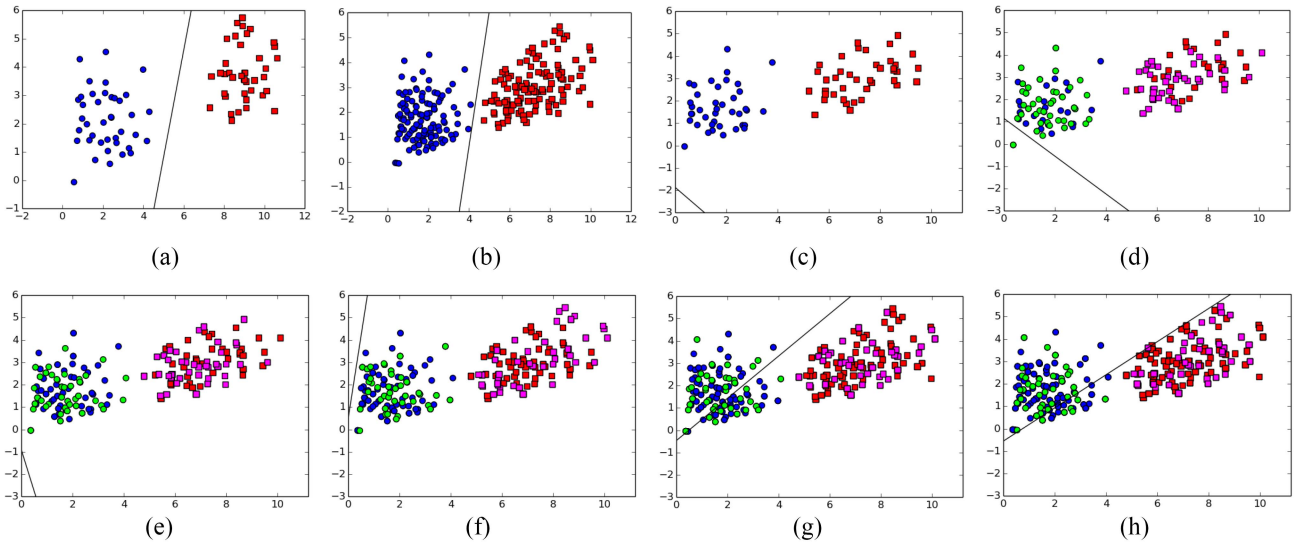


Fig. 3. Training processes from the original data, augment data and MiM. (a) and (b) are the original data and the augment data, respectively. The *green* and *purple* points represent the samples selected in the current epoch. (a) Original data. (b) Augment data. (c) Epoch 1. (d) Epoch 2. (e) Epoch 3. (f) Epoch 4. (g) Epoch 5. (h) Epoch 6.

If $\hat{f}$ is the output of $L_{\mathcal{D}}(f)$ and $f^* = \inf_{f\in\mathcal{F}} \ell_{\mathcal{D}}(f)$. The expected excess risk κ can be upper bounded

$$\kappa \leq \mathbb{E}_{\mathcal{D}}\left[\underbrace{L_{\mathcal{G}}\left(\hat{f}\right) - L_{\mathcal{D}}\left(\hat{f}\right)}_{\epsilon_{\text{gen}}}\right] + \mathbb{E}_{\mathcal{D}}\left[\underbrace{L_{\mathcal{D}}\left(\hat{f}\right) - L_{\mathcal{D}}\left(f^*\right)}_{\epsilon_{\text{opt}}}\right] \quad (11)$$

where $\mathbb{E}_{\mathcal{D}}[\Delta_{\text{gen}}]$ is the expected generalization error and $\mathbb{E}_{\mathcal{D}}[\Delta_{\text{opt}}]$ is the expected optimization error.

Based on (1), (2), and Observations 1 and 2, in the method of generating noise in feature space, $\mathcal{D}$ is altered to $\mathcal{D}^+$. Thus, the optimization error ϵ^g_{opt} of the generating noise methods at the t th epoch are formulated as follows:

$$\epsilon^g_{\text{opt}} = L_{\bigcup_{k=1}^{t}\mathcal{D}_{\varepsilon_k}}\left(\hat{f}^g_t\right) - L_{\mathcal{D}}\left(f^*_t\right). \quad (12)$$

In the MSGD method, the training images in the kth epoch can be seen as noised by the produced multiplicative noise 0 or 1, indicated by ε^s_k, then, the optimization error ϵ^s_{opt} of MSGD is

$$\epsilon^s_{\text{opt}} = L_{\bigcup_{k=1}^{t}\mathcal{D}_{\varepsilon^s_k}}\left(\hat{f}^s_t\right) - L_{\mathcal{D}}\left(f^*_t\right). \quad (13)$$

According to the corresponding theory of MSGD [40]–[42], MSGD can produce inherent noise to the gradient and enlarge the gradient variance, resulting in slower convergence asymptotically and optimization decay. From (11) and (13), the discrepancy between ϵ_{opt} and ϵ^s_{opt} of the MSGD method is

$$\Delta^s_{\text{opt}} = \epsilon^s_{\text{opt}} - \epsilon_{\text{opt}} = L_{\bigcup_{k=1}^{t}\mathcal{D}_{\varepsilon^s_k}}(\hat{f}^s_t) - L_{\mathcal{D}}(\hat{f}_t) < 0. \quad (14)$$

Based on Observation 2, this kind of generating noise in feature space methods are the enhanced versions of MSGD. Therefore, according to (14), we can obtain that the discrepancy between ϵ^g_{opt} of the method of generating noise in the feature space and ϵ_{opt} at the t th epoch is

$$\Delta^g_{\text{opt}} = \epsilon^g_{\text{opt}} - \epsilon_{\text{gen}} = L_{\bigcup_{k=1}^{t}\mathcal{D}_{\varepsilon_k}}\left(\hat{f}^g_t\right) - L_{\mathcal{D}}\left(\hat{f}_t\right) < 0. \quad (15)$$

■

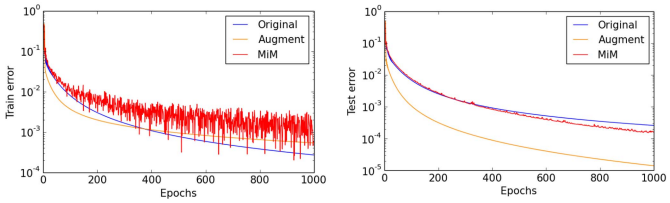
In this way, we can demonstrate our observation. This is the end of proof. In order to explicitly observe that the MiM can cause an unstable learning in the training process, we simulate the process of MiM training on a simple logistic regression,[1] which is shown in Fig. 3. Fig. 3(a) and (b) are the original data and the augmented data generated by the multiplicative noise (regularization weights) ω from a discrete uniform distribution $\omega \in \{0.7, 0.8, 0.9\}$. Fig. 3(c)–(h) sequentially illustrates the results of each training epoch. It is clear that, the output functions have a big difference in successive epochs, which indicates that the learning is unstable. Furthermore, the training errors and test errors are also compared in Fig. 4. From the results, we can observe that MiM is unstable, converges slowly and produces high training error. Notably, it can achieve better performance than the original datasets in the test. This suggests although MiM can achieve better generalization, it changes the distributions of feature greatly in training, resulting in the unstable learning.

IV. Proposed Regularization Method

A. Motivations

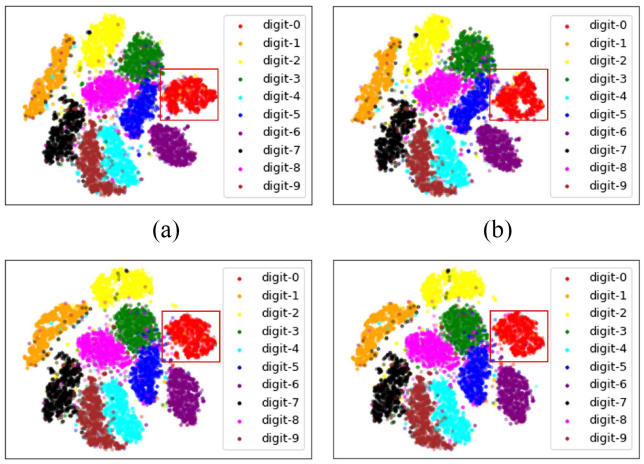
Inspired by the observations discovered in Section III, the noise generated in the feature space is the prominent factor of the optimization decay in the training process. In order to explicitly visualize the effects on the intermediate feature's distributions from multiplicative noise and additional noise, we perform two networks, respectively, injected with multiplicative noise and additional noise on the MNIST datasets under the same classification performance and visualize the pure feature and noised features of the hidden layers. Fig. 5 shows the final results. From this figure, compared to additional noise, multiplicative noise has a greater change to

[1] In CNNs, the information will be dissolved by pooling and ReLU layers [37]. Therefore, we use a simple logistic regression (a fully connected layer and a Sigmoid layer) to briefly illustrate the MiM process.



(a) (b)

Fig. 4. Comparisons of (a) train error and (b) test error.



(a) (b)

(c) (d)

Fig. 5. Comparisons of effects on intermediate feature's distributions from multiplicative noise and additional noise under the same accuracy on the MNIST dataset. (a) Pure feature in Multi-Reg. (b) Multiplicative noisy feature. (c) Pure feature in Addi-Reg. (d) Additional noisy feature.

the original feature's distribution, especially for the distribution of digit zero marked by red box. This will cause a larger calculation perturbation in learning and lead to slower convergence than the additional noise. This implies that the way of scaling the feature may contribute to a great change of the distributions between the original data and the noised data.

The goal of this article is to alleviate the optimization decay and inflexibility caused by the Multi-Reg methods. According to the previous researches, since the noised feature in the hidden layers from the Multi-Reg method are produced based on multiplication of the weights and feature. Therefore, the multiplicative noised feature have high dependence on the original feature, leading to the changes of the intermediate feature and slow convergence. To this end, we will address these challenging problems from the following aspects.

1) Initially, the key factor to resolve the problem is to decrease the noised feature's dependence on the original feature. This can also improve the universality of the regularization method and can be utilized in the networks with different structures, because it will not need the identity structure to stabilize the learning any more.
2) Furthermore, in Multi-Reg methods, the regularization weights distributions are determined through prior manual searching, leading to inflexibility, heavy time costs, and labor costs of the methods. Therefore, it is critical to improve the flexibility of the regularization method.

Algorithm 1 Addi-Reg Algorithm

Initialize: parameters of convolutional layer θ;
$\theta^* = \{\mu, \sigma, \alpha\}$ in $N(\mu, \sigma^2)$ and $U(-\alpha, \alpha)$
Repeat:
  Convolutional layer input: x
  Generate noise: $\varepsilon = \varepsilon_g + \varepsilon_u$
  Generate Block Attention noise: $\varepsilon_b = f_{BA}(x; \varepsilon)$
  Share filter: $x' = f(\theta; x)$; $x'_\varepsilon = f(\theta; x + \varepsilon_b)$
  Predict weight: $\omega = f_\omega(x'; x'_\varepsilon)$
  Fuse feature: $y = \omega x' + (1 - \omega) x'_\varepsilon$
  Compute loss and update θ and θ^*
Until convergence

3) Finally, since CNNs are hierarchical in structure, with the depth increasing, the concepts of the feature learned in different layers are also different. Employing the same noise distribution to produce regularization weights for different layers are not adaptive, pertinent and suitable enough, which may also handicap the regularization. Thus, improving the pertinence of the regularized noise is beneficial to the generalization.

B. Overview of the Proposed Addi-Reg method

To consummate the deficiencies demonstrated in Section IV-A, this article proposes the additional noise regularization method (Addi-Reg). The Addi-Reg method includes the following mechanisms.

1) Generating additional noise instead of scaling the feature in our method. The additional noise is produced from the learnable noise distributions, which can improve the flexibility of the method and produce more individualized noise for various layers. Furthermore, since the additional noise generated in such learnable way is independent of intermediate features, a more appropriate noise can be learned to stabilize the learning in the training process.
2) We propose the block attention (BA) mechanism to blur the feature maps only in the required local areas rather than blurring the global feature map. This can limit the additional noise in feature maps, which further maintains the distribution of the augment data.
3) From the filter's aspect, we propose sharing the filters on the pure data and noisy data. From the feature's aspect, a predicting weight module is utilized to produce weights for the convolutional outputs from pure data and noisy data to fuse them together. These two approaches can promote the stability of the learning process and decrease the dependence on the expectation noise to some extent in the test. In our experiments, our method can still work when the additional noise is cut off in the test, while Multi-Reg is failed.

These mechanisms formulate the proposed Addi-Reg method, shown in Algorithm 1. We will explicitly demonstrate these steps as follows.

C. Generate Suitable Additional Noise

Since the generated noise should meet the requirement of insensitiveness to the input sample, the additional noise can be simply produced from the Gaussian distribution. However, only adding Gaussian noise may decrease the distances of the samples in the feature space. This will also degenerate the performance of classification. In order to preserve the differences among samples, the Gaussian noise is supplemented with uniform noise. Therefore, the additional noise ε is formulated as follows:

$$\varepsilon = \varepsilon_g + \varepsilon_u; \text{ where } \begin{cases} \varepsilon_g \sim N(\mu, \sigma^2) \\ \varepsilon_u \sim U(-\alpha, \alpha) \end{cases} \tag{16}$$

where $N(\mu, \sigma^2)$ and $U(-\alpha, \alpha)$ denote the normal distribution and uniform distribution, respectively. In particular, different from the Multi-Reg methods, these two noise distributions can be learned in training, that is, μ, σ, and α are learnable. Therefore, such learning strategy can improve the method's flexibility and produce suitable noise for each layer.

The forward learning process includes two steps: 1) predicting parameters μ, σ, and α and 2) sampling noise from these learned noise distributions. However, since the sampling operation is not differentiable, this operation cannot generate gradients for the parameters of noise distributions in the back propagation pass. Therefore, we employ reparameterization trick [43] utilized in VAEs to address this problem. Specifically, we first sample Gaussian noise ε'_g and uniform noise ε'_u from $N(0, 1)$ and $U(-1, 1)$, respectively. Then, the parameters μ, σ, and α are utilized to transform the noise ε'_g and ε'_u. Therefore, the final additional noise ε can be reformulated as follows:

$$\begin{aligned} &\varepsilon = \varepsilon_g + \varepsilon_u \\ &\text{where } \begin{cases} \varepsilon_g = \sigma\left(\varepsilon'_g + \mu\right) & \left(\varepsilon'_g \sigma \sim N(0, 1)\right) \\ \varepsilon_u = \alpha \varepsilon'_u & \left(\varepsilon'_u \sim U(-1, 1)\right). \end{cases} \end{aligned} \tag{17}$$

In this way, the parameters from noise distributions can be learned and updated in the training process.

D. Block Attention Module

To limit the additional noise added to the feature, we can inject the noise only in one subarea of the feature. Generally, the activated area of the intermediate feature map plays a key role in the final classification for CNNs. Therefore, we can only blur the activated area to make feature difficult from the filters to learn the patterns, which can effectively avoid overfitting and shrink the additional noise. The attention mechanism is widely used in many tasks to catch the interesting area. However, in our method, directly employing the attention scheme does not work well because its location is too precise, which greatly destroys the activated area and leads to performance loss [see the red dashed box painted in the attention noise (AN) feature in Fig. 6]. Thus, we propose a BA mechanism f_{BA} to provide block location, which can blur both the activated area and background. Fig. 6 shows the comparisons of block noise (BN), AN and BA noise (BAN). Apparently, the BAN can not only preserve the general activated area, but also record the lower activated areas to decrease the noise added. To this end, the HardReLU function δ, a special case of ReLU, is developed to formulate the attention mechanism. It is shown as

$$\delta(x) = \begin{cases} \min(x, 1), & x \geq 0 \\ 0, & \text{otherwise.} \end{cases} \tag{18}$$

In this way, δ locates the activated area and weighs the additional noise when the activated values are in the range of 0 and 1.

Next, the block effect on the location area is implemented by the average pooling (Downsamp) and nearest interpolation (Upsamp) operations, shown in Fig. 6. As such, the BAN ε_b is obtained by

$$\begin{cases} A' = \delta\ (\text{ Downsamp } (x, s\ )\ ) \\ A = \text{Upsamp}\ \left(\ A', s\ \right) \\ \varepsilon_b = A \cdot \varepsilon \end{cases} \tag{19}$$

where s is the scale of downsample and upsample. A' and A are the downsample BA map and BA map, respectively.

E. Share Filters and Weigh the Outputs

To better alleviate the perturbations in calculations, the pure input and noisy input will share the same convolutions to make the filters be adaptive to the noisy data. If the parameters of the shared filters are indicated by θ, the outputs of the pure data and noisy data through the same filters are

$$x' = f(\theta; x); \quad x'_\varepsilon = f(\theta; x + \varepsilon_b). \tag{20}$$

Moreover, a learnable predicting weight module is employed to predict weights for x' and x'_ε to accurately control the strong perturbations from the noisy output. In this module, x' and x'_ε are first concatenated along the channel axis. Inspired by squeeze-and-excitation networks (SE-Nets) [44], two "1×1" convolutional layers ($f_{1\times1}$) and one ReLU activation layer between them are aggregated together followed by a Sigmoid layer. If a global average pooling (G-AvgPool) layer is installed before the first convolutional layer, the module produces the weights at channel level, otherwise, at pixel level. The weight module f_ω is defined as

$$\begin{cases} x_C = \text{Concat}\ \left(\ x', x'_\varepsilon\ \right) \\ x_C = \text{G-AvgPool}\ (x_C)\ \text{(if channel level)} \\ x'_C = \text{ReLU}\ \left(\ f_{1\times1}\left(\ x_C, 2c, \frac{2c}{a}\ \right)\ \right) \\ \omega = \text{Sigmoid}\ \left(\ f_{1\times1}\left(\ x'_C, \frac{2c}{a}, c\ \right)\ \right) \end{cases} \tag{21}$$

where $f_{1\times1}$ denotes the "pointwise" convolutions with spatial kernel size "1×1," and c represents the channels of x' and x'_ε. In $f_{1\times1}$, $2c$ and $(2c/a)$ denote the number of input channels and output channels, respectively. a is utilized to regularize the parameters of the convolutional filters. Finally, the fused output of pure data and noisy data is formulated as

$$y = \omega x' + (1 - \omega) x'_\varepsilon. \tag{22}$$

The full algorithm of our Addi-Reg method is outlined in Algorithm 1.

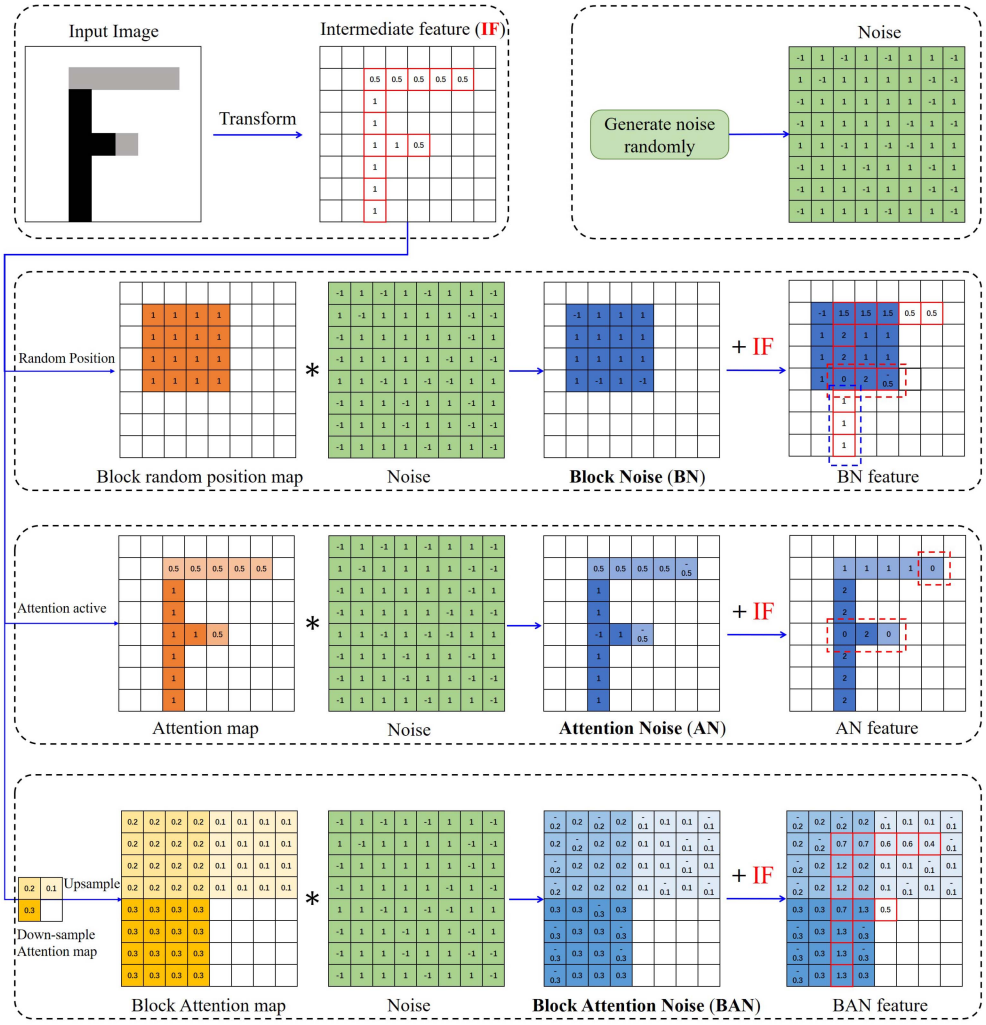


Fig. 6. Comparisons of BN, AN, and BAN. The values in blank grids are zero.

V. Experimental Results and Analysis

In this section, we will evaluate the effectiveness of the proposed Addi-Reg method on different datasets by comparing state-of-the-art Multi-Reg models. Generally, the overfitting occurring in the networks is mainly caused by two factors: 1) small number of training images and 2) inadequate spatial information of the training images [45]. In order to effectively verify the ability of resolving the overfitting problem of the proposed Addi-Reg regularization method, the datasets easily resulting in an overfitting in networks are selected according to the two factors demonstrated above.

Following the common practices [19], [21], [22], [46]–[50], the proposed Addi-Reg methods are evaluated on five datasets: 1) CIFAR-10; 2) CIFAR-100 [51]; 3) low resolution ImageNet-32 × 32; 4) low resolution ImageNet-16 × 16 [45]; and 5) regular ImageNet datasets [52]. CIFAR-10 and CIFAR-100 consist of a small number of training images. The low resolution ImageNet-32 × 32 and ImageNet-16 × 16 are downsampled from regular ImageNet with same number of samples and categories. Thus, their spatial information is much less abundant, which is more difficult than the regular ImageNet dataset [45]. These four datasets can easily cause an overfitting for deep CNN models. Since the regular ImageNet dataset is widely utilized in many research areas, it also been employed in this article to validate the performances of the regularization methods.

A. Datasets

CIFAR datasets include CIFAR-10 [51] and CIFAR-100 [53]. They both have 60 000 colored nature scene images in total and the images' size is "32 × 32." There are 50 000 images for training and 10 000 images for testing in 10 and 100 classes. Augmentations of these two datasets are the same to [54] and [55].

Regular ImageNet dataset [52] is used for ILSVRC, which contains more than one million training images and 50 000 validation images in 1000 categories. In training, all the training images are randomly resize cropped with size "224 × 224," while the validation images are obtained by center cropped from size "256 × 256." The other augmentations of this dataset are the same to [6] and [26].

Low Resolution ImageNet datasets [45] are the downsampled variants of regular ImageNet [52], and contain the same number of classes and images of regular ImageNet dataset.

TABLE I
TEST ERROR RATE ON IMAGENET-32 × 32 AND IMAGENET-16 × 16 DATASETS. THE BASIC WIDTH OF WIDERESNETS IS 16. k REPRESENTS THE TIMES OF THE BASIC WIDTH. THE DEPTHS OF WIDERESNET IS 28. THE RESULTS MARKED BY ⋆ ARE FROM [45]. AND THE OTHER ERROR RATES ARE PRODUCED BY OUR IMPLEMENTATIONS

Methods	ImageNet-32 × 32			ImageNet-16 × 16		
	Params (M)	Top-1 err. (%)	Top-5 err. (%)	Params (M)	Top-1 err. (%)	Top-5 err. (%)
WideResNet ($k = 3$) (baseline)	3.5	51.06	26.08	0.9	74.13	50.80
WideResNet ($k = 3$) + Addi-Reg (**ours**)	3.6	**47.72**	**23.44**	0.9	**69.59**	**45.84**
WideResNet ($k = 5$) (baseline) [45]	9.5	45.46⋆	21.36⋆	2.3	66.60⋆	41.59⋆
WideResNet ($k = 5$) + Addi-Reg (**ours**)	9.7	**43.06**	**19.64**	2.4	**63.30**	**38.50**
WideResNet ($k = 10$) (baseline) [45]	37.1	40.96⋆	18.87⋆	8.9	59.94⋆	35.10⋆
WideResNet ($k = 10$) + Addi-Reg (**ours**)	37.9	**39.34**	**16.79**	9.1	**56.78**	**32.25**

There are three versions in total: 1) ImageNet-64 × 64; 2) ImageNet-32 × 32; and 3) ImageNet-16 × 16, which indicates the downsampled images' sizes are "64 × 64," "32 × 32," and "16 × 16," respectively. In this article, the experiments are performed on the ImageNet-32 × 32 and ImageNet-16 × 16 datasets. The augmentations of these two datasets are the same as [45].

B. Baseline Models and Initializations

We validate the efficacy of our model by comparing the current state-of-the-art networks, including ResNet [], WideResNet [9], DenseNet [11], and PyramidNet [10], which are more efficient than the multiple branch networks. The Addi-Reg regularization is applied to "3 × 3" convolutional layers. In WideResNet and DenseNet, it is applied to the second layer in each module, while in ResNet and PyramidNet, it is used in the middle layer. In all the networks, the noise distributions are initialized with $N(0, 1)$ and $U(-1, 1)$. In the BA mechanism, the size of the downsampled attention map is set to "4 × 4" in each module. In the predicting weight module, the number of output channels in the first "1 × 1" convolutions is decreased by 8 times [$a = 8$ in (21)]. Furthermore, the weights are generated at the channel level.

On CIFAR: On CIFAR-10 and CIFAR-100 datasets, the mini-batch size is set to 128 in ResNet, WideResNet and 96 in DenseNet and PyramidNet on four GPUs. The training epochs are set to 200 in WideResNet and 300 in ResNet, DenseNet and PyramidNet. In ResNet, the learning rate starts from 0.25 and is divided by 10 at the 150th and 225th epoch. In WideResNet, the learning rate starts from 0.1 and is divided by 5 at the 60th, 120th, and 160th epoch. In DenseNet and PyramideNet, the learning rate starts from 0.1 and is divided by 10 at the 150th and 225th epoch. The weight decay is set to $1e - 4$.

On Regular ImageNet: The mini-batch size is set to 256 in PyaramidNet on four GPUs. The training epochs are set to 90. The learning rate starts from 0.05 and is divided by 10 at the 30th, 60th, and 80th epoch. The weight decay is set to $1e - 4$.

On Low Resolution ImageNet: The mini-batch size is set to 128 in WideResNet on four GPUs. The training epochs are set to 40. In WideResNet, the learning rate starts from 0.01 and is divided by 5 at the 10th, 20th, and 30th epoch. The weight decay is set to $2e - 4$.

Finally, SGD and Nesterov momentum [13] are used in training on these datasets. Our experiments are conducted from three aspects: 1) compare the generalization in Section V-C; 2) compare the optimization in Section V-D; and 3) ablation comparisons in Section V-E, including comparisons between the additional noise and multiplicative noise, flexibility, independence of the additional noise, and the effects of different components in the Addi-Reg method.

C. Generalization Comparisons

On Low Resolution ImageNet-32 × 32 and ImageNet-16 × 16 Datasets: Table I compares the test errors of the baseline networks (WideResNets) and the proposed Addi-Reg methods on the low resolution ImageNet-32 × 32 and ImageNet-16 × 16 datasets. It is apparent that, our Addi-Reg method achieves promising generalization improvements compared to the baseline WideResNets on these two datasets. Especially, on the more challenge low resolution ImageNet-16 × 16, our method decreases the Top-1 and Top-5 error rates by 4.54% and 4.96% when it is employed in the smallest networks WideResNet ($k = 3$). This demonstrates that, despite small models suffer from a minor overfitting, our Addi-Reg method can still significantly improve the generalization. Therefore, our method can effectively address the overfitting problem caused by less abundant spatial information in the training datasets. This also proves our Addi-Reg method is more general to different datasets compared to the Multi-Reg methods.

Table II shows the comparisons between the Addi-Reg method and other popular Multi-Reg methods. The results show that, only our Addi-Reg improves the generalization with a significant drop of the test errors on both these two datasets. Since the information of these two low resolution datasets is much less abundant than the regular ImageNet, turning off the nodes or shaking the intermediate feature may cause excessive regularization and thus aggravate the poor quality of the feature. Therefore, Dropout and ShakeDrop are inferior to ours. Furthermore, ShakeDrop method produces poor performances on ImageNet-16 × 16 datasets. ShakeDrop utilizes manual-designed noise distributions to generated multiplicative noise for networks. These noise distributions are not individualized to different datasets. Thus, for the datasets with more difficulties, ShakeDrop method may easily fail to regularize the networks. However, in our method, the noise distributions can

TABLE II
COMPARISONS ON LOW RESOLUTION IMAGENET DATASETS. THE BASELINE NETWORK IS WIDERESNET ($k = 10$) WITH DEPTH 28. THE ERROR RATES MARKED BY ⋆ ARE FROM [45]. THE OTHER RESULTS WITHOUT ANY MARKS ARE FROM OUR IMPLEMENTATIONS. "↑" INDICATES THE METHOD PRODUCES WORSE PERFORMANCE COMPARED TO BASELINE, AND, THE OPPOSITE IS REPRESENTED BY "↓"

Methods	ImageNet-32 × 32			ImageNet-16 × 16		
	Params	Top-1 err. (%)	Top-5 err. (%)	Params	Top-1 err. (%)	Top-5 err. (%)
Baseline	37.1M	40.96⋆	18.87⋆	8.9M	59.94⋆	35.10⋆
+ Dropout	37.1M	50.23 (↑ 9.27)	25.95 (↑ 7.08)	8.9M	61.37 (↑ 1.43)	36.74 (↑ 1.64)
+ ShakeDrop	37.1M	40.44 (↓ 0.52)	18.12 (↓ 0.75)	8.9M	61.79 (↑ 1.85)	37.30 (↑ 2.20)
+ **Addi-Reg (ours)**	37.9M	**39.34** (↓ **1.62**)	**16.79** (↓ **2.08**)	9.1M	**56.78** (↓ **3.16**)	32.25 (↓ **2.85**)

TABLE III
COMPARISONS OF ERROR RATES ON CIFAR DATASETS. THE BASELINE NETWORKS ARE WIDERESNET ($k = 10$), DENSENET-BC ($growth = 40$) AND PYRAMIDNET-BC-α200. THE RESULTS MARKED BY ⋆ ARE FROM THE RELEVANT LITERATURE LISTED IN THE LEFT COLUMN. "↑" INDICATES THE METHOD PRODUCES WORSE PERFORMANCE COMPARED TO BASELINE, AND, THE OPPOSITE IS REPRESENTED BY "↓"

Methods	Params (M)	CIFAR-10 err. (%)	CIFAR-100 err. (%)
WideResNet ($k = 10$) (baseline) [9]	36.5	4.00⋆	19.25⋆
WideResNet ($k = 10$) + ShakeDrop	36.5	3.78 (↓ 0.22)	20.50⋆ (↑ 1.25)
WideResNet ($k = 10$) + **Addi-Reg (ours)**	36.8	**3.45**(↓ **0.55**)	**18.63** (↓ **0.62**)
DenseNet-BC ($growth = 40$) (baseline) [11]	25.6	3.46⋆	17.18⋆
DenseNet-BC ($growth = 40$) + ShakeDrop (our imple.)	25.6	3.86 (↑ 0.40)	18.39 (↑ 1.21)
DenseNet-BC ($growth = 40$) + **Addi-Reg (ours)**	25.6	**3.29** (↓ **0.17**)	**17.00** (↓ **0.18**)
PyramidNet-BC-α200 (baseline) [10]	26.2	3.31⋆	16.35⋆
PyramidNet-BC-α200 + ShakeDrop [29]	26.2	3.41⋆ (↑ 0.10)	14.96⋆ (↓ 1.39)
PyramidNet-BC-α200 + ShakeDrop [29] (our imple.)	26.2	3.40 (↑ 0.09)	15.97 (↓ 0.38)
PyramidNet-BC-α200 + **Addi-Reg (ours)**	26.8	3.06 (↓ 0.25)	15.79 (↓ 0.56)
PyramidNet-BC-α200 + ShakeDrop + **Addi-Reg (our best)**	26.8	**2.58** (↓ **0.73**)	**14.11** (↓ **2.24**)

TABLE IV
COMPARISONS OF DIFFERENT REGULARIZATION METHODS ON CIFAR DATASETS. THE BASELINE NETWORKS ARE RESNET-BC-110. "↑" INDICATES THE METHOD PRODUCES WORSE PERFORMANCE COMPARED TO "MAXUP + CUTMIX," AND, THE OPPOSITE IS REPRESENTED BY "↓"

Networks	Regularization methods	CIFAR-10		CIFAR-100	
		Top-1 err. (%)	Top-5 err. (%)	Top-1 err. (%)	Top-5 err. (%)
ResNet-BC-110	Maxup + CutMix	5.40	0.16	23.38	5.49
ResNet-BC-110	DropBlock + CutMix	4.57 (↓ 0.83)	0.15 (↓ 0.01)	21.14 (↓ 2.24)	4.90 (↓ 0.59)
ResNet-BC-110	ShakeDrop + CutMix	7.65 (↑ 2.25)	0.22 (↑ 0.06)	23.57 (↑ 0.19)	5.25 (↓ 0.24)
ResNet-BC-110	**Addi-Reg (ours)** + CutMix	**4.45** (↓ **0.95**)	**0.12** (↓ **0.04**)	**20.92** (↓ **2.46**)	**4.56** (↓ **0.93**)
ResNet-BC-110	**Addi-Reg (ours)** + Maxup + CutMix	**4.23** (↓ **1.17**)	**0.11** (↓ **0.05**)	**20.06** (↓ **3.32**)	**4.15** (↓ **1.34**)

be learned in training, and we employ BA mechanism to pertinently inject the noise. Hence, our method can produce more adaptive and suitable noise for different datasets and regularize the networks effectively. This further demonstrates that our Addi-Reg method is more universal to different datasets than the existing Multi-Reg method.

On CIFAR-10 and CIFAR-100 Datasets: Initially, our Addi-Reg method is performed on CIFAR-10 and CIFAR-100 datasets in WideResNet, DenseNet and PyramidNet to evaluate the generalization. The experimental results are summarized in Table III. From Table III, our Addi-Reg achieves better generalization than all the baseline models on both CIFAR-10 and CIFAR-100 with only a small improvement of parameters, while ShakeDrop does not gain the same performance. Furthermore, the Addi-Reg also outperforms the ShakeDrop in almost all cases. In addition, when the ShakeDrop is combined with our Addi-Reg in PyramidNet, the test errors are significantly decreased compared to the ShakeDrop. Notably, the proposed Addi-Reg can achieves the state-of-the-art results on the CIFAR-10 with 2.58% error and the CIFAR-100 with 14.11% error, respectively. Moreover, we can see that, DenseNet utilizes dense concatenation instead of identity-mapping addition, which may also cause the perturbations in each layer when the feature are weighed by the ShakeDrop. This may be the reason why ShakeDrop significantly improves the test errors in DenseNet. However, since our Addi-Reg exploits the mechanisms proposed in Section IV to control the additional noise, it can maintain the feature as stable as possible. Therefore, it can still improve the generalization without the identity-mapping. This also indicates our method is more universal than ShakeDrop.

Furthermore, our methods are also compared to current state-of-the-art regularization method (such as Maxup + CutMix [56]) by augmenting the training images. Experimental results are shown in Table IV. From this table, we can observe that, compared to the method of "Maxup + CutMix," our method "Addi-Reg + CutMix" decreases the Top-1 error rates 0.95% and 2.46% on CIFAR-10 and

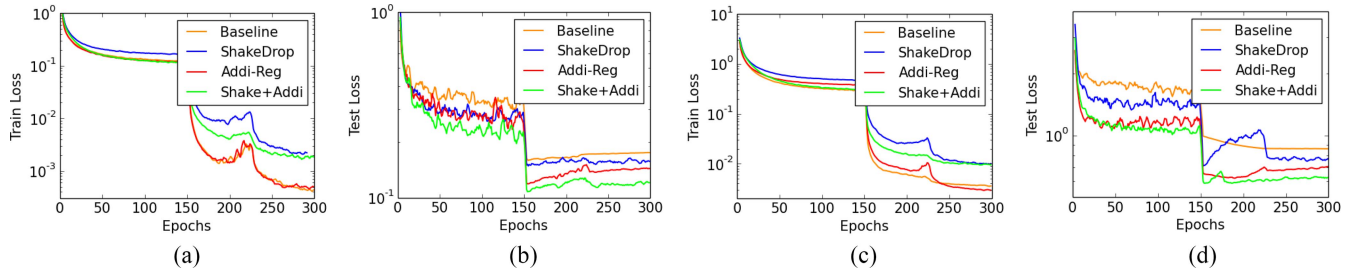


Fig. 7. Training loss and test error on CIFAR. PyramidNet-BC-α200 with depth 272 is the baseline network. (a) Training loss on CIFAR-10. (b) Test loss on CIFAR-10. (c) Training loss on CIFAR-100. (d) Test loss on CIFAR-100.

TABLE V
COMPARISONS ON REGULAR IMAGENET DATASETS. THE BASELINE NETWORK IS PYRAMIDNET-BC-α300 WITH DEPTH 50. THE ERROR RATES MARKED BY $\star$ ARE FROM [29]

Methods	Params (M)	Top-1 err. (%)
Baseline	16.9	24.06*
+ RandomDrop	16.9	23.89* ($\downarrow$ 0.17)
+ ShakeDrop	16.9	23.95* ($\downarrow$ 0.11)
+ **Addi-Reg** (**ours**)	17.2	**23.36** ($\downarrow$ **0.70**)

TABLE VI
COMPARISONS OF TEST ERROR (%) FROM LONG-TERM TRAINING ON CIFAR. THE BASELINE MODEL IS PYRAMIDNET-BC-α200 WITH DEPTH 272. C10: CIFAR-10, C100: CIFAR-100

Methods	Dataset	300 Epoch	1800 Epoch
ShakeDrop [29]	C10	3.41*	2.67*
ShakeDrop + **Addi-Reg**		**2.58**	–
ShakeDrop [29]	C100	14.90*	**13.99***
ShakeDrop + **Addi-Reg**		**14.11**	–

CIFAR-100 datasets, respectively. Furthermore, compared to the methods (DropBlock and ShakeDrop) with regularizing in the feature space, our Addi-Reg method also achieves the best performances on both the CIFAR-10 and CIFAR-100 datasets. Finally, since the proposed method is orthogonal to the Maxup and CutMix methods, the Addi-Reg method is performed on these two methods and produces the lowest error rates on these two CIFAR datasets. According to these comparisons from different aspects, the proposed Addi-Reg method achieves satisfactory regularization effectiveness in various cases.

On Regular ImageNet Dataset: Table V compares the Addi-Reg method and Multi-Reg methods on regular ImageNet dataset. In this table, RandomDrop and ShakeDrop can only achieve a small decay in the test error with 0.17% and 0.11%, respectively. However, our Addi-Reg produces a better generalization with decreasing the test error by a margin of 0.70%. This further indicates our method can effectively regularize the networks on the regular datasets.

Through the experiments on these five datasets, it sufficiently demonstrates the proposed Addi-Reg method can effectively regularize the networks compared to Multi-Reg methods, which also validate our claim that the proposed method is more universal for different networks and different datasets.

D. Optimization Comparisons

Comparisons of Convergence: In order to validate the superiority on training convergence of the proposed Addi-Reg method, Fig. 7 separately illustrates the training losses and test losses of baseline, ShakeDrop, Addi-Reg, and ShakeDrop+Addi-Reg on CIFAR. Apparently, in Fig. 7(a) and (c), ShakeDrop has the slowest convergence and the highest training loss, while our Addi-Reg method can have a similar convergence compared to the baseline networks. Particulary, it even produces lower training loss than the baseline on CIFAR-100. For the test loss, as shown in Fig. 7(b) and (d), on these two datasets, the Addi-Reg method also achieves a better generalization than those of the ShakeDrop and baseline. Different from the Multi-Reg methods, who increase generalization but sacrifice optimization, Fig. 7 demonstrates our Addi-Reg can simultaneously improves the generalization and optimization.

Addi-Reg Drives Faster Convergence of Multi-Reg: ShakeDrop converges slowly, leading to more training epochs requirement to improve the performance. For example, the test error is greatly decreased when the model is trained for 1800 epochs compared to that of 300 epochs [29]. However, it lacks adequate elegant and practical abilities to spend such tremendous amounts of time to train a model. Therefore, Addi-Reg is performed on ShakeDrop to test whether our method can accelerate the convergence of ShakeDrop. From Fig. 7, the convergence speed is improved effectively. Furthermore, in Table VI, compared to the ShakeDrop's test error of 1800 epochs, the assembled method achieves lower test error on CIFAR-10 with only 300 epochs. It also produces competitive test error compared to the result of 1800 epochs on CIFAR-100. These results prove the proposed Addi-Reg method can accelerate the convergence of unstable regularization methods and achieve a satisfactory generalization in a short-term training. The proposed Addi-Reg method generates learnable additional noise instead of random multiplicative noise for every layer in networks. This indicates that the additional noise can be injected into feature space on the basis of regularization with the Multi-Reg method. Since the additional noise can be adaptively predicted by the proposed Addi-Reg method, the additional noise will preserve learning stability of the networks through the learnable noise distributions. Therefore, although the Multi-Reg method cause an unstable learning in regularization, the Addi-Reg method can alleviate this problem to

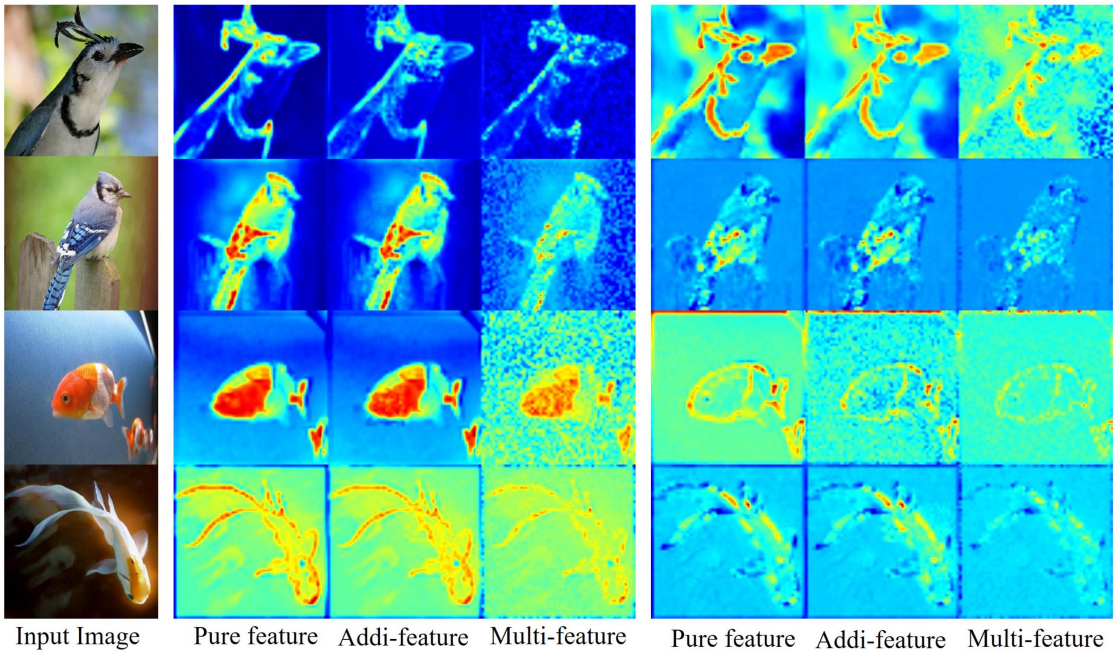


Fig. 8. Visualization of original pure feature (*Pure feature*), noisy feature produced by our Addi-Reg method (*Addi-Feature*) and noisy feature produced by Multi-Reg (ShakeDrop) method (*Multi-feature*). The baseline network is PyramidNet-BC-$\boldsymbol{\alpha}$300 with depth 50.

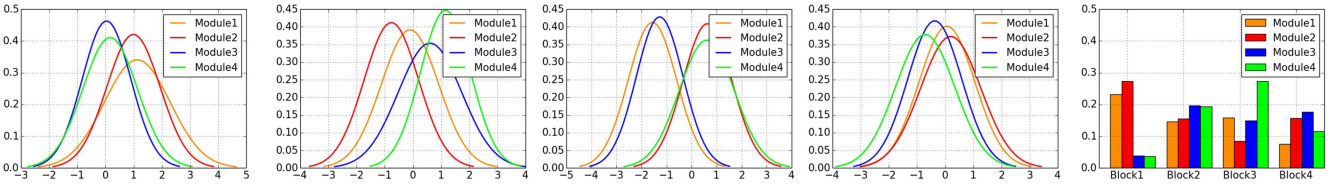


Fig. 9. Visualization of the learned noise distribution. The baseline network is PyramidNet-BC-$\boldsymbol{\alpha}$300 with depth 50, which has four blocks with four modules in each block. The first four figures are the Gaussian distributions from Block1 to Block4. The last figure is the comparisons of α in the Uniform distribution $U(-\alpha, \alpha)$ from Block1 to Block4.

some extent. This leads to a faster convergence with better generalization performance.

E. Ablation Comparisons

Comparisons Between the Additional Noise and Multiplicative Noise: The main reason of the optimization problems of Multi-Reg methods is that the multiplicative noise may contribute to a great change of the distributions between the original and noisy data. Therefore, we propose the Addi-Reg method to produce suitable additional noise and maintain the feature's distribution as stable as possible. In order to prove our Addi-Reg method has much smaller change of distributions between the original and noisy data than the Multi-Reg methods, we explicitly show the original pure feature, noisy feature of Addi-Reg and Multi-Reg method in Fig. 8. It is apparent that, compared to the Multi-Reg methods, the noisy feature produced by our Addi-Reg method is much more similar to the original pure feature. This implies the proposed Addi-Reg method can effectively stabilize the distributions between the original and noisy data and thus lead to stabler learning. Moreover, these comparisons also confirm that, in the Addi-Reg method, the proposed mechanisms, that is, generating noise from learnable distributions and employing BA module to produce small additional noise, can accomplish the goal of initial design.

Flexibility of the Additional Noise in Addi-Reg: In the Addi-Reg, since the noise distribution is learned in training without any manual searches, it can effectively improve the flexibility of the regularization. In Fig. 9, we explicitly plot the learned Gaussian distributions and the values of α from Uniform distributions in the PyramidNet-BC-$\boldsymbol{\alpha}$300 with depth 50 on Regular ImageNet datasets. In different layers, the learned noise distributions also have discrepancies. According to the experimental results, we can observe that our method can have a better generalization than the Multi-Reg methods. The differences among these learned noise distributions in our method indicate the Addi-Reg method is more flexible and can produce more adaptive and suitable noise for various layers.

Furthermore, in order to further validate the effectiveness of the learned noise distributions, the learned noise and random noise are injected in feature space to regularize the networks, respectively. The results of comparisons are shown in Table VII. The additional noise in the Addi-Reg method is produced through the summation of Gaussian noise and uniform noise. The random noise are generated from

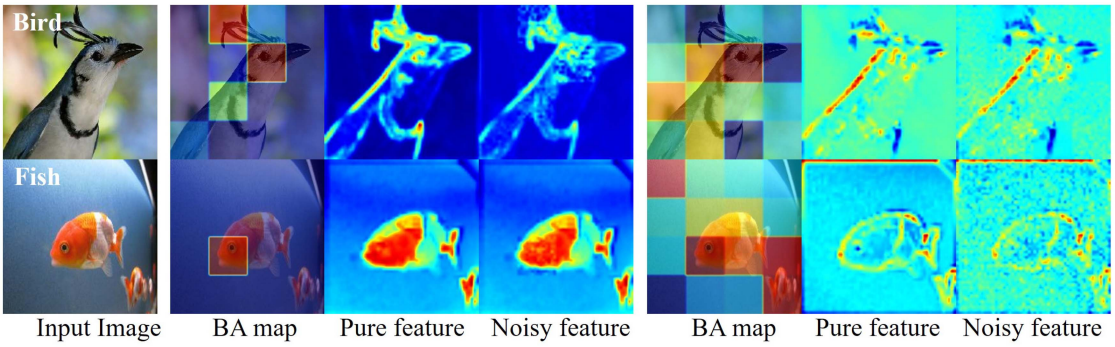


Fig. 10. Visualization of the BA map, pure feature map, and the noisy feature map.

TABLE VII
COMPARISONS OF ERROR RATES FROM LEARNABLE NOISE AND RANDOM NOISE ON CIFAR. THE BACKBONE NETWORKS ARE WIDERESNET ($k = 10$)

Gaussian noise	Uniform noise	CIFAR-10 (%)	CIFAR-100 (%)
random	random	4.09	19.28
random	learnable	3.88	19.10
learnable	random	3.68	19.12
learnable	learnable	**3.45**	**18.63**

TABLE VIII
COMPARISONS OF DIFFERENT COMPONENTS ON CIFAR-100. BASELINE SETTINGS ARE SAME AS THE INITIALIZATIONS IN SECTION V-B EXPLOITED ON PYRAMIDNET-BC-α200 WITH DEPTH 272. §, † AND ‡ INDICATE REPLACING THE SETTINGS IN BA MODULE, PREDICTING WEIGHT MODULE AND ADDITIONAL NOISE

Settings	Err. (%)	Settings	Err. (%)
Baseline	**15.79**	Gaussian Noise‡	15.96
Global Noise§	16.09	Uniform Noise‡	16.34
Block Noise§	16.28	Pixel level†	16.84
Attention Noise§	15.90	Drop Sharing filter	16.92
Max-Pool§	15.88	Drop Weight module	16.30

manual-designed noise distributions, which are determined through our searching under the best performances. Clearly, in this table, the best performance is achieved by generating noise from both learnable Gaussian noise and learnable uniform noise distributions. However, generating noise from both manual-designed noise distributions cannot produce satisfactory regularization effect. These results demonstrated that the proposed Addi-Reg method may predict more appropriate noise for the networks, leading to better regularization.

Independence of the Additional Noise in Addi-Reg: To test the independence of noise, we respectively turn off the ShakeDrop and Addi-Reg in the assembled method in testing phase. We find when removing the Addi-Reg term, it decreases the performance slightly (2.70% versus 2.58% on CIFAR-10, 14.39% versus 14.11% on CIFAR-100), while ShakeDrop is completely failed when it is cut off. This proves the additional noise in Addi-Reg is less sensitive to input feature, leading to stabler learning.

Effects of Different Components in Addi-Reg: We, respectively, replace the components in Addi-Reg with the settings listed in Table VIII.

1) It proves the BAN with average pooling is more effective than the Global noise, BN, AN and max pooling in the BA module. To explicitly verify the effectiveness of the BA module, we visualize this process in Fig. 10. It is clear that, BA module can only blur the feature in the required areas rather than the global feature. Moreover, the difference between the pure feature and noisy feature is trivial, which is consistent with our goal.
2) The additional noise generated from both the Gaussian noise and uniform noise performs better.
3) The weights generated at channel level in the weight module are better than the pixel level.
4) Dropping the sharing filters and weight module[2] will handicap the generalization, respectively. These comparisons confirm the proposed method shown in Section IV is reasonable.

VI. CONCLUSION

This article provided a new viewpoint that injecting noise into hidden layers in CNNs can be seen as implicitly augmenting the input data. Based on this, the Multi-Reg's fundamental mechanism, that is, regularizing the networks in a MiM way, was explicitly revealed and thus the optimization decay in Multi-Reg was proved. Subsequently, we highlighted the discover that the way of weighing the feature to regularize the networks could greatly affect the distributions between the original features and the noised feature. Finally, the novel Addi-Reg method was proposed to generate additional noise instead of multiplicative noise from learnable distributions. Particularly, Addi-Reg applies BA module to limit the additional noise. It shares the filters on the pure and noisy data and employs a predicting weight module to fuse the outputs, which can make the learning stable from both the filters and feature aspects. Extensive experiments proved the flexibility and university of the proposed Addi-Reg method, which also holds a better generation-optimization tradeoff than the state-of-the-art Multi-Reg models. Furthermore, we hope our insightful discover of MiM can provide a new direction to stable regularization methods.

[2]When the weight module is dropped, the concatenation is utilized to combine pure feature and noisy feature.

REFERENCES

[1] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016.

[2] C. Szegedy *et al.*, "Going deeper with convolutions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun 2015, pp. 1–9.

[3] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proc. 32nd Int. Conf. Mach. Learn.*, Jul. 2015, pp. 448–456.

[4] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 2818–2826.

[5] C. Szegedy, S. Ioffe, A. Alemi, and V. Vanhoucke, "Inception-v4, inception–ResNet and the impact of residual connections on learning," 2016. [Online]. Available: arXiv:1602.07261.

[6] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Los Alamitos, CA, USA, Jun. 2016, pp. 770–778.

[7] S. Xie, R. Girshick, P. Dollar, Z. Tu, and K. He, "Aggregated residual transformations for deep neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 5987–5995.

[8] S. Targ, D. Almeida, and K. Lyman, "ResNet in ResNet: Generalizing residual architectures," 2016. [Online]. Available: arXiv:1603.08029.

[9] S. Zagoruyko and N. Komodakis, "Wide residual networks," in *Proc. Brit. Mach. Vis. Conf. (BMVC)*, Sep. 2016, pp. 87.1–87.12.

[10] D. Han, J. Kim, and J. Kim, "Deep pyramidal residual networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Los Alamitos, CA, USA, Jul. 2017, pp. 6307–6315.

[11] G. Huang, Z. Liu, L. Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Los Alamitos, CA, USA, Jul. 2017, pp. 2261–2269.

[12] L. Bottou, "Online algorithms and stochastic approximations," in *Online Learning and Neural Networks*, vol. 5, D. Saad, Ed. Cambridge, U.K.: Cambridge Univ. Press, 1998.

[13] I. Sutskever, J. Martens, G. Dahl, and G. Hinton, "On the importance of initialization and momentum in deep learning," in *Proc. Int. Conf. Mach. Learn.*, 2013, pp. 1139–1147.

[14] S. J. Nowlan and G. E. Hinton, "Simplifying neural networks by soft weight-sharing," *Neural Comput.*, vol. 4, no. 4, pp. 473–493, Jul. 1992.

[15] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Commun. ACM*, vol. 60, no. 6, pp. 84–90, May 2017.

[16] L. Prechelt, *Early Stopping—But When?* Berlin, Germany: Springer, 2012, pp. 53–67.

[17] J. Gou, L. Wang, Z. Yi, Y. Yuan, W. Ou, and Q. Mao, "Weighted discriminative collaborative competitive representation for robust image classification," *Neural Netw.*, vol. 125, pp. 104–120, May 2020.

[18] Y. He, X. Zhang, and J. Sun, "Channel pruning for accelerating very deep neural networks," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 1398–1406.

[19] Y. Lu, G. Lu, R. Lin, J. Li, and D. Zhang, "SRGC-Nets: Sparse repeated group convolutional neural networks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, no. 8, pp. 2889–2902, Aug. 2020.

[20] Y. Lu, G. Lu, J. Li, Y. Xu, and D. Zhang, "High-parameter-efficiency convolutional neural networks," *Neural Comput. Appl.*, vol. 32, no. 14, pp. 10633–10644, 2020.

[21] Y. Lu, G. Lu, B. Zhang, Y. Xu, and J. Li, "Super sparse convolutional neural networks," in *Proc. 33rd AAAI Conf. Artif. Intell.*, vol. 33, 2019, pp. 4440–4447.

[22] N. Ma, X. Zhang, H.-T. Zheng, and J. Sun, "ShuffleNet V2: Practical guidelines for efficient CNN architecture design," in *Computer Vision—ECCV*, V. Ferrari, M. Hebert, C. Sminchisescu, and Y. Weiss, Eds. Cham, Switzerland: Springer, 2018, pp. 122–138.

[23] J. Gou, B. Yu, S. J. Maybank, and D. Tao, "Knowledge distillation: A survey," 2020. [Online]. Available: arXiv:2006.05525.

[24] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res.*, vol. 15, no. 56, pp. 1929–1958, 2014.

[25] L. Wan, M. D. Zeiler, S. Zhang, Y. LeCun, and R. Fergus, "Regularization of neural networks using dropconnect," in *Proc. 30th Int. Conf. Mach. Learn. (ICML)*, Atlanta, GA, USA, Jun. 2013, pp. 1058–1066.

[26] G. Huang, Y. Sun, Z. Liu, D. Sedra, and K. Weinberger, "Deep networks with stochastic depth," in *Proc. Eur. Conf. Comput. Vis.*, 2016, pp. 646–661.

[27] G. Ghiasi, T.-Y. Lin, and Q. V. Le, "DropBlock: A regularization method for convolutional networks," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 31, 2018, pp. 10727–10737.

[28] X. Gastaldi, "Shake-shake regularization," 2017. [Online]. Available: arXiv:1705.07485.

[29] Y. Yamada, M. Iwamura, and K. Kise, "Shakedrop regularization," 2018. [Online]. Available: arXiv:1802.02375v1.

[30] X. Chang, Z. Ma, Y. Yang, Z. Zeng, and A. G. Hauptmann, "Bi-level semantic representation analysis for multimedia event detection," *IEEE Trans. Cybern.*, vol. 47, no. 5, pp. 1180–1197, May 2017.

[31] C. Zhang, Z. Yu, H. Fu, P. Zhu, L. Chen, and Q. Hu, "Hybrid noise-oriented multilabel learning," *IEEE Trans. Cybern.*, vol. 50, no. 6, pp. 2837–2850, Jun. 2020.

[32] B. Du, T. Xinyao, Z. Wang, L. Zhang, and D. Tao, "Robust graph-based semisupervised learning for noisy labeled data via maximum correntropy criterion," *IEEE Trans. Cybern.*, vol. 49, no. 4, pp. 1440–1453, Apr. 2019.

[33] H. Wang, P. X. Liu, and B. Niu, "Robust fuzzy adaptive tracking control for nonaffine stochastic nonlinear switching systems," *IEEE Trans. Cybern.*, vol. 48, no. 8, pp. 2462–2471, Aug. 2018.

[34] Z. Lai, D. Mo, W. K. Wong, Y. Xu, D. Miao, and D. Zhang, "Robust discriminant regression for feature extraction," *IEEE Trans. Cybern.*, vol. 48, no. 8, pp. 2472–2484, Aug. 2018.

[35] L. Bottou and O. Bousquet, "The tradeoffs of large scale learning," in *Proc. Adv. Neural Inf. Process. (NISP)*, vol. 20, 2007, pp. 1–8.

[36] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol, "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion," *J. Mach. Learn. Res.*, vol. 11, pp. 3371–3408, Dec. 2010.

[37] S. Sabour, N. Frosst, and G. E. Hinton, "Dynamic routing between capsules," 2017. [Online]. Available: arXiv:1710.09829.

[38] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.

[39] S. Ruder, "An overview of gradient descent optimization algorithms," 2016. [Online]. Available: arXiv:1609.04747.

[40] E. Moulines and F. R. Bach, "Non-asymptotic analysis of stochastic approximation algorithms for machine learning," in *Advances in Neural Information Processing Systems 24*, J. Shawe-Taylor, R. S. Zemel, P. L. Bartlett, F. Pereira, and K. Q. Weinberger, Eds. Red Hook, NY, USA: Curran Assoc., Inc., 2011, pp. 451–459.

[41] P. Chaudhari and S. Soatto, "Stochastic gradient descent performs variational inference, converges to limit cycles for deep networks," in *Proc. Inf. Theory Appl. Workshop (ITA)*, Feb. 2018, pp. 1–10.

[42] L. Bottou, F. E. Curtis, and J. Nocedal, "Optimization methods for large-scale machine learning," *SIAM Rev.* vol. 60, no. 2, pp. 223–311, 2016.

[43] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," 2014. [Online]. Available: arXiv:1312.6114.

[44] J. Hu, L. Shen, S. Albanie, G. Sun, and E. Wu, "Squeeze-and-excitation networks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 42, no. 8, pp. 2011–2023, Aug. 2020.

[45] P. Chrabaszcz, I. Loshchilov, and F. Hutter, "A downsampled variant of ImageNet as an alternative to the CIFAR datasets," 2017. [Online]. Available: arXiv:1707.08819.

[46] A. G. Howard *et al.*, "MobileNets: Efficient convolutional neural networks for mobile vision applications," 2017. [Online]. Available: arXiv:1704.04861.

[47] M. Sandler, A. G. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Inverted residuals and linear bottlenecks: Mobile networks for classification, detection and segmentation," 2018. [Online]. Available: arXiv:1801.04381.

[48] X. Zhang, X. Zhou, M. Lin, and J. Sun, "ShuffleNet: An extremely efficient convolutional neural network for mobile devices," 2017. [Online]. Available: arXiv:1707.01083.

[49] T. Zhang, G. Qi, B. Xiao, and J. Wang, "Interleaved group convolutions," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Los Alamitos, CA, USA, Oct. 2017, pp. 4383–4392.

[50] G. Xie, J. Wang, T. Zhang, J. Lai, R. Hong, and G. Qi, "Interleaved structured sparse convolutional neural networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Los Alamitos, CA, USA, Jun. 2018, pp. 8847–8856.

[51] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," Dept. Comput. Sci., Univ. Toronto, Toronto, ON, Canada, Rep. TR-2009, 2009.

[52] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2009, pp. 248–255.

[53] A. Torralba, R. Fergus, and W. T. Freeman, "80 million tiny images: A large data set for nonparametric object and scene recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 30, no. 11, pp. 1958–1970, Nov. 2008.

[54] K. He, X. Zhang, S. Ren, and J. Sun, "Identity mappings in deep residual networks," in *Computer Vision—ECCV*, B. Leibe, J. Matas, N. Sebe, and M. Welling, Eds. Cham, Switzerland: Springer, 2016, pp. 630–645.

[55] G. Larsson, M. Maire, and G. Shakhnarovich, "FractalNet: Ultra-deep neural networks without residuals," 2016. [Online]. Available: arXiv:1605.07648.

[56] C. Gong, T. Ren, M. Ye, and Q. Liu, "MaxUp: A simple way to improve generalization of neural network training," 2020. [Online]. Available: arXiv:2002.09024.

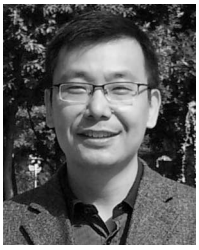
Yao Lu received the B.S. degree from the Department of Computer Science and Technology, Huaqiao University, Xiamen, China, in 2015, and the Ph.D. degree from the Department of Computer Science and Technology, Harbin Institute of Technology, Harbin, China, in 2020.

Her research interests include pattern recognition, deep learning, computer vision, and relevant applications.

Zheng Zhang (Senior Member, IEEE) received the M.S. degree in computer science and the Ph.D. degree in computer applied technology from the Harbin Institute of Technology, Harbin, China, in 2014 and 2018, respectively.

He was a Postdoctoral Research Fellow with the University of Queensland, Brisbane, QLD, Australia. He is currently an Assistant Professor with the Harbin Institute of Technology, Shenzhen, China. He has published over 50 technical papers at prestigious international journals and conferences. His current research interests include machine learning, computer vision, and multimedia analytics.

Dr. Zhang serves/served as a Guest Editor for *Neurocomputing*, a Publication Chair of the 16th International Conference on Advanced Data Mining and Applications (ADMA 2020), and an SPC/PC member of several top conferences.



Guangming Lu (Member, IEEE) received the B.S. degree in electrical engineering, the M.S. degree in control theory and control engineering, and the Ph.D. degree in computer science from the Harbin Institute of Technology, Harbin, China, in 1998, 2000, and 2005, respectively.

From 2005 to 2007, he was a Postdoctoral Fellow with Tsinghua University, Beijing, China. He is currently a Professor with the Harbin Institute of Technology, Shenzhen, China. His research interests include pattern recognition and automated biometric technologies and applications.

Yicong Zhou (Senior Member, IEEE) received the B.S. degree in electrical engineering from Hunan University, Changsha, China, in Changsha, China, and the M.S. and Ph.D. degrees in electrical engineering from Tufts University, Medford, MA, USA.

He is currently an Associate Professor and a Director of the Vision and Image Processing Laboratory, Department of Computer and Information Science, University of Macau, Macau, China. His research interests include image processing, computer vision, machine learning, and multimedia security.

Jinxing Li received the B.S. degree from the Department of Automation, Hangzhou Dianzi University, Hangzhou, China, in 2012, the M.S. degree from the Department of Automation, Chongqing University, Chongqing, China, in 2015, and the Ph.D. degree from Biometrics Research Centre, Hong Kong Polytechnic University, Hong Kong, in 2018.

His research interests are pattern recognition, medical biometrics, and machine learning.

David Zhang (Life Fellow, IEEE) graduated in computer science from Peking University, Beijing, China, the M.Sc. and Ph.D. degrees in computer science from the Harbin Institute of Technology (HIT), Harbin, China, in 1982 and 1985, respectively, and the second Ph.D. degree in electrical and computer engineering from the University of Waterloo, Waterloo, ON, Canada, in 1994.

From 1986 to 1988, he was a Postdoctoral Fellow with Tsinghua University, Beijing, and an Associate Professor with Academia Sinica, Beijing. He is currently a Professor with the Chinese University of Hong Kong (Shenzhen), Shenzhen, China. He also serves as a Visiting Chair Professor with Tsinghua University and HIT, Shenzhen, and an Adjunct Professor with Shanghai Jiao Tong University, Shanghai, China, Peking University, Beijing, National University of Defense Technology, Changsha, China, and the University of Waterloo. He has published over 20 monographs, 400 international journal papers, and 40 patents from USA/Japan/HK/China.

Prof. Zhang is the Founder and the Editor-in-Chief of *International Journal of Image and Graphics*; a Book Editor of *International Series on Biometrics* (Springer); Organizer of the first International Conference on Biometrics Authentication; an Associate Editor of more than ten international journals, including IEEE TRANSACTIONS and so on. He is a Croucher Senior Research Fellow, a Distinguished Speaker of the IEEE Computer Society, and a Fellow of IAPR.