Digital Signal Processing 123 (2022) 103397

Contents lists available at ScienceDirect

Digital Signal Processing

www.elsevier.com/locate/dsp

Necessary conditions for convergence of CNNs and initialization of convolution kernels

Huibin Zhang [a,b,*], Liping Feng [b], Xiaohua Zhang [a,c], Yuchi Yang [a,d], Jing Li [b]

[a] Institute of Information Science and Technology, Yanshan University, Qinhuang Dao, 066004, Hebei province, China
[b] Computer Department of Xinzhou Teachers University, Xinzhou, 034000, Shanxi province, China
[c] School of Mathematics and Information Science and Technology, Hebei Normal University of Science and Technology, Qinhuangdao, 066004, China
[d] Hebei University of Economics and Business, School of Information Technology, Shijiazhuang, 050000, Hebei province, China

ARTICLE INFO

Article history:
Available online 11 January 2022

Keywords:
Convolutional Neural Networks (CNNs)
Convergence
Batch normalization
Initialization of the convolution kernel
Initial learning rate
Gaussian distribution

ABSTRACT

Despite the great success of deep learning in many fields such as computer vision, natural language processing, and information retrieval, there are relatively few studies on the convergence of deep convolutional neural networks (CNNs), and there is a lack of theoretical studies on the necessary conditions for the convergence of CNNs. The initialization of the convolution kernel of CNNs is an important factor in whether the network can converge. However, the existing initialization methods do not analyze the influence of their methods on the convergence performance of CNNs and did not analyze the conditions of their application, and thus the performance is not the best in different network models. In this work, the computational process of both forward and backward propagation of CNNs is considered as a mapping in linear normed space, and thus a necessary condition for CNNs stable converge is proposed. According to this necessary condition of convergence, we first derive initialization formulas for plain networks applicable to any activation function, and derive the initialization method of plain networks whose activation function is ReLU and PReLU. Secondly, the necessary conditions for convergence of CNNs proposed in this work can explain the mathematical reasons why the BN contribute to the training of CNNs, and this problem has always been an active research topic. Finally, we find that the learning rate and the initialization of the convolutional kernel jointly affect the convergence performance of the network. Based on the plain networks convolution kernel initialization method, we derive the convolution kernel initialization method of network with BN layer related to the learning rate. In order to verify the effectiveness of the proposed initialization method, we test it on the CIFAR-10 and CIFAR-100 datasets, and performed image classification with four network models (VGG-19, ResNet-110, DenseNet-100 and WideResNet28-10), and the experimental results showed that the initialization method proposed in this work improved the accuracy of image classification.

1. Introduction

Deep convolutional neural networks (CNNs) [1] have been crucial to the success of deep learning, and deep learning has made impressive progress on a wider variety of computer vision tasks, especially for image classification [2,3]. The successful application of CNNs in many fields is largely attributed to various optimization algorithms [4–7] and various network structures [8–11], in addition to the development of GPU. Among them, batch normalization [12] is also one of the main contributions to the development of CNNs.

It is well known that convolutional neural networks are very sensitive to the initialization of convolutional kernels, which determines whether CNNs can converge. Each convolutional layer of CNNs is composed of multiple convolutional kernels, activation functions, etc., and the convolutional layer is calculated progressively. Obviously, the training process of CNNs can be regarded as a process of numerical iterative calculation. Therefore, it is necessary to analyze the convergence of CNNs training process in order to ensure network convergence.

The existing initialization methods have obtained good results on many network structures. However, the convergence of CNNs is hardly analyzed in these literatures, so these initialization methods cannot be applied to different network structures. For example, the

* Corresponding author at: Institute of Information Science and Technology, Yanshan University, Qinhuang Dao, 066004, Hebei province, China.
E-mail addresses: 927433441@qq.com (H. Zhang), 851846513@qq.com (L. Feng), xiaohua@126.com (X. Zhang), 43537195@qq.com (Y. Yang), 421320610@qq.com (J. Li).

https://doi.org/10.1016/j.dsp.2022.103397

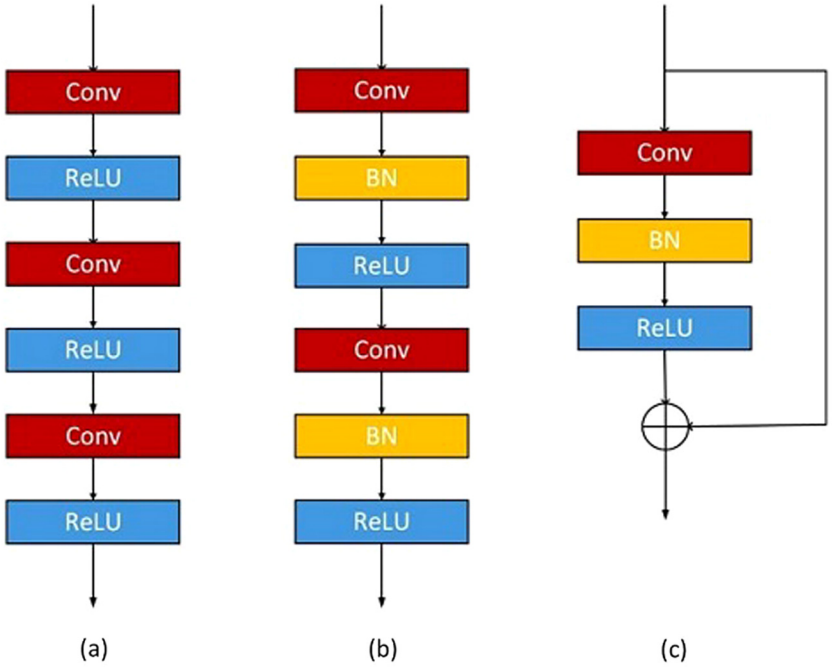

Fig. 1. Diagram of the plain CNNs without BN architecture (a) and the non-plain CNNs with BN architecture (b and c) in this paper.

Kaiming method [13] is used to initialize the convolution kernel on the ResNet network without BN, which causes the network to fail to converge [14].

In this work, the convergence of CNNs is analyzed in the linear normed space. The forward or backward calculation output of each layer of CNNs can be regarded as a linear normed space vector, and the layer-to-layer forward and back propagation calculation of the network can be regarded as the mapping of two linear normed Spaces. During the network iterative training process, it is ensured that the forward output vector norm and the reverse output error vector norm of the network layer remain stable, that is, the two norms neither tend to infinity nor to 0 during the propagation process, which is a necessary condition for the training and learning of CNNs to converge stably. The divergence of the forward vector norm and the reverse error vector norm is the cause of the gradient explosion, and they tend to 0 during the propagation process, which is the cause of the gradient disappearance. The CNNs without skip connections and without BN are called as plain network (as in Fig. 1, the structure diagram of plain networks and non-plain networks). According to the requirements for the necessary conditions of convergence of CNNs, the initialization method of plain network is derived, and derive the initialization methods whose activation functions are ReLU [15] and PReLU [13].

The batch normalization (BN) alleviates the problem of gradient disappearance and explosion to some extent, speeds up the convergence of network training, allows the training of deeper neural networks, and prevents overfitting. It also allows the network to use a larger learning rate and reduces the requirement for tuning parameters [12]. Batch normalization has achieved great empirical success for training deep networks, but now there is still no general consensus on why BN is helpful to the training process of CNNs [14], and the effectiveness of the BN is still not well understood by scholars [16], and its mathematical explanation has been the subject of research by scholars. Intrigued by this topic, according to the necessary conditions for convergence of CNNs proposed in this work, the mathematical reasons why BN is helpful to train CNNs can be explained. At the same time, it is found that the Kaiming initialization method is not the best choice for the networks with BN due to the influence of the value of initial learning rate, which leads to the derivation of the initialization method of networks with BN layer.

Finally, on the CIFAR-10 and CIFAR-100 dataset [17] in the field of image classification, we use four network models (VGG-19, ResNet-110, DenseNet-100 and WideResNet28-10) to test ours initialization method. In the case that the network structure is exactly the same as other optimization algorithms, our initialization method makes the convergence speed of CNNs faster, and the test results are better than other initialization methods, and even surpass other optimization algorithms. Our main contributions are as follows:

(1) The necessary conditions for the convergence of CNNs are proposed in linear normed space.

(2) The initialization method of the convolution kernel of the plain network based on the necessary conditions for convergence of CNNs is derived.

(3) The mathematical reasons why BN contribute to the training of CNNs can be explained according to the necessary conditions for convergence of CNNs.

(4) We found that the learning rate and initialization of networks with BN layer jointly affect the convergence of the networks, so we derived the initialization method corresponding to the size of the learning rate value on the basis of initialization method of the plain network.

The remainder of paper is organized as follows. In Section 2, recent work related to the initialization method of the convolution kernel is reviewed. We propose the necessary conditions for the convergence of CNNs in the linear normed space in Section 3. According to the necessary conditions for convergence of CNNs, we derive methods for plain networks whose activation functions are ReLU and PReLU are derived in section 3. We derive methods for CNNs with BN in Section 4. Section 5 shows the experimental settings and the results and comparisons with other Initialization method. The conclusion of our paper and future work are discussed in Section 6.

2. Related work

The convergence of CNNs is mainly determined by the joint action of several factors such as the learning rate, the initialization of the convolutional kernel, the activation function, the optimization algorithm of the loss function, and the optimization algorithm of the corresponding network model. The optimization algorithms of loss function include SGD (Stochastic Gradient Descent), Adam [5] and other algorithms. Arjovsky [18] improved the optimization algorithm of GAN network [19], called WGAN network, including removing the last layer of activation function and modifying the loss function to completely solve the convergence problem of GAN network. Based on the Lipschitz constant theory of deep learning architectures, Scaman and Virmaux [7] proposed an optimization algorithm to improve the network convergence performance. The initialization of the convolution kernel is an important factor in the convergence of CNNs, as is the optimization algorithm.

Nowadays, deep CNNs are mainly initialized with Gaussian distributions to randomly initialize the convolution kernels. The convolution kernel is randomly initialized by a Gaussian distribution with a standard deviation of 0.01 [2], but this method lacks mathematical theoretical derivation. In experiments, it was found that it is difficult to converge for CNNs with more than 8 layers, as reported by the VGG team [3]. Meanwhile, it can be obtained from our initialization theory that this initialization method does not converge for deep networks.

Glorot and Bengio [20] has proposed an initialization method for fully connected neural networks, which was called Xavier initialization method by [21] and has gradually gained the attention of scholars. It is an initialization method for uniform distribution of the weight of the convolution kernel, and the range of the value is determined by the input and output dimension of the layer. Xavier initialization method is derived on the premise that the activation function is a linear function, namely, identity mapping. However, the activation function of the convolutional neural network is a non-linear function, so the Xavier initialization method does not make consistent the data variance of each layer of the convolutional neural network, which has some effect on the convergence performance of the convolutional neural network, but the effect is limited. Nevertheless, Xavier initialization method for the first time proposed to initialize the convolutional kernel of the network under the premise that the variance of all layers should be consistent during the calculation of CNNs' forward propagation.

Based on the theory of Xavier initialization method, He [13] proposed the Kaiming initialization method for ReLU activation function and PReLU activation function. Greatly accelerating the convergence speed of the network and improving the performance of the network convergence, Kaiming initialization method became the most dominant initialization method for CNNs nowadays. Zhang [14] points out Kaiming initialization method is suitable for plain networks without skip connections (such as VGG networks), while for the networks with skip connections and without BN (e.g. ResNet without BN), it causes the lower bound on the gradient norm of some layers to grow indefinitely with increasing network depth, resulting in a gradient explosion, and a fixup initialization method is proposed to ensure that the update of the convolution kernel is kept within a certain range by adjusting the Kaiming initialization method. The initialization method proposed in [22] is similar to fixup initialization method. Brock [23,24] propose the method of Adaptive Gradient Clipping (AGC), which clip the gradient based on the unit scale of gradient norms and parameter norms. The common feature of these methods is that they scale the convolution kernel of each layer of the network, so that the variance of each layer of the network does not increase with the depth of the network, thus avoiding the occurrence of gradient explosion.

The Orthonorm initialization method put forward by [25] regards fully connected neural networks as nonlinear dynamic systems and proposes an initialization method of orthogonal matrix, and for the first time proposes the Dynamical Isometry theory. Using this theory, [26,27] provide a more detailed explanation of the forward and back propagation calculations of the initialization method. [28] proposed to extend the Orthonorm initialization method to the iterative process to adjust the convolution kernel of each layer: replace the weights with Gaussian distribution of unit variance, and then decompose them into orthonormal basis with QR or SVD decomposition, and replace the weights with one of the components. By virtue of Mean Field Theory, Dynamical Isometry theory, and the conditions for singular value equilibrium of input-output Jacobi Matrix, [29] proposes delta-orthogonal Initialization, which can train ten thousand layers of CNNs or even more.

Dauphin and Schoenholz [30] proposes an automatic initialization method called MetaInit. By virtue of the descent of iterative training gradient to adjust the norm of the initial weight matrix, the weight norm was minimized to initialize the weight. Zhu [31] made a big improvement on the MetaInit initialization method by proposing the GradInit initialization method, which scaled the weight parameters through iteratively training and learning the scalar coefficients. This set of scalar coefficients was optimized to adjust the variance of each network layer, so that each step of SGD or Adam could produce the smallest possible loss value, while preventing the explosion of the initial gradient norm.

These initialization methods do not analyze the impact on the convergence of CNNs. In this work, we first propose the necessary conditions for the convergence of CNNs and derive the initialization methods for convolutional kernels when the necessary conditions for convergence are satisfied.

3. Necessary conditions for convergence of CNNs

The forward propagation of convolutional neural network is calculated to the output layer through iterative computation layer by layer. Each layer is a vector space (linear space), and is obviously a linear normed space, too. Suppose the convolution layer of CNNs has n layers, the output of Layer l is $\mathbf{y}_l \in R^{B\times C\times H\times W}$ $(l = 1, 2, \cdots, n)$. The output of the preceding layer is $\mathbf{y}_{l-1}$. Let the calculation of convolution kernel, bias value and activation function from the Layer $l-1$ to the Layer l be the mapping of these two vector Spaces, T_l, then we get:

$$\mathbf{y}_l = T_l\mathbf{y}_{l-1} \tag{1}$$

Its norm is:

$$\|\mathbf{y}_l\| = \|T_l\mathbf{y}_{l-1}\| \le \|T_l\|\|\mathbf{y}_{l-1}\| \tag{2}$$

It is obvious that the mappings T_l are bounded operators. Therefore, there is a supremum which is set to $M_l \ge 0 (l = 1, 2, \cdots, n)$, such that

$$\|\mathbf{y}_l\| \le \|T_l\|\|\mathbf{y}_{l-1}\| \le M_l\|\mathbf{y}_{l-1}\|. \tag{3}$$

Since the mapping T_l is continuously differentiable, there must be a real number $0 \le \alpha_l \le M_l (l = 1, 2, \cdots, n)$ such that

$$\|\mathbf{y}_l\| = \alpha_l\|\mathbf{y}_{l-1}\|. \tag{4}$$

Let the data of the input layer be $\mathbf{x}_0$. After successive iterative calculation, by the nth layer, we can get

$$\|\mathbf{y}_l\| = \prod_{l=1}^{n} \alpha_l\|\mathbf{x}_0\|. \tag{5}$$

When n is relatively large, it can be concluded from Eqn. (5) that if $\alpha_l > 1(l = 1, 2, \cdots, n)$ is true, $\|\mathbf{y}_l\|$ will be amplified layer by layer in the forward propagation process, and $\|\mathbf{y}_n\|$ will diverge, thus the network will not converge, resulting in an explosion of the gradient. When $\alpha_l < 1(l = 1, 2, \cdots, n)$ is true, $\|\mathbf{y}_n\|$ tends to 0, and the gradient disappears accordingly.

Obviously, when $\alpha_l = 1(l = 1, 2, \cdots, n)$ is true, the forward propagation vector norm of each layer of the network is equal, which is a necessary condition for the convergence of convolutional neural network. If there are only a few $\alpha_l \neq 1$, this does not cause $\|\mathbf{y}_n\|$ of the forward propagation to diverge or converge to zero, and therefore do not cause the gradient to explode or disappear.

For backpropagation, let the error of the output layer be δ, the derived function of the activation function and the convolution kernel operation be the Mapping T_l', and the error of the convolution layer be $\delta_l(l = 1, 2, \cdots, n)$. The calculation process of backward error propagation from Layer l to Layer $l-1$, namely, the mapping of two vector paces δ_l and δ_{l-1}, is denoted as:

$$\delta_{l-l} = T_l'\delta_l. \tag{6}$$

Similarly, the mapping T_l' is a continuous bounded operator. Therefore, it has a supremum which is set to $N_l \geq 0(l = 1, 2, \cdots, n)$. Then Eqn. (6) can be described as follows:

$$\|\delta_{l-l}\| = \|T_l'\delta_l\| \leq \|T_l{}'\|\|\delta_l\| \leq N_l\|\delta_l\|. \tag{7}$$

Since the mapping T_l' is continuously differentiable, there must be a real number $0 \leq \beta_l \leq N_l(l = 1, 2, \cdots, n)$ such that

$$\|\delta_{l-l}\| = \beta_l\|\delta_l\|. \tag{8}$$

After the reverse iteration calculation layer by layer, by the first layer, the error will be as follows:

$$\|\delta_1\| = \prod_{l=n}^{1}\beta_l\|\delta\|. \tag{9}$$

When n is relatively large, if $\beta_l > 1(l = 1, 2, \cdots, n)$ is true, the error vector δ_l in the back propagation process of the network is amplified layer by layer and $\|\delta_1\|$ will diverge. As a result, there will be a gradient explosion, and the network will not converge; If $\beta_l < 1(l = 1, 2, \cdots, n)$ is true, and $\|\delta_1\|$ tends to 0, the gradient disappears accordingly.

If $\beta_l = 1(l = 1, 2, \cdots, n)$ is true, the error norm of back propagation each layer of the network will be equal, which is a necessary condition for the convergence of convolutional neural network. Only a few numbers that make $\beta_l \neq 1$ is true will not cause $\|\delta_1\|$ of backpropagation to diverge or approach zero, and therefore does not cause the gradient to explode or disappear.

Based on the analysis of CNNs' convergence above, it can be concluded that the necessary conditions for convergence CNNs are as follows: in the learning and training process of CNNs, the forward propagation vector norm will neither diverge nor tend to 0, and the backpropagation error vector norm will neither diverge nor approach 0.

Whether the forward propagation vector norm and the backpropagation error vector norm of the first iteration of CNNs, namely, $\|\mathbf{y}_n\|$ and $\|\delta_1\|$, can converge stably determines whether the network can converge. Obviously, the initialization of weight parameters is very important, and so is the update of weight parameters.

We set that the learning rate is η, and the vector of convolution kernel of the layer l is $\mathbf{w}_l(l = 1, 2, \cdots, n)$, and $\mathbf{w}_l^u$ is the updated convolution kernel, then we can get a equation as follows:

$$\mathbf{w}_l^u = \mathbf{w}_l - \eta\mathbf{y}_l\delta_{l+1}. \tag{10}$$

According to Eqn. (10), the learning rate η determines whether the network can converge stably. A larger learning rate causes $\mathbf{w}_l(l = 1, 2, \cdots, n)$ to diverge. As a result, the forward propagation vector norm of the subsequent iterative calculation of the network will diverge, and gradient explosion will occur, resulting in non-convergence of the network.

From the above analysis, we can see that a good initialization and a good learning rate are two important factors to meet the requirements for the convergence of CNNs.

4. Initialization of convolution kernel

In this section, we first derive the Gaussian distribution initialization formula of the forward propagation process of the plain network, and then derive the Gaussian distribution initialization method of the back propagation error process in the same way, and finally derive the Gaussian distribution initialization method for the network with BN on the basis of the plain network initialization method.

The weight parameters of CNNs are randomly initialized, so the output of each layer is a random vector. If the mean and variance of the forward propagating vector are equal at each layer, $E(\mathbf{y}_{l-1}) = E(\mathbf{y}_l)$ and $Var(\mathbf{y}_{l-1}) = Var(\mathbf{y}_l)$, then we can have: $\|\mathbf{y}_l\| = \|\mathbf{y}_{l-1}\|$. In other words, in Eqn. (5), the forward propagation vector norm $\|\mathbf{y}_n\|$ will converge stably.

In the process of backpropagation, if the mean and variance of the vector of backpropagation error of each layer are equal, $E(\delta_{l-1}) = E(\delta_l)$ and $Var(\delta_{l-1}) = Var(\delta_l)$, then we can have an equation like this $\|\delta_l\| = \|\delta_{l-1}\|$. That is to say, $\|\beta_l\| = 1$ in Eqn. (9). This can make the vector norm of backpropagation error converge stably.

If the mean and variance of the forward propagation vector and the backpropagation error vector of each layer are equal, the necessary conditions for the convergence of CNNs are satisfied.

4.1. The Gaussian distribution initialization method of forward propagation process

The mapping $T_l(l = 1, 2, \cdots, n)$ is composed of the computation of convolution kernel, the bias value and the activation function. Since the activation function is deterministic, the fact that only the convolution kernel and bias values can be properly initialized makes this true: in Eqn. (5), $\alpha_l = 1(l = 1, 2, \cdots, n)$.

We let the vector before the layer l was inputted into the activation function be $\mathbf{x}_l(l = 2, 3, \cdots, n)$, $\mathbf{b}_l$ is the bias value, and the activation function is $\varphi()$. Obviously, the output of each layer is $\mathbf{y}_l = \varphi(\mathbf{x}_l)$. For $\mathbf{x}_l$ of each layer, there exists a functional relationship as follows:

$$\mathbf{x}_l = \mathbf{w}_l\mathbf{y}_{l-1} + \mathbf{b}_l = \mathbf{w}_l\varphi(\mathbf{x}_{l-1}) + \mathbf{b}_l, \tag{11}$$

where, we let the input channels of layer l be c_l (equivalent to the output channels of layer $l-1$), and the convolution kernel in the layer l be the Matrix $k_l \times k_l$. Any point x_l of vector $\mathbf{x}_l$ can be regarded as scalars obtained from the inner product operation of both the one-dimensional row vector $\mathbf{w}$ constituted by $c_l \times k_l^2$ scalars on vector $\mathbf{w}_l$ and the one-dimensional column vector $\mathbf{y}$ constituted by $c_l \times k_l^2$ scalars on the output of previous layer $\mathbf{y}_{l-1}$. The equation is as follows:

$$x_l = \mathbf{w}\mathbf{y} + \mathbf{b} = \sum_{i=1}^{c_l\times k_l^2} w_i \times y_i = \sum_{i=1}^{c_l\times k_l^2} w_i y_i \varphi(x_{l-1,i}) + b, \tag{12}$$

where $x_{l-1,i}$ is a scalar of the Vector $\mathbf{x}_{l-1}$ inputted into the activation function for calculation by layer $l-1$. Let $n_l = c_l \times k_l^2$. Considering b acts as a deviation to the mean value of the whole data, let it be initialized to 0 for sake of research purposes. Because $\mathbf{w}_l$ and the output of the previous layer $\mathbf{y}_{l-1}$ are independent of each other, there are:

$$E(x_l) = E(\sum_{i=1}^{c_l \times k_l^2} w_i \varphi(x_{l-1,i})) = n_l E(w_i) E(\varphi(x_{l-1,i})), \tag{13}$$

$$Var(x_l) = Var(\sum_{i=1}^{c_l \times k_l^2} w_i \varphi(x_{l-1,i})) = n_l Var(w_i \varphi(x_{l-1,i})), \tag{14}$$

where w_i is a scalar of $\mathbf{w}_l$, so w_i can be regarded as an individual from the population $\mathbf{w}_l$. As a result, the scalar w_i shares the same mean and variance with $\mathbf{w}_l$, and $x_{l-1,i}$ as well as $\mathbf{x}_{l-1}$ also have the same mean and variance for the same reason. So Eqn. (13) can also be expressed as:

$$E(\mathbf{x}_l) = n_l E(\mathbf{w}_l) E(\varphi(\mathbf{x}_{l-1})). \tag{15}$$

It is easy to conclude from Eqn. (15) that if the mean value $E(\mathbf{w}_l)$ of the convolution Kernel $\mathbf{w}_l$ does not equal 0, then the mean value of the forward propagation vector at each layer will diverge, causing gradient explosion. Therefore, $E(\mathbf{w}_l) = 0 (l = 1, 2, \cdots, n)$ is a necessary condition for CNNs' initialization.

When $E(\mathbf{w}_l) = 0$, it is obvious that $E(\mathbf{w}_l \varphi(\mathbf{x}_{l-1})) = 0$ is true. By the calculation formula of variance based on probability theory, we can have the equation as follows:

$$Var(\mathbf{w}_l \varphi(\mathbf{x}_{l-1})) = E(\mathbf{w}_l^2 [\varphi(\mathbf{x}_{l-1})]^2). \tag{16}$$

Because $\mathbf{w}_l$ and the output of the previous layer $\mathbf{y}_{l-1}$ are independent of each other, Eqn. (16) can be expressed as:

$$Var(\mathbf{w}_l \varphi(\mathbf{x}_{l-1})) = E(\mathbf{w}_l^2) E([\varphi(\mathbf{x}_{l-1})]^2). \tag{17}$$

Since $E(\mathbf{w}_l^2) = Var(\mathbf{w}_l)$, Eqn. (17) can be expressed as:

$$Var(\mathbf{x}_l) = n_l Var(\mathbf{w}_l) E([\varphi(\mathbf{x}_{l-1})]^2). \tag{18}$$

Since there is no activation function in the input layer of the forward propagation process (layer 1), Eqn. (18) in the first layer of the convolution layer is as follows:

$$Var(\mathbf{x}_1) = n_1 Var(\mathbf{w}_1) E([\varphi(\mathbf{x}_0)]^2). \tag{19}$$

$\mathbf{x}_0$ is the training data set, and both its mean value and variance can be obtained. Let $Var(\mathbf{x}_0) = \sigma^2$ and $E(\mathbf{x}_0) = \mu$, and let $\hat{\mathbf{x}}_0 = \mathbf{x}_0 - \mu$ as a linear variation of $\mathbf{x}$. We can get $E(\hat{\mathbf{x}}_0) = 0$, and $Var(\hat{\mathbf{x}}_0) = E(\hat{\mathbf{x}}_0^2) = \sigma^2$. Thus, the Eqn. (19) can be transformed into:

$$Var(\mathbf{x}_1) = n_1 Var(\mathbf{w}_1) E([\varphi(\hat{\mathbf{x}}_0)]^2). \tag{20}$$

Let $Var(\mathbf{w}_1) = \frac{1}{n_1}$, then $Var(\mathbf{x}_1) = E(\hat{\mathbf{x}}_0^2) = Var(\mathbf{x}_0) = \sigma^2$ can be obtained and the initialization equation of Gaussian distribution is as follows:

$$\mathbf{w}_1 \sim N(0, \frac{1}{n_1}). \tag{21}$$

When the first layer of the convolution layer is initialized according to Eqn. (21), and the variance of $\mathbf{x}_1$ can be calculated, what can be obtained from Eqn. (18) is $Var(\mathbf{x}_2) = n_2 Var(\mathbf{w}_2) Var(\mathbf{y}_1) = n_2 Var(\mathbf{w}_2) E([\varphi(\mathbf{x}_1)]^2)$. $E([\varphi(\mathbf{x}_1)]^2)$ is unknown value, so let $E([\varphi(\mathbf{x}_1)]^2) = \lambda_2 Var(\mathbf{x}_1)$, in which λ_2 is an unknown real number. Then we have $Var(\mathbf{x}_1) = \lambda_2 n_2 Var(\mathbf{w}_2) Var(\mathbf{x}_1)$.

Obviously, let $Var(\mathbf{w}_2) = \frac{1}{\lambda_2 n_2}$, there is $Var(\mathbf{x}_2) = Var(\mathbf{x}_1)$. In the same way, let $Var(\mathbf{w}_l) = \frac{1}{\lambda_l n_l}$, we have $Var(\mathbf{x}_{l+1}) = \lambda_l n_l Var(\mathbf{w}_l) Var(\mathbf{x}_l) (l = 2, 3, \cdots, n)$. Let $Var(\mathbf{w}_l) = \frac{1}{\lambda_l n_l}$, we have $Var(\mathbf{x}_{l+1}) = Var(\mathbf{x}_l)$. If we do the recursion to the last level of the convolution layer in this way, there is $Var(\mathbf{x}_n) = Var(\mathbf{x}_1)$. Thus, it meets the necessary convergence conditions for the forward propagation calculation of CNNs.

According to the analysis above, we initialize the convolution Kernel Vectors $\mathbf{w}_l (l = 1, 2, \cdots, n)$ with zero-mean Gaussian distribution whose variance is $Var(\mathbf{w}_l) = \frac{1}{\lambda_l n_l}$. The equation is as follows:

$$\mathbf{w}_l \sim N(0, \frac{1}{\lambda_l n_1}), \tag{22}$$

where $\lambda_l = \frac{Var(\mathbf{x}_l)}{E([\varphi(\mathbf{x}_l)]^2)} (l = 2, 3, \cdots, n)$ is an unknown parameter whose value is determined by the specific form of the activation function.

When the activation function $\varphi()$ is the ReLU function, then $\varphi(\mathbf{x}_l)$ is the vector composed of those scalars greater than 0 in $\mathbf{x}_l$, and the probability of each scalar being 0.5 is greater than 0, and $E(\mathbf{w}_l) = 0$, then we can get:

$$E([\varphi(\mathbf{x}_1)]^2) = Var(\mathbf{y}_l) = \frac{1}{2} Var(\mathbf{x}_l). \tag{23}$$

According to Eqn. (23), we can get $\lambda_l = \frac{1}{2} (l = 2, 3, \cdots, n)$ and the forward propagation initialization equation of the Gaussian distribution of $\mathbf{w}_1 (l = 2, 3, \cdots, n)$, which is as follows:

$$\mathbf{w}_l \sim N(0, \frac{2}{n_l}). \tag{24}$$

When the activation function is a PReLU function, in the same way, $\lambda_l = 1 + \alpha^2$ can be obtained. Given α is the slope of the PReLU function, the initialization equation of Gaussian distribution of $\mathbf{w}_l (l = 2, 3, \cdots, n)$ is as follows:

$$\mathbf{w}_l \sim N(0, \frac{1 + \alpha^2}{n_l}). \tag{25}$$

When the activation functions are ReLU and PRelu, the initialization Eqn. (24) and Eqn. (25) are exactly the same as the Kaiming initialization equation of [13]. When the activation function is Sigmoid or Tanh, λ_l is not a constant, and the value of λ_l for each iteration computation is also different.

4.2. The Gaussian distribution initialization formula of backward propagation error

In the backpropagation process of the error vector, let $\varphi'()$ be the derived function of the activation function $\varphi()$, and $\delta_l (l = 1, 2, \cdots, n)$ is the error vector of layer l, and the convolution kernel of layer l is a matrix of $k_l \times k_l$, and c_l is the number of input channels of back propagation at layer $l-1$ and is also the number of output channels of forward propagation at layer l. Any scalar ξ_{l-1} on the error vector δ_{l-1} of the layer l-1 can be regarded as a one-dimensional row vector $\mathbf{w}$ composed of $c_l \times k_{l-1}^2$ scalars on the convolution kernel vector $\mathbf{w}_{l-1}$ of layer $l-1$ and the scalar obtained by the inner product operation of the one-dimensional column vector $\varphi'(\delta)$ composed of $c_l \times k_{l-1}^2$ scalars on the error vector δ_l of the layer l. The formula is as follows:

$$\xi_{l-1} = \mathbf{w} \varphi'(\delta) = \sum_{i=1}^{c_l \times k_{l-1}^2} w_i \times \varphi'(\xi_{l,i}), \tag{26}$$

were $\xi_{l,i}$ is a scalar of the error vector δ_l of the layer l. Let $m_l = c_l \times k_{l-1}^2$, and since the convolution kernel $\mathbf{w}_{l-1}$ and the error vector δ_l are independent of each other, there are two equations as follows:

$$E(\xi_{l-1}) = E(\sum_{i=1}^{c_l \times k_{l-1}^2} w_i \varphi'(\xi_{l,i})) = m_l E(w_i) E(\varphi'(\xi_{l,i})), \tag{27}$$

$$Var(\xi_{l-1}) = Var(\sum_{i=1}^{c_l \times k_{l-1}^2} w_i \varphi'(\xi_{l,i})) = m_l Var(w_i \varphi'(\xi_{l,i})), \tag{28}$$

where w_i is a scalar of $\mathbf{w}_{l-1}$. It can be considered as an individual from the population $\mathbf{w}_{l-1}$, so the scalars w_i and $\mathbf{w}_{l-1}$ have the same mean and variance. Similarly, $\xi_{l,i}$ and δ_l have the same mean and variance, $\xi_{l-1,i}$ and δ_{l-1} have the same mean and variance, too. So, Eqn. (27) can be expressed as:

$$E(\delta_{l-1}) = E(\sum_{i=1}^{c_l \times k_{l-1}^2} \mathbf{w}_{l-1} \varphi'(\delta_l)) = m_l E(\mathbf{w}_l) E(\varphi'(\delta_l)). \tag{29}$$

Eqn. (28) can be expressed as:

$$Var(\delta_{l-1}) = m_l Var(\mathbf{w}_{l-1} \varphi'(\delta_l)). \tag{30}$$

Since $E(\mathbf{w}_{l-1}) = 0$, it can be derived from Eqn. (29) that $E(\delta_{l-1}) = 0 (l = n, n-1, \cdots, 2)$, and according to the variance formula in probability theory, we can get an equation like this:

$$\begin{aligned} Var(\mathbf{w}_{l-1} \varphi'(\delta_l)) &= E(\mathbf{w}_{l-1}^2 [\varphi'(\delta_l)]^2) - [E(\mathbf{w}_{l-1} \varphi'(\delta_l))]^2 \\ &= E(\mathbf{w}_{l-1}^2 [\varphi'(\delta_l)]^2). \end{aligned} \tag{31}$$

Since $\mathbf{w}_{l-1}$ and the error vector output δ_l are independent of each other, Eqn. (31) can be expressed as:

$$Var(\mathbf{w}_{l-1} \varphi'(\delta_l)) = E(\mathbf{w}_{l-1}^2) E([\varphi'(\delta_l)]^2). \tag{32}$$

Because $E(\mathbf{w}_l) = 0$, there is $E(\mathbf{w}_l^2) = Var(\mathbf{w}_l)$. According to Eqn. (31), the following equation can be obtained:

$$Var(\delta_{l-1}) = m_l Var(\mathbf{w}_{l-1}) E([\varphi'(\delta_l)]^2). \tag{33}$$

Let $E([\varphi'(\delta_n)]^2) = \lambda'_{n-1} Var(\delta_n)$, and λ'_{n-1} is an unknown real number, then $Var(\delta_{n-1}) = \lambda'_{n-1} m_n Var(\mathbf{w}_{n-1}) Var(\delta_n)$ can be obtained. Let $Var(\mathbf{w}_{n-1}) = \frac{1}{\lambda'_n m_n}$, then $Var(\delta_{n-1}) = Var(\delta_n)$ can be obtained. By the same recurrence, let $E([\varphi'(\delta_l)]^2) = \lambda'_{l-1} Var(\delta_l)$, then according to Eqn. (33), there is $Var(\delta_{l-1}) = \lambda'_{l-1} m_n Var(\mathbf{w}_{l-1}) Var(\delta_l) (l = n, n-1, \cdots, 2)$. We let $Var(\mathbf{w}_l) = \frac{1}{\lambda'_l m_{l+1}}$, then $Var(\delta_{l-1}) = Var(\delta_l)$ can be obtained. In this way, let the back propagation go to the first level of the convolution layer. Thus, $Var(\delta_1) = Var(\delta_n)$ can be obtained. Obviously, the error vector norm neither diverges nor tends to 0, which meets the requirements for the convergence of back propagation CNNs.

According to the analysis above, we initialize the convolution kernel vectors $\mathbf{w}_l (l = 1, 2, \cdots, n)$ with zero-mean Gaussian distribution whose variance is $Var(\mathbf{w}_l) = \frac{1}{\lambda'_l m_{l+1}}$. The equation is as follows:

$$\mathbf{w}_l \sim N(0, \frac{1}{\lambda'_l m_{l+1}}), \tag{34}$$

where $\lambda'_l = \frac{Var(\delta_l)}{E([\varphi'(\delta_l)]^2)}$ is an unknown parameter, whose value is determined by the specific form of the activation function.

When the activation function $\varphi()$ is ReLU, the derivative function of ReLu turns the scalar of the vector δ_l to 0 with a probability of 0.5. Because of $E(\delta_l) = 0$, $\varphi'(\delta_l)$ is also a vector symmetric to zero, such that $E(\varphi'(\delta_l)) = 0$. According to $E([\varphi'(\delta_l)]^2) = Var(\varphi'(\delta_l))$, the following equation can be obtained:

$$E([\varphi'(\delta_l)]^2) = Var(\varphi'(\delta_l)) = \frac{1}{2} Var(\delta_l). \tag{35}$$

It can be concluded that $\lambda'_l = \frac{1}{2} (l = 1, 2, \cdots, n)$, and the back propagation initialization equation of Gaussian distribution of $\mathbf{w}_l$ is as follows:

$$\mathbf{w}_l \sim N(0, \frac{2}{m_{l+1}}). \tag{36}$$

When the activation function is PRelu, we can get $\lambda'_l = \frac{1}{1+\alpha^2}$ for similar reason. The initialization formula of Gaussian distribution of the backpropagation of $\mathbf{w}_l (l = 1, 2, \cdots, n)$ is as follows:

$$\mathbf{w}_l \sim N(0, \frac{1+\alpha^2}{m_{l+1}}). \tag{37}$$

When the activation function is Sigmoid, its derivative function is $\varphi'(x) = \varphi(x)(1 - \varphi(x))$, and the value range is $(0, 0.25]$. In its backpropagation calculation, λ'_l of Eqn. (34) is not a constant. With the layered iterative calculation of $\varphi'(x)$, there is $Var(\varphi'(\delta_l)) = 0$, which is why the Sigmoid causes the gradient of the neural network to disappear.

The calculation of the fully connected linear layers can be regarded as the convolution calculation of 1×1 convolution kernel, so the initialization equation of the convolution kernel is also applicable to the fully connected linear layers.

4.3. Initialization of the convolution kernel of networks with BN layer

In the current neural network architecture, batch normalization is an important technique that has been widely used to improve model generalization, stabilize the training process, and accelerate model convergence, while BN also makes it possible to train models with higher learning rates. According to the necessary conditions for convergence of CNNs proposed in section 3, the mathematical reasons why BN is helpful in training CNNs can be partially explained.

Owing to BN, the output of each convolutional layer of CNNs is independent of each other, and all the norms of forward propagation vector $\mathbf{y}_l (l = 1, 2, \cdots, n)$ are equal, and furthermore, $Var(\mathbf{y}_l) = 0.5 (l = 1, 2, \cdots, n)$ is true. Even with a larger learning rate and a bad initialization method, forward propagation vector norm of each layer remains unchanged, which meets the requirements for the convergence of forward propagation put forward in Section 3 of this work. As a result, the training process is greatly stabilized and the convergence of the model is accelerated.

The variance of $\mathbf{y}_n$, the output at the last layer with BN, is always 0.5, no matter how many iterations are made. Therefore, the error vector δ_n of this layer is also stable. According to the formula $\delta_{l-1} = \mathbf{w}_{l-1} \varphi'(\delta_l)$, as long as the appropriate activation function is selected and $\|\mathbf{w}_l\|$ does not diverge, the norm of the backpropagation vector δ_l is also stable and does not diverge. Thus, the requirement for the backpropagation convergence proposed in section 2 of this work is easily met.

$\mathbf{y}_l$ and δ_l $(l = 1, 2, \cdots, n)$ are stable. As can be seen from the update Eqn. (10) of convolution kernel, CNNs are still likely to converge with bigger learning rate and bad initialization.

Networks with BN layer are not very sensitive to initialization, but that is not to say that it is not important, as good initialization makes networks with BN layer converge faster and result is better.

Let the vector input to the BN layer is $\mathbf{x}_l$, and let the output vector calculated by BN is $\hat{\mathbf{x}}_l$. For the convenience of description, the relationship between $\hat{\mathbf{x}}_l$ and $\mathbf{x}_l$ is simplified as follows:

$$\hat{\mathbf{x}}_l = \frac{\mathbf{x}_l - \mu}{\sigma}, \tag{38}$$

where μ and σ are mean and standard deviation (std) of the vector calculated by BN layer, respectively. Obviously, $\hat{\mathbf{x}}_l$ is a linear transformation of $\mathbf{x}_l$. Let $\gamma_l = \frac{1}{\sigma}$ and $\beta_l = \frac{-\mu}{\sigma}$, then the relationship between $\hat{\mathbf{x}}_l$ and $\mathbf{x}_l$ can be expressed by the following equation:

$$\hat{\mathbf{x}}_l = \gamma_l \mathbf{x}_l + \beta_l. \tag{39}$$

Obviously, the derivative of CNNs' back propagation calculation in the BN layer is a constant γ_l. Therefore, when the activation function is ReLU, the variance of δ_l is:

$$Var(\delta_{l-1}) = \frac{1}{2}\gamma_l m_l Var(\mathbf{w}_{l-1}) Var(\delta_l). \tag{40}$$

Thus, the backpropagation initialization equation of the Gaussian distribution of $\mathbf{w}_l (l = 1, 2, \cdots, n)$ is:

$$\mathbf{w}_l \sim N(0, \frac{2}{\gamma_l m_{l+1}}), \tag{41}$$

where γ_l is a parameter that needs to be manually set, and its value is set according to the learning rate value.

Let $\mathbf{w}_l^u$ be the updated convolution kernel, then the updated equation of convolution kernel of CNNs with BN is:

$$\mathbf{w}_l^u = \mathbf{w}_l - \eta \nabla \mathbf{w}_l = \mathbf{w}_l - \eta \gamma_l \mathbf{y}_l \delta_{l+1}. \tag{42}$$

If CNNs are initialized by Eqn. (24) or Eqn. (36) with a bigger initial learning rate η, it is possible to make $\|\mathbf{w} - \eta \nabla \mathbf{w}\| > \|W\|$. Thus, $\|\delta_l\| (l = 1, 2, \cdots, n)$ will become oscillatory and unstable, then the training process of the network will be oscillatory and unstable, too. As a result, the convergence speed will slow down.

The oscillation of network caused by bigger learning rate can be alleviated by appropriately scaling the variance of convolution kernel $\mathbf{w}$. According to Eqn. (41), when initializing the convolution kernel, let $\gamma_l < 1$ make $Var(\mathbf{w}_l)$ bigger. At the same time, $\|\eta \nabla \mathbf{w}_l\| = \|\eta \gamma_l \mathbf{y}_l \delta_{l+1}\|$ will become smaller according to Eqn. (42), thus the network training oscillations can be avoided to some extent.

When the initial learning rate η is small, we can appropriately reduce the variance of the convolution kernel $Var(\mathbf{w})$. According to Eqn. (41), let $\gamma_l > 1$ make $Var(\mathbf{w}_l)$ smaller. At the same time, $\|\eta \nabla \mathbf{w}_l\| = \|\eta \gamma_l \mathbf{y}_l \delta_{l+1}\|$ will become bigger according to Eqn. (42). In this way, the convergence speed of the network can be accelerated.

From the above analysis, we can see that general rule of initialization convolution kernel for CNNs with BN: We initialize the convolution kernel of CNNs according to Eqn. (41). When the initial learning rate is bigger, let $\gamma_l < 1$ make $Var(\mathbf{w}_l)$ bigger; and when the initial learning rate is small, let $\gamma_l > 1$ make $Var(\mathbf{w}_l)$ smaller.

5. Experiments

We perform a series of experiments to test our proposed initialization method for networks with BN layer in this section.

5.1. Experimental setup

Dataset We evaluate our initialization method on standard image classification dataset CIFAR-10 and dataset CIFAR-100. CIFAR-10 and CIFAR-100 datasets consist of 32×32 color images drawn from 10 and 100 classes split into 50,000 train and 10,000 test images.

Network Model Four network structures are used to evaluate our initialization method on CIFAR10 and CIFAR100 dataset. They are VGG-19 [3] which has sixteen convolutional layers and one fully connected layer, ResNet [9] which Original width is sixteen and conv layers in each residual block are two, DenseNet100 [10] and WideResNet [11] with Widen Factor 10 (WideResNet28-10).

Learning rate and training epochs In the experiment of the relationship between the initial learning rate and convolution kernel initialization, the initial learning rate we use is 0.1, 0.01, 0.001 respectively. In the test experiment, the initial learning rate is 0.1 and cosine learning rate decay method [32,33] which learning rate decays after each iteration and decays to 0 in the last iteration. Training epochs is 200 or 300.

Weight decay and dropout We use wd=2.0e-4 (weight decay) on the networks of VGG-19, ResNet-110, Densnet100, and we use wd=1.25e-4 on the networks of WideResNet28-10 on CIFAR10 dataset. We use wd=3.0e-4 (weight decay) on the networks of VGG-19, ResNet-110, Densnet100, and we use wd=1.0e-4 on the networks of WideResNet28-10 on CIFAR100 dataset. Weight decay is applied only to the convolution layer and the full connection layer [32,33]. The bias and batch normalization parameters are not used in the weight decay method. We do not use dropout in any of the experiments.

Batch size is 128 in the training learning process, and batch size is 100 in the testing process.

Data augmentation The basic data augmentation is random crop and random left-right flipping. For different experiment, data enhancement methods such as cutout [34], random erasing [35], AutoAugment [36] and Mixup [37] were selectively used.

Besides, stochastic gradient descent with Momentum of 0.9 was used to optimize the algorithm, and Nesterov [38] was used to accelerate the optimizer. The activation function is the ReLU function. We implement the experiment with Python using the Pytorch [39] application program framework. All of experiments were performed on a single NVIDIA GTX-1070Ti GPU with 8 GB.

5.2. Experimental results of the relationship between learning rate and initialization

We adopt VGG-19 network and the ResNet-34 network both with BN. We use the initial learning rate to be 0.1,0.01, and 0.001 respectively, corresponding to these three learning rates, initialize the convolution kernel according to Eqn. (41). The full connected layer is regarded as the convolution calculation of 1×1 convolution kernel. Since the activation function is not used, the full connected layer is initialized according to the forward initialization Eqn. (21) with all bias set as false. The implementation details of them are shown in Fig. 2 to 7.

In Figs. 2–7, the horizontal coordinates indicate the number of epoch. The left vertical coordinate represents the training accuracy, while the right vertical coordinate is the value of the loss. The convolution kernel of the network is initialized according to the backpropagation Eqn. (41), e.g. $Var(\mathbf{w}) = 5/m$ indicates that the parameter $\gamma = 0.4$ is taken according to the backpropagation initialization Eqn. (41).

As can be seen from the above figure, for CNNs with BN, it can be concluded that there is a strong relationship between the initialization method and the initial learning rate, so the Kaiming initialization formula may not be the best choice.

The value of the unknown parameter γ in the initialization Eqn. (41) will be determined by the value of the initial learning rate. If the initial learning rate is bigger, the variance of initialization convolution kernel will be bigger, so γ will be appropriately smaller; if the initial learning rate is small, and the variance of

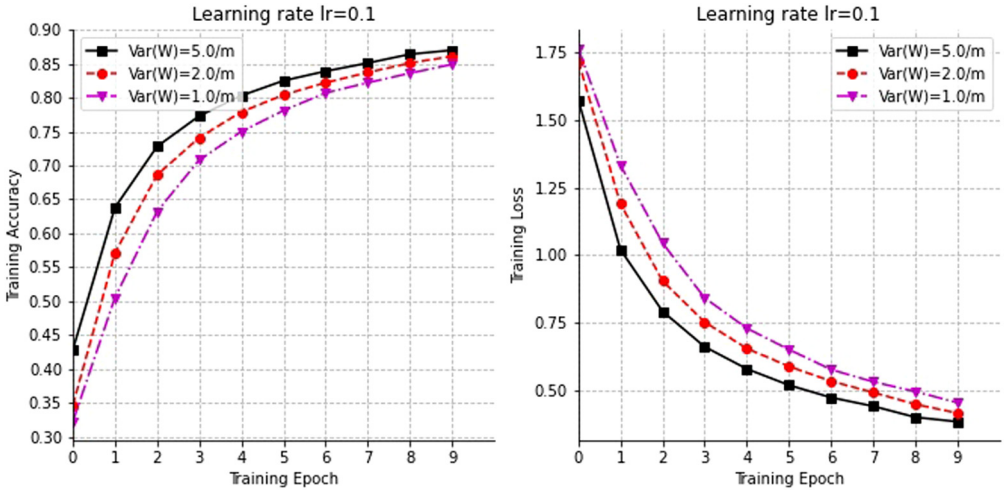

Fig. 2. Training accuracy and loss values for different initializations when lr=0.1 for VGGNet-19 with BN.

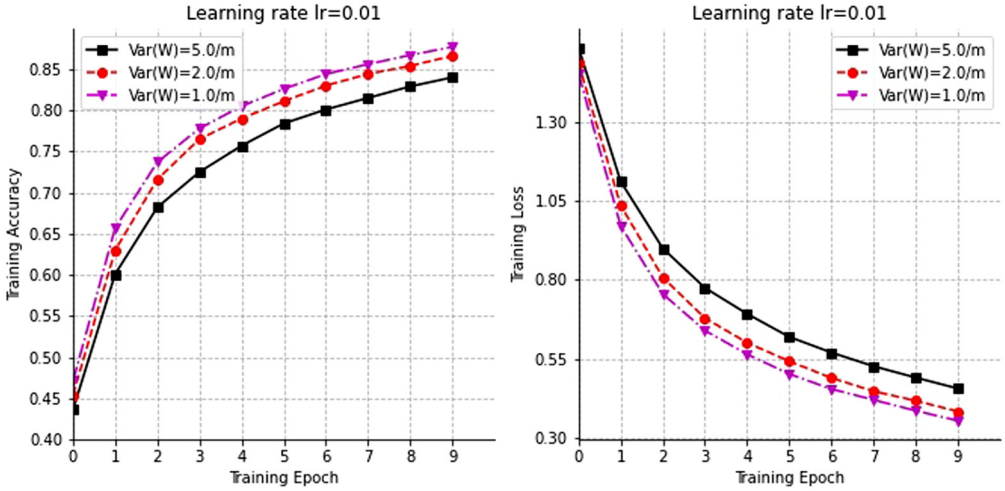

Fig. 3. Training accuracy and loss values for different initializations when lr=0.01 for VGGNet-19 with BN.

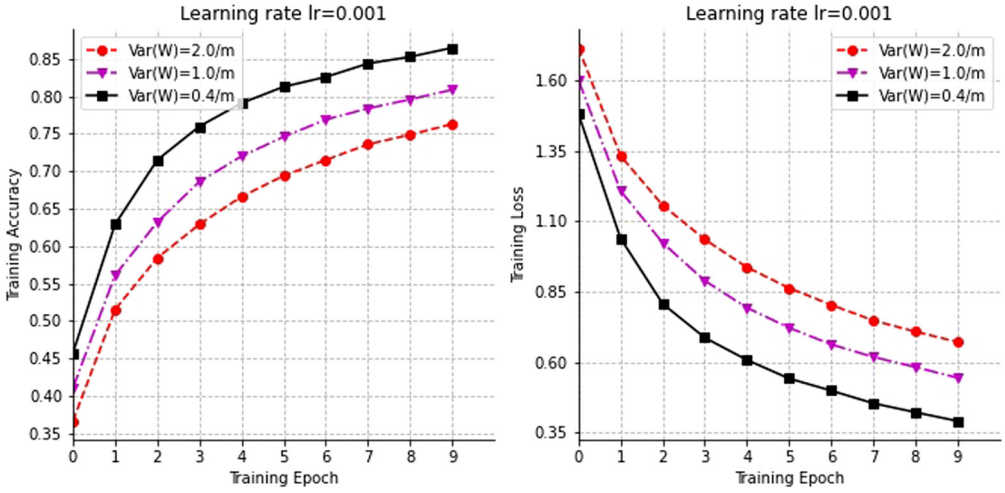

Fig. 4. Training accuracy and loss values for different initializations when lr=0.001 for VGGNet-19 with BN.

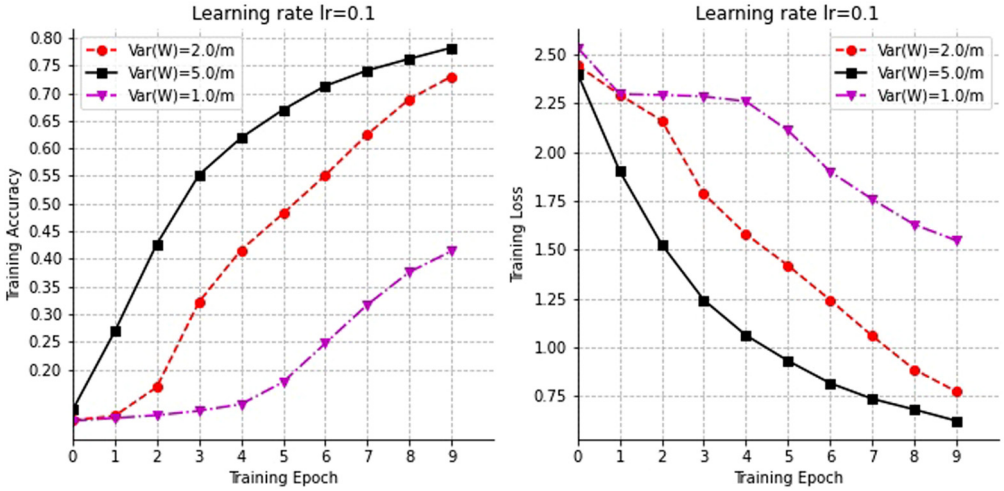

Fig. 5. Training accuracy and loss values for different initializations when lr=0.1 for ResNet-34.

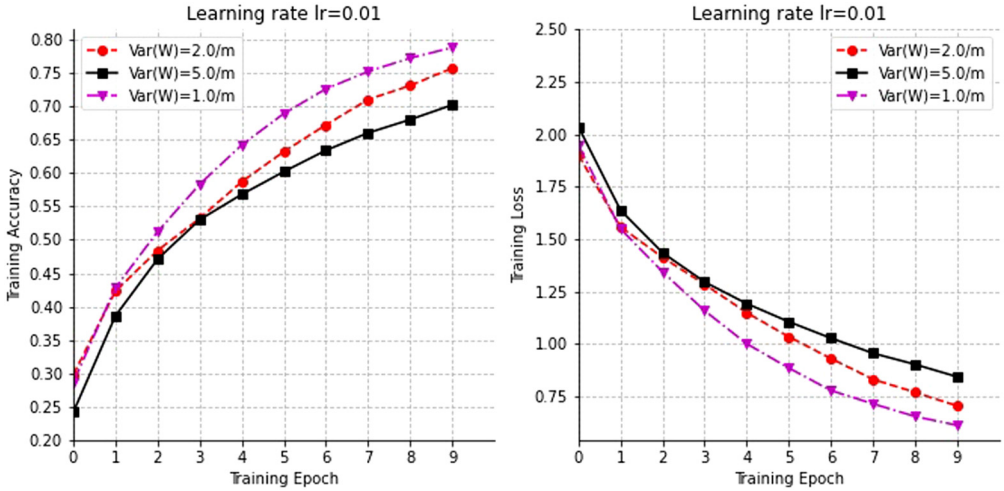

Fig. 6. Training accuracy and loss values for different initializations when lr=0.01 for ResNet-34.

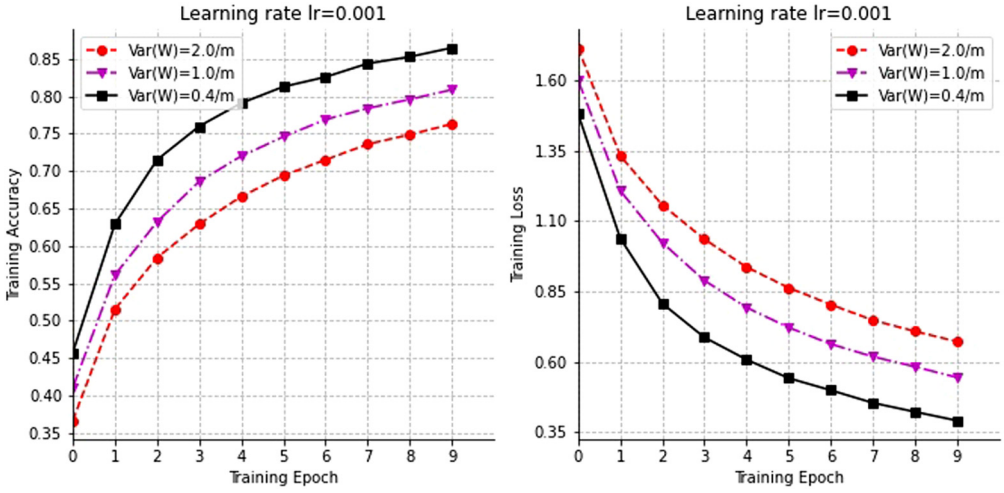

Fig. 7. Training accuracy and loss values for different initializations when lr=0.001 for ResNet-34.

Table 1
Results on CIFAR-10.

Model	Params	Initialization method	Data augmentation	Acc (best) %
VGGNet-19	20.04M	Kaiming	Cutout	94.41
		MetaInit		94.64
		GradInit		95.13
		ours		**95.50**
ResNet-110	1.73M	Kaiming	Cutout	94.98
		MetaInit		94.76
		GradInit		95.38
		ours		**95.58**
DenseNet-100	0.77M	Kaiming	Cutout	95.46
		MetaInit		95.47
		GradInit		95.50
		ours		**96.00**
WideResNet28-10	36.49M	Kaiming	Cutout+Mixup	97.22
		MetaInit		97.10
		GradInit		97.29
		ours		**97.44**

Table 2
Test results comparisons on the CIFAR-10 dataset with the Wideresnet28-10 network.

Approach	Epoch	Acc (best) %
AutoAugment [36] + Cutout	200	97.40
Mixup [37]		97.30
AutoAugment + Cutout + our initialization		**97.47**
[40] AutoAugment + random erasing + Mixup	300	97.72
Manifold Mixup [41]		97.45
ResizeMix [42]		97.60
Cut-MixMo [43] + CutMix [44]		97.73
AutoAugment + random erasing + Mixup + our initialization		**97.79**

initialization the convolution kernel will be smaller, so γ will be appropriately bigger.

Goyal et al. [32] proposed the Warmup method for learning rate preheating: when CNNs are first trained, the convolutional kernels of the network are randomly initialized, and if a large learning rate is chosen, it may bring instability (oscillation) in the network training. Therefore, the solution is to use a smaller learning rate in the first few epochs of training, so that the network can slowly stabilize, and then choose a pre-set large learning rate for training when the network is relatively stable, so that the network converges faster and the network runs better.

The initialization Eqn. (41) can solve the network oscillation problem caused by large learning rate to some extent, playing the role of the Warmup method. At the same time, it can also solve the problem of slow convergence speed caused by small learning rate to some extent.

5.3. Results on CIFAR-10 dataset

The convolution kernel of VGGNet-19, ResNet-110 and Densnet100 were initialized with $Var(\mathbf{w}_l) = \frac{5}{m_{l+1}}$ according to Eqn. (41), and the full-connected layer was initialized according to the forward initialization Eqn. (21). The convolution kernel of WideResNet28-10 network is initialized with $Var(\mathbf{w}_l) = \frac{3}{m_{l+1}}$, and the full-connected layer is initialized according to the forward initialization formula $Var(\mathbf{w}_l) = \frac{0.25}{m_l}$.

On CIFAR-10 dataset, our initialization is compared with other initialization methods (e.g. MetaInit [30], GradInit [31]), as shown in Table 1. The result data comes from [31]. Both MetaInit and GradInit need to initialize the convolution kernel through pre-trained network.

The bold highlights in Table 1 represent the best performance. As can be seen, the test results of our initialization method on the four network models using the same data augmentation method surpass other initialization methods. In the above experiment, we did not use any dropout.

On WideResNet28-10 network model, although some data augmentation methods and other algorithms have achieved very high accuracy, as a result of the test, our initialization method still surpasses these methods in terms of test results, and the results are shown in Table 2.

As can be seen from Table 2, our initialization method achieves a accuracy of 97.79% on the CIFAR-10 dataset with the Wideresnet28-10 network, which exceeds the test results of some other methods. The experimental results also indicate that initialization can achieve significant improvement and mathematical theoretical derivation of the initialization method in this paper is correct.

5.4. Results on CIFAR-100 dataset

On CIFAR-100 dataset, convolution kernel of VGGNet-19 and ResNet-110 were initialized with $Var(\mathbf{w}_l) = \frac{1.5}{m_{l+1}}$, and the full-connected layer was initialized with the constant 0.01, and convolution kernel of Densnet100 was initialized with $Var(\mathbf{w}_l) = \frac{1.2}{m_{l+1}}$, and the full-connected layer is initialized according to the forward initialization formula $Var(\mathbf{w}_l) = \frac{0.25}{m_l}$. The convolution kernel of WideResNet28-10 network is initialized with $Var(\mathbf{w}_l) = \frac{2.5}{m_{l+1}}$, and the full-connected layer is initialized according to the forward initialization formula $Var(\mathbf{w}_l) = \frac{1.0}{m_l}$.

On the CIFAR-100 dataset, we have re-implemented the Kaiming initialization method and compared the test results with our initialization method in Table 3. All experiments were trained for 200 epochs.

Table 3
Results on CIFAR-100.

Model	Params	Initialization method	Data augmentation	Acc (best) %
VGGNet-19	20.04M	Kaiming ours	Cutout	75.01 **75.29**
ResNet-110	1.73M	Kaiming ours	Cutout	74.44 **75.22**
DenseNet-100	0.77M	Kaiming ours	Cutout	78.82 **79.35**
WideResNet28-10	36.49M	Kaiming ours	AutoAgument + Cutout	82.84 **83.17**

5.5. Discussion

For image classification, the fully connected linear classification layer which is the last layer has no activation function and BN, and its initialization method is mainly using the Xavier method [20] or initialized with constant values. When the initial learning rate is bigger, we let the initial variance of the fully connected linear layer be smaller, and the corresponding back propagation error vector norm $\|\delta\|$ of the layer becomes smaller, as shown in Eqn. (42), which can avoid network training oscillation to some extent; when the initial learning rate is small, we let the initial variance of the fully connected linear layer be larger, and the corresponding back propagation error vector norm $\|\delta\|$ of the layer becomes bigger, as shown in Eqn. (42), which can speed up the network convergence speed. In summary, the fully connected linear layer for image classification is more appropriately initialized with Eqn. (41), here, m is the number of input channels.

6. Conclusion

In this paper, we derive the necessary conditions for the convergence of CNNs. Based on this necessary condition we derive the initialization equation for plain network, and further derive the initialization method of plain network whose activation functions are ReLU and Prelu. At the same time, the necessary conditions for the convergence of neural networks proposed in this article can explain part of the mathematical reasons for the benefits brought by the BN method to neural networks. The initial learning rate has a significant impact on the convergence and convergence speed of CNNs, and we deduce the initialization method of CNN with BN method which is related to the initial learning rate. Experiments were conducted on VGGNet-19 and ResNet-34 networks, and the results verified our inference that different initial learning rates correspond to different initialization methods. Finally our initialization method was tested on four network models with the CIFAR-10 and CIFAR-10 dataset, it achieved significant improvement.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] Y. LeCun, B. Boser, J.S. Denker, D. Henderson, R.E. Howard, W. Hubbard, L.D. Jackel, Backpropagation applied to hand written zip code recognition, Neural Comput. 1 (4) (Dec 1989) 541–551.
[2] A. Krizhevsky, I. Sutskever, G.E. Hinton, Imagenet classification with deep convolutional neural networks, in: Proceedings of the 25th International Conference on Neural Information Processing Systems, vol. 1, Dec 2012, pp. 1097–1105.
[3] Karen Simonyan, Andrew Zisserman, Very deep convolutional networks for large-scale image recognition, in: International Conference on Learning Representations (ICLR), May 2015, pp. 1–14.
[4] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, Ruslan Salakhutdinov Dropout, A simple way to prevent neural networks from overfitting, J. Mach. Learn. Res. 15 (56) (Jun 2014) 1929–1958.
[5] Diederik Kingma, Jimmy Ba Adam, A method for stochastic optimization, in: 2015 International Conference on Learning Representations (ICLR), 2015.
[6] Ilya Sutskever, James Martens, George Dahl, Geoffrey Hinton, On the importance of initialization and momentum in deep learning, in: International Conference on Acoustics, Speech and Signal Processing (IEEE-ICASSP), 2013.
[7] Kevin Scaman, Aladin Virmaux, Lipschitz regularity of deep neural networks: analysis and efficient estimation, in: 32nd Conference on Neural Information Processing Systems (NeurIPS 2018), 2018.
[8] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, Andrew Rabinovich, Going deeper with convolutions, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015.
[9] Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun, Deep residual learning for image recognition, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), IEEE, 2016, pp. 770–778.
[10] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, Kilian Q. Weinberger, Densely connected convolutional networks, in: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), IEEE, Nov 2017, pp. 2261–2269.
[11] Sergey Zagoruyko, Nikos Komodakis, Wide residual networks, in: Proceedings of the British Machine Vision Conference (BMVC), Sep 2016, pp. 87.1–87.12.
[12] Sergey Ioffe, Christian Szegedy, Batch normalization: accelerating deep network training by reducing internal covariate shift, in: 2015 International Conference on Machine Learning (ICML), 2015, pp. 448–456.
[13] Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun, Delving deep into rectifiers: surpassing human-level performance on imagenet classification, in: Proceedings of the IEEE International Conference on Computer Vision, 2015, pp. 1026–1034.
[14] Hongyi Zhang, Yann N. Dauphin, Tengyu Ma, Fixup initialization: residual learning without normalization, in: International Conference on Learning Representations (ICLR), 2019.
[15] V. Nair, G.E. Hinton, Rectified linear units improve restricted Boltzmann machines, in: International Conference on Machine Learning (ICML), 2010, pp. 807–814.
[16] Shibani Santurkar, D. Tsipras, Andrew Ilyas, A. Madry, How does batch normalization help optimization? (No, it is not about internal covariate shift), in: NIPS'18: Proceedings of the 32nd International Conference on Neural Information Processing Systems, Dec 2018, pp. 2488–2498.
[17] A. Krizhevsky, Learning multiple layers of features from tiny images, Technical report, 2009.
[18] Martin Arjovsky, Soumith Chintala, Léon Bottou, Wasserstein generative adversarial networks, in: ICML'17: Proceedings of the 34th International Conference on Machine Learning (ICML), vol. 70, 2017, pp. 214–223.
[19] Ian J. Goodfellow, Jean Pouget-Abadie Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, Yoshua Bengio, Generative adversarial nets, in: NIPS'14: Proceedings of the 27th International Conference on Neural Information Processing Systems, vol. 2, 2014, pp. 2672–2680.
[20] X. Glorot, Y. Bengio, Understanding the difficulty of training deep feedforward neural networks, in: International Conference on Artificial Intelligence and Statistics, 2010, pp. 249–256.
[21] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, T. Darrell Caffe, Convolutional architecture for fast feature embedding, in: Proceedings of the 22nd ACM International Conference on Multimedia, 2014, pp. 675–678.
[22] Soham De, Samuel L. Smith, Batch normalization biases residual blocks towards the identity function in deep networks, in: 34th Conference on Neural Information Processing Systems (NeurIPS 2020), 2020.

[23] Andrew Brock, Soham De, Samuel L. Smith, Characterizing signal propagation to close the performance gap in unnormalized resnets, in: International Conference on Learning Representations (ICLR), 2021.
[24] Andrew Brock, Soham De, Samuel L. Smith, Karen Simonyan, Highperformance large-scale image recognition without normalization, in: International Conference on Learning Representations (ICLR), 2021.
[25] A.M. Saxe, James L. McClelland, Surya Ganguli, Exact solutions to the nonlinear dynamics of learning in deep linear neural networks, in: International Conference on Learning Representations (ICLR), 2014.
[26] Jeffrey Pennington, Samuel Schoenholz, Surya Ganguli, Resurrecting the sigmoid in deep learning through dynamical isometry: theory and practice, in: Advances in Neural Information Processing Systems (NeurIPS 2017), 2017, pp. 4785–4795.
[27] Boris Hanin, Which neural net architectures give rise to exploding and vanishing gradients?, in: 32nd Conference on Neural Information Processing Systems (NeurIPS 2018), 2018.
[28] Di Xie, Jiang Xiong, Shiliang Pu, All you need is a good init, in: 2016 International Conference on Learning Representations (ICLR), 2016, pp. 1–13.
[29] Lechao Xiao, Yasaman Bahri, Jascha Sohl-Dickstein, Samuel S. Schoenholz, Jeffrey Pennington, Dynamical isometry and a mean field theory of cnns: how to train 10,000-layer vanilla convolutional neural networks, in: Proceedings of the 35th International Conference on Machine Learning (PMLR), vol. 80, 2018, pp. 5393–5402.
[30] Yann N. Dauphin, Samuel S. Schoenholz Metainit, Initializing learning by learning to initialize, in: 33rd Conference on Neural Information Processing Systems (NeurIPS 2019), 2019, pp. 12645–12657.
[31] Chen Zhu, Renkun Ni, Zheng Xu, Kezhi Kong, W.R. Huang, T. Goldstein. Gradinit, Learning to initialize neural networks for stable and efficient training, arXiv: 2102.08098 [abs].
[32] Priya Goyal, Piotr Dollár, Ross B. Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, Kaiming He, Accurate, large minibatch sgd: training imagenet in 1 hour, Technical report, arXiv, June 2017.
[33] Tong He, Zhi Zhang, Hang Zhang, Zhongyue Zhang, Junyuan Xie, Mu Li, Bag of tricks for image classification with convolutional neural networks, in: 2019 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2019.
[34] Terrance Devries, Graham W. Taylor, Improved regularization of convolutional neural networks with cutout, arXiv:1708.04552 [abs], 2017.
[35] Zhun Zhong, Liang Zheng, Guoliang Kang, Shaozi Li, Yi Yang, Random erasing data augmentation, in: 34th Association for the Advancement of Artificial Intelligence (AAAI 2020), 2020.
[36] Ekin D. Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, Quoc V. Le Autoaugment, Learning augmentation strategies from data, in: 2019 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), IEEE, 2019.
[37] Hongyi Zhang, Moustapha Cissé, Yann N. Dauphin, David Lopez-Paz, Mixup: beyond empirical risk minimization, in: International Conference on Learning Representations (ICLR), 2018.
[38] Yurii Nesterov, A method for unconstrained convex minimization problem with the rate of convergence $O(1/k^2)$, Dokl. Akad. Nauk SSSR (Sov. Math. Dokl.) 269 (1983) 543–547.
[39] Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, Soumith Chintala, Adam Paszke, Sam Gross, Pytorch: an imperative style, high-performance deep learning library, in: 33rd Conference on Neural Information Processing Systems (NeurIPS 2019), 2019, pp. 8026–8037.
[40] Shuai Zhao, Liguang Zhou, Wenxiao Wang Deng Cai, Tin Lun Lam, Yangsheng Xu, Towards better accuracy-efficiency trade-offs: divide and co-training, arXiv: 2011.14660 [abs], 2021.
[41] Vikas Verma, Alex Lamb, Christopher Beckham, Amir Najafi, Ioannis Mitliagkas, David Lopez-Paz, Yoshua Bengio, Manifold mixup: better representations by interpolating hidden states, in: International Conference on Machine Learning (ICML), 2019.
[42] Jie Qin, Jiemin Fang, Qian Zhang, Wenyu Liu, Xingang Wang, Xinggang Wang Resizemix, Mixing data with preserved object information and true labels, arXiv:2012.11101 [abs], 2020.
[43] Alexandre Ramé, Rémy Sun, Matthieu Cord, Mixmo: mixing multiple inputs for multiple outputs via deep subnetworks, arXiv:2103.06132 [abs], 2021.
[44] Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, Youngjoon Yoo Cutmix, Regularization strategy to train strong classifiers with localizable features, in: IEEE International Conference on Computer Vision (ICCV), IEEE, 2019.

Huibin Zhang is a Ph.D. candidate in Yanshan University. He is currently a associate professor at Computer Department of Xinzhou Teachers University, Xinzhou, PR China. His current research interests include applied mathematics, deep learning and big data.

Liping Feng received the Ph.D. degree in Chongqing University, in 2013. She is currently a professor in Computer Department of Xinzhou Teachers University, Xinzhou, PR China. Her current research interests include distributed optimization, network security, and dynamical modeling.

Xiaohua Zhang is a Ph.D. candidate in Yanshan University. She is currently a lecturer at the School of Mathematics and Information Science and Technology, Hebei Normal University of Science and Technology, PR China. Her current research interests include compressed sensing, deep learning and magnetic resonance imaging.

Yuchi Yang is a Ph.D. candidate at Yanshan University. She joined as a lecturer at Hebei University of Economics and Business. Her current research interests include signal and image processing, holographic superresolution, and compressed sensing.

Jing Li She is currently a lecturer at Computer Department of Xinzhou Teachers University, Xinzhou, PR China. Her current research interests include applied mathematics, deep learning and big data.