

# Understanding the Disharmony between Weight Normalization Family and Weight Decay: $\epsilon$ -shifted $L_2$ Regularizer

Xiang Li, Shuo Chen, Yan Xia and Jian Yang\*

**Abstract**—The merits of fast convergence and potentially better performance of the weight normalization family have drawn increasing attention in recent years. These methods use standardization or normalization that changes the weight  $\mathbf{W}$  to  $\mathbf{W}'$ , which makes  $\mathbf{W}'$  independent to the magnitude of  $\mathbf{W}$ . Surprisingly,  $\mathbf{W}$  must be decayed during gradient descent, otherwise we will observe a severe under-fitting problem, which is very counter-intuitive since weight decay is widely known to prevent deep networks from over-fitting. Moreover, if we substitute (e.g., weight normalization)  $\mathbf{W}' = \frac{\mathbf{W}}{\|\mathbf{W}\|}$  in the original loss function  $\sum_i^N L(f(\mathbf{x}_i; \mathbf{W}'), y_i) + \frac{1}{2}\lambda\|\mathbf{W}'\|^2$ , it is observed that the regularization term  $\frac{1}{2}\lambda\|\mathbf{W}'\|^2$  will be canceled as a constant  $\frac{1}{2}\lambda$  in the optimization objective. Therefore, to decay  $\mathbf{W}$ , we need to explicitly append this term:  $\frac{1}{2}\lambda\|\mathbf{W}\|^2$ . In this paper, we *theoretically* prove that  $\frac{1}{2}\lambda\|\mathbf{W}\|^2$  merely modulates the effective learning rate for improving objective optimization, and has no influence on generalization when the weight normalization family is compositely employed. Furthermore, we also expose several critical problems when introducing weight decay term to weight normalization family, including the missing of global minimum and training instability. To address these problems, we propose an  $\epsilon$ -shifted  $L_2$  regularizer, which shifts the  $L_2$  objective by a positive constant  $\epsilon$ . Such a simple operation can theoretically guarantee the existence of global minimum, while preventing the network weights from being too small and thus avoiding gradient float overflow. It significantly improves the training stability and can achieve slightly better performance in our practice. The effectiveness of  $\epsilon$ -shifted  $L_2$  regularizer is comprehensively validated on the ImageNet, CIFAR-100, and COCO datasets. Our codes and pretrained models will be released in <https://github.com/implus/PytorchInsight>.

**Index Terms**—Weight normalization, weight standardization, weight decay, deep neural networks, disharmony, gradient float overflow.



## 1 INTRODUCTION

THE normalization methodologies on *features* have made great progress in recent years, with the introduction of BN [20], IN [44], LN [1], GN [47] and SN [32]. These methods mainly focus on a zero mean and unit variance normalization operation on a specific dimension (or multiple dimensions) of *features*, which makes deep neural architectures [12], [13], [18], [46] much easier to optimize, leading to robust solutions with favorable generalization performance.

Beyond *feature* normalization, there is an increasing interest on the normalization of network *weights*. Weight Normalization (WN) [39] first separates the learning of the length and direction of weights, and it performs satisfactorily on several relatively small datasets. In some contexts of generative adversarial networks (GAN) [8], Weight Normalization with Translated ReLU [48] is shown to achieve superior results. Later, Centered Weight Normalization (CWN) [19] further powers WN by additionally centering their input weights, ulteriorly improving the conditioning and convergence speed. Recently, very similar to CWN, Weight Standardization (WS) [36] aims to standardize the weights

with zero mean and unit variance. On the large-scale tasks (ImageNet [5] classification/COCO [27] detection), WS further enhances optimization convergence and generalization performance, under the cooperation of feature normalizations such as GN and BN.

In terms of weight normalization family, despite its appealing success, there is still one confusing mystery – the disharmony between weight normalization family and weight decay [24]. Note that weight decay is widely interpreted as a form of  $L_2$  regularization [30] because it can be derived from the gradient of the  $L_2$  norm of the weights [31]. Specifically, we consider training a single-layer single-output neural network  $f(\mathbf{x}; \mathbf{W}')$ , where  $\mathbf{x}, \mathbf{W}' \in \mathcal{R}^n$  with the following loss function to be minimized:

$$\hat{\mathcal{L}}(\mathbf{W}') = \sum_i^N L(f(\mathbf{x}_i; \mathbf{W}'), y_i) + \frac{1}{2}\lambda\|\mathbf{W}'\|^2. \quad (1)$$

In Eq. (1),  $N$  denotes the number of training samples,  $\hat{\mathcal{L}}$  consists of task-related loss  $L(\cdot, \cdot)$  w.r.t. the input/label pair  $(\mathbf{x}_i, y_i)$ , and the regularization term  $\frac{1}{2}\lambda\|\mathbf{W}'\|^2$  with a constant  $\lambda$  to balance against  $L(\cdot, \cdot)$ . For simplicity, we use weight normalization to re-parameterize  $\mathbf{W}'$ , regardless the learning of its length. By substituting  $\mathbf{W}' = \frac{\mathbf{W}}{\|\mathbf{W}\|}$  in Eq. (1), we can get  $\sum_i^N L(f(\mathbf{x}_i; \frac{\mathbf{W}}{\|\mathbf{W}\|}), y_i) + \frac{1}{2}\lambda$ , which is equivalent to minimize the following function

$$\sum_i^N L(f(\mathbf{x}_i; \frac{\mathbf{W}}{\|\mathbf{W}\|}), y_i). \quad (2)$$

- Xiang Li, Shuo Chen, and Jian Yang are with the School of Computer Science and Engineering, Nanjing University of Science and Technology, Nanjing 210094, China (E-mail: (xiang.li.implus, shuochen, csjyang)@njust.edu.cn). Xiang Li is also a visiting scholar at Momenta. Yan Xia (E-mail: xiayan@momenta.ai) is the research and development director of Momenta. (Corresponding author: Jian Yang)

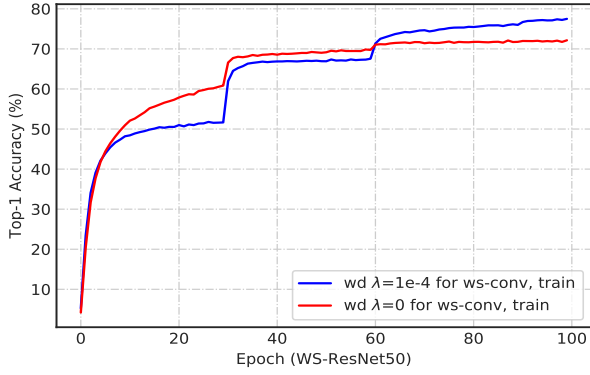


Fig. 1. Counter-intuitive performance degradation (under-fitting) by setting weight decay parameter  $\lambda$  to 0 for WS-equipped convolutions. Top-1 Accuracy via single  $224 \times$  crop on the ImageNet training set is plotted.

Interestingly, the weight decay term has indeed *disappeared*. In the case of WS, we can get similar conclusions by replacing  $\mathbf{W}' = \frac{\mathbf{W} - \bar{\mathbf{W}}}{\sqrt{\frac{\|\mathbf{W} - \bar{\mathbf{W}}\|^2}{n}}}$ :

$$\sum_i^N L(f(x_i; \frac{\mathbf{W} - \bar{\mathbf{W}}}{\sqrt{\frac{\|\mathbf{W} - \bar{\mathbf{W}}\|^2}{n}}}, y_i), \quad (3)$$

where  $\bar{\mathbf{W}} = \frac{\sum_i^n \mathbf{W}_i}{n}$ . It probably makes sense since weight decay will not take effect on a fixed distribution of normalized weights. However, when we apply Eq. (3) to WS-equipped ResNet-50 (WS-ResNet-50) on ImageNet dataset, i.e., setting weight decay ratio  $\lambda$  to 0 for all WS-equipped convolutions, we observe a severe degradation with significant performance drop in training set (Fig. 1). It is incredibly *strange* that weight decay is known to prevent the training from over-fitting the data, but it appears that, removing the weight decay instead puts the network into a serious under-fitting, which is very counter-intuitive.

To answer above questions, in this paper, we first prove that in Eq. (2) (and Eq. (3)), the addition of weight decay term of  $\mathbf{W}$  does not change the optimization goal. Therefore, weight decay loses its original role that finds a better generalized solution by traditionally introducing a different loss part against the task-related one. At the same time, basing on the derivation of the gradient formula of  $\mathbf{W}$ , we further prove that weight decay only takes effect in modulating the effective learning rate to help the gradient descent process when the weight normalization family is employed simultaneously, and empirically demonstrate how it adjusts the effective learning rate. Interestingly, we also get an additional empirical discovery: training a network with weight normalization and weight decay implicitly includes an approximate warmup [9] process, which probably explains the slightly improved performance on the original baselines.

The current common and default operation [36], [39] to optimize networks with weight normalization family is to continue to preserve the traditional decay term of  $\mathbf{W}$  for better convergence that comes from ensuring the stable effective learning rate, i.e., to explicitly add  $\frac{1}{2}\lambda\|\mathbf{W}\|^2$  on Eq. (2):

$$\sum_i^N L(f(x_i; \frac{\mathbf{W}}{\|\mathbf{W}\|}), y_i) + \frac{1}{2}\lambda\|\mathbf{W}\|^2. \quad (4)$$

However, there are many potential problems in taking the final optimization objective as Eq. (4). First, we prove that Eq. (4) has no global minimum theoretically. In addition, the improper selection of  $\lambda$  will push the magnitude ( $\|\mathbf{W}\|$ ) of weights to 0 and easily lead to training failures due to gradient float overflow (as the corresponding gradient is proportional to  $\frac{1}{\|\mathbf{W}\|}$ ), especially for certain adaptive gradient methods (e.g., Adam [22]) which accumulates the square of gradients.

To address these problems, we propose a very simple yet effective  $\epsilon$ -shifted  $L_2$  regularizer, which shifts the  $L_2$  objective by a positive constant  $\epsilon$ . The shifted  $\epsilon$  prevents network weights from being too small, thus it will directly avoid gradient float overflow risks. Such a simple operation can theoretically guarantee the existence of global minimum, whilst greatly improving the training stability. Beyond the training stability, it further brings gains on performance over a wide range of architectures, probably due to its dynamic decay mechanism, which we will discuss later. The effectiveness of our method is comprehensively demonstrated by experiments on the ImageNet [5], CIFAR-100 [23] and COCO [27] datasets.

To summarize our contributions:

- We thoroughly analyze the disharmony between weight normalization family and weight decay, and expose the critical problems including lack of global minimum and training instability, which are caused by the optimization of weight decay term in the final loss objective when weight normalization is simultaneously applied.
- We theoretically prove that weight decay loses the ability to enhance generalization in the weight normalization family, and only plays a role in regulating effective learning rate to help training. We demonstrate that when optimizing with SGD, the weight decay term can be cancelled by simply scaling the learning rate with a constant at each gradient descent step, where the constant is only determined by the hyper-parameters and irrelevant to the training process.
- We propose a simple yet effective  $\epsilon$ -shifted  $L_2$  regularizer to overcome the problems via introducing weight decay into weight normalization family, which significantly improves the training stability whilst achieving better performance over a large range of network architectures on both classification and detection tasks.

## 2 RELATED WORKS

**Weight Normalization Family:** Weight Normalization (WN) [39] takes the first attempt to reparameterize weights by the separation of direction  $\frac{\mathbf{W}}{\|\mathbf{W}\|}$  and length  $g$ :

$$\mathbf{W}' = g \frac{\mathbf{W}}{\|\mathbf{W}\|}. \quad (5)$$

The normalization operation participates in the gradient flow, resulting in accelerated convergence of stochastic gradient descent optimization. WN shows certain advantages in some tasks of supervised image recognition, generative modelling, and deep reinforcement learning. However, [7] points out that in the large-scale ImageNet dataset, the final test accuracy of WN is significantly lower ( $\sim 6\%$ ) than that

of BN [20]. Later, Centered Weight Normalization (CWN) is proposed to further improve the conditioning and accelerate the convergence of training deep networks. The central idea of CWN is an additional centering operation based on WN:

$$\mathbf{W}' = g \frac{\mathbf{W} - \bar{\mathbf{W}}}{\|\mathbf{W} - \bar{\mathbf{W}}\|}. \quad (6)$$

Recently, in order to alleviate the problem of degraded performance of GN [47], Weight Standardization (WS) [36] is proposed, which is very close to CWN but with the learning length  $g$  removed:

$$\mathbf{W}' = \frac{\mathbf{W} - \bar{\mathbf{W}}}{\sqrt{\frac{\|\mathbf{W} - \bar{\mathbf{W}}\|^2}{n}}}, \quad (7)$$

WS is recommended to cooperate with feature normalization methods (such as GN and BN), which leads to further enhanced performance in large-scale tasks and can significantly accelerate the convergence. Introducing WS on the basis of GN or BN can consistently bring gains to multiple downstream visual tasks, including image classification, object detection, instance segmentation, video recognition, semantic segmentation, and point cloud recognition. In this paper, we mainly focus on the weight normalization family and conduct a series of analyses on their properties, especially on their relations to weight decay.

**Weight Decay:** weight decay can be traced back to [24], which is defined as multiplying each weight in the gradient descent at each epoch by a factor  $\lambda$  ( $0 < \lambda < 1$ ). In the Stochastic Gradient Descent (SGD) setting, weight decay is widely interpreted as a form of  $L_2$  regularization [34] because it can be derived from the gradient of the  $L_2$  norm of the weights [31]. It is known to be beneficial for the generalization of neural networks. Recently, [51] identify three distinct mechanisms by which weight decay improves generalization: increasing the effective learning rate for BN, reducing the Jacobian norm, and reducing the effective damping parameter. Similarly, a series of recent work [15], [45] also demonstrates that when using BN, weight decay improves optimization only by fixing the norm to a small range of values, leading to a more stable step size for the weight direction. Although related, these works differ from our work in at least four aspects: 1) they mainly focus on the discussion between the feature normalization (especially BN) and weight decay, whilst we are the first to give a thorough analysis on the disharmony between weight normalization family and weight decay; 2) they solely demonstrate *empirical* results that the accuracy gained by using weight decay can be achieved without it, but only by adjusting the learning rate. However, we give *theoretical* proof and derive how to linearly scale the learning rate at each step, which is also purely determined by the training hyper-parameters; 3) they fail to discover the problems by introducing weight decay into the loss objective with normalized weights, which is heavily revealed and discussed in this article; 4) although weight decay has several potential problems with normalization methods, they have not proposed a solution to replace weight decay. In contrast, our proposed  $\epsilon$ -shifted  $L_2$  regularizer can successfully guarantee the global minimum and training stability to overcome the existing drawbacks, whilst achieving superior performance over a range of tasks.

### 3 ROLES OF WEIGHT DECAY IN WEIGHT NORMALIZATION FAMILY

In this section, we explain the *roles* of weight decay in weight normalization family in details. The theoretical analyses on the roles of weight decay help to understand why weight decay loses the ability to enhance the generalization, but controls the effective learning rate to help the training of deep networks.

#### 3.1 Weight Decay Doesnot Change Optimization Goal

We first prove that in the networks equipped with weight normalization family, the introduction of weight decay does not change the goal of optimization, indicating that weight decay faithfully brings no additional generalization benefits. For analyses, we simply use the concepts of variable decomposition. Specifically, we choose two representative methods from weight normalization family, namely WN and WS, and discuss each in turn. Note that for simplicity, we ignore the learning of the length  $g$  in WN in the following analyses and experiments.

In WN, we aim to prove that minimizing

$$\hat{\mathcal{L}}(\mathbf{W}) = \sum_i^N L(f(\mathbf{x}_i; \frac{\mathbf{W}}{\|\mathbf{W}\|}), y_i) + \frac{1}{2} \lambda \|\mathbf{W}\|^2, \quad (8)$$

is equal to minimizing

$$\mathcal{L}(\mathbf{W}) = \sum_i^N L(f(\mathbf{x}_i; \frac{\mathbf{W}}{\|\mathbf{W}\|}), y_i). \quad (9)$$

Specifically, we let  $\mathbf{A} = \frac{\mathbf{W}}{\|\mathbf{W}\|}$  and  $k = \|\mathbf{W}\|$  ( $k > 0$ ), and decompose the direction and length of  $\mathbf{W}$  as two independent variables. Then the objectives of Eq. (8) and (9) can be rewritten as

$$\begin{aligned} \min_{\mathbf{A}, k} \hat{\mathcal{L}}(\mathbf{A}, k) &= \sum_i^N L(f(\mathbf{x}_i; \mathbf{A}), y_i) + \frac{1}{2} \lambda k^2, \\ \text{s.t. } \|\mathbf{A}\| &= 1, k > 0, \end{aligned} \quad (10)$$

and

$$\begin{aligned} \min_{\mathbf{A}} \mathcal{L}(\mathbf{A}) &= \sum_i^N L(f(\mathbf{x}_i; \mathbf{A}), y_i), \\ \text{s.t. } \|\mathbf{A}\| &= 1, \end{aligned} \quad (11)$$

respectively. Since  $\mathbf{A}$  and  $k$  are two independent variables, we have

$$\begin{aligned} \min_{\mathbf{A}, k} \hat{\mathcal{L}}(\mathbf{A}, k) &= \min_{\mathbf{A}} \sum_i^N L(f(\mathbf{x}_i; \mathbf{A}), y_i) + \min_k \frac{1}{2} \lambda k^2 \\ &= \min_{\mathbf{A}} \mathcal{L}(\mathbf{A}) + \min_k \frac{1}{2} \lambda k^2, \end{aligned} \quad (12)$$

which shows that minimizing  $\hat{\mathcal{L}}$  actually contains the task of minimizing  $\mathcal{L}$ , and it completes the proof. Similarly, in WS we can further decompose the mean and variance of  $\mathbf{W}$  by letting  $\mathbf{B} = \frac{\mathbf{W} - \bar{\mathbf{W}}}{\sqrt{\frac{\|\mathbf{W} - \bar{\mathbf{W}}\|^2}{n}}}$ ,  $m = \bar{\mathbf{W}}$  and  $v = \sqrt{\frac{\|\mathbf{W} - \bar{\mathbf{W}}\|^2}{n}}$  ( $v > 0$ ), where  $\mathbf{B}$ ,  $m$ ,  $v$  are also mutually independent. Again we can have

$$\min_{\mathbf{B}, m, v} \hat{\mathcal{L}}(\mathbf{B}, m, v) = \min_{\mathbf{B}} \mathcal{L}(\mathbf{B}) + \min_{m, v} \frac{1}{2} \lambda n (m^2 + v^2). \quad (13)$$

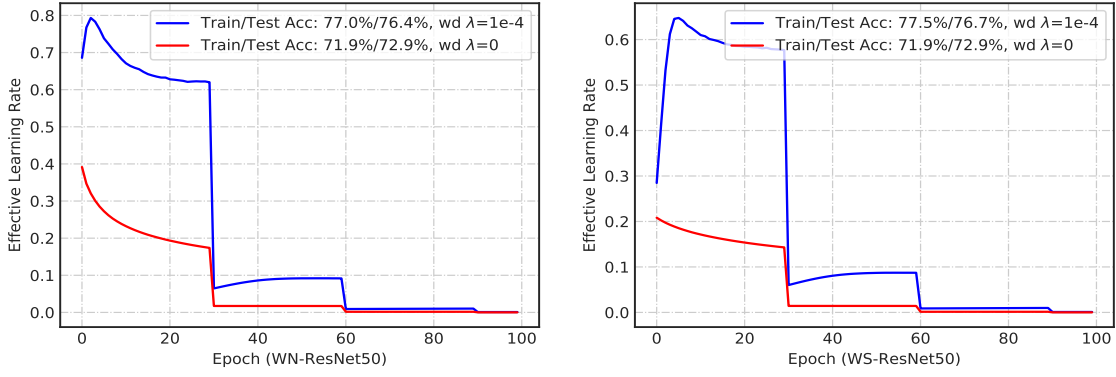


Fig. 2. The comparisons of effective learning rate for different weight decay  $\lambda$  at each epoch during the optimization of WN-ResNet (left) and WS-ResNet (right) on the ImageNet dataset, where the effective learning rates of their first convolutional layer are depicted. The blue curve ensures a larger effective learning rate which helps the networks converge.

Therefore, according to the above analyses, for the networks with weight normalization family employed, the introduction of weight decay does not essentially change the learning objective, which implies that it takes no effect on the network generalization capability.

### 3.2 Weight Decay Ensures Effective Learning Rate

Since weight decay does not bring a regularization effect to a network with weight normalization family, why is it indispensable in the training process? The central reason is that weight decay helps to control the effective learning rate in a stable and reasonable range. Taking WN as an example, we can derive the gradient of  $\mathbf{W}$  as (the deviate to Eq. (9)):

$$\frac{\partial \mathcal{L}}{\partial \mathbf{W}} = \frac{\partial \mathcal{L}}{\partial \mathbf{A}} * \left( \frac{1}{\|\mathbf{W}\|} - \frac{\mathbf{W} * \mathbf{W}}{\|\mathbf{W}\|^3} \right), \quad (14)$$

where  $*$  denotes the element-wise product. If we consider one gradient descent update at  $t$  step for an element in  $\mathbf{W}$ , i.e.,  $W_i$ , with the use of learning rate  $\eta$ , we have:

$$W_i^{t+1} = W_i^t - \frac{\eta}{\|\mathbf{W}^t\|} \left( 1 - \frac{(W_i^t)^2}{\sum_j^n (W_j^t)^2} \right) \frac{\partial \mathcal{L}}{\partial A_i}. \quad (15)$$

Eq. (15) demonstrates that even if  $\mathbf{A} = \frac{\mathbf{W}}{\|\mathbf{W}\|}$  is fixed, the gradient update in terms of  $\mathbf{W}$  can vary according to its magnitude  $\|\mathbf{W}\|$ . The reason is that fixed  $\mathbf{A}$  can only leads to fixed  $\frac{\partial \mathcal{L}}{\partial A_i}$  and  $(1 - \frac{(W_i^t)^2}{\sum_j^n (W_j^t)^2})$  term. Consequently, the entire update step size can be determined by  $\frac{\eta}{\|\mathbf{W}\|}$ , which exactly controls the effective learning rate similarly defined as in [15], [45]. Here we have two conclusions:

- If we do not limit  $\|\mathbf{W}\|$  during the update process, the weights can grow unbounded  $\|\mathbf{W}\| \rightarrow \infty$ , and the effective learning rate goes to 0 ( $\frac{\eta}{\|\mathbf{W}\|} \rightarrow 0$ ).
- On the contrary, if we decay  $\|\mathbf{W}\|$  too much during the optimization ( $\|\mathbf{W}\| \rightarrow 0$ ), the effective learning rate will grow unbounded ( $\frac{\eta}{\|\mathbf{W}\|} \rightarrow \infty$ ), which leads to gradient float overflow and training failures. This is part of the motivation of our proposed method and we will discuss it in details later.

The similar analysis can be conducted in the case of WS,

where one update step is:

$$W_i^{t+1} = W_i^t - \frac{\eta}{\sqrt{\frac{\|\mathbf{W}^t - \bar{\mathbf{W}}^t\|^2}{n}}} \left( 1 - \frac{1}{n} - \frac{(W_i^t - \bar{W}^t)^2}{\sum_j^n (W_j^t - \bar{W}^t)^2} \right) \frac{\partial \mathcal{L}}{\partial B_i}, \quad (16)$$

which has  $\frac{\eta}{\sqrt{\frac{\|\mathbf{W}^t - \bar{\mathbf{W}}^t\|^2}{n}}}$  term as its effective learning rate.

To prove that weight decay only ensures effective learning rate, we conduct theoretical analyses as follows:

**Theorem 1.** Using SGD, the training trajectory of  $\hat{\mathcal{L}}$  (Eq. (8) with weight decay) can be completely reproduced by simply scaling the learning rate at each step when optimizing  $\mathcal{L}$  (Eq. (9) without weight decay).

*Proof.* Here we focus on the normalized variables to represent the training trajectory, e.g.,  $\mathbf{A}$  (or  $\mathbf{B}$ ), as they are the ultimate weights with which networks use to operate. In the case of WN, we suppose optimizing  $\hat{\mathcal{L}}(\mathbf{A}, k)$  and  $\mathcal{L}(\mathbf{A})$  take  $T$  steps in total, and at each step, we feed the same data batch to both of them. For the ease of reference, the corresponding variables at  $t$  step for optimizing  $\hat{\mathcal{L}}$  are marked with superscript  $\hat{\mathcal{L}}_t$ , e.g.,  $\mathbf{W}^{\hat{\mathcal{L}}_t}$ . Such notations are kept similarly in optimizing  $\mathcal{L}$ , e.g.,  $\mathbf{W}^{\mathcal{L}_t}$ . We further assume that two optimization processes start from the same initial weights  $\mathbf{A}^{\hat{\mathcal{L}}_0} = \mathbf{A}^{\mathcal{L}_0}$ , which also means  $\mathbf{W}^{\hat{\mathcal{L}}_0} = p_0 \mathbf{W}^{\mathcal{L}_0}$ , where  $p_t$  is a scale between  $\mathbf{W}^{\hat{\mathcal{L}}_t}$  and  $\mathbf{W}^{\mathcal{L}_t}$  (if  $p_t$  exists). Specifically, we aim to prove that there exists a sequence of  $\{d_1, \dots, d_T\}$  as multipliers (note that  $d$  must be independent of the training process) for learning rate  $\eta$  during optimizing  $\mathcal{L}$ , and it ensures  $\mathbf{A}^{\hat{\mathcal{L}}_t} = \mathbf{A}^{\mathcal{L}_t}$  for every step  $t$  from  $\{1, \dots, T\}$ . We take the standard SGD [2], [38] for analysis via mathematical induction:

1) As stated in the assumption, we have  $\mathbf{A}^{\hat{\mathcal{L}}_0} = \mathbf{A}^{\mathcal{L}_0}$  hold for  $t = 0$ , indicating  $\mathbf{W}^{\hat{\mathcal{L}}_0} = p_0 \mathbf{W}^{\mathcal{L}_0}$ . Here we do not have  $d_0$  since the gradient descent step does not start in the initialization phase.

2) Suppose  $\mathbf{A}^{\hat{\mathcal{L}}_q} = \mathbf{A}^{\mathcal{L}_q}$  holds for  $t = q$  with  $\mathbf{W}^{\hat{\mathcal{L}}_q} = p_q \mathbf{W}^{\mathcal{L}_q}$ , we needs to prove  $\mathbf{A}^{\hat{\mathcal{L}}_{q+1}} = \mathbf{A}^{\mathcal{L}_{q+1}}$  under certain expression of  $d_q$ . Let us expand  $\mathbf{W}^{\hat{\mathcal{L}}_{q+1}}$  and  $\mathbf{W}^{\mathcal{L}_{q+1}}$  by



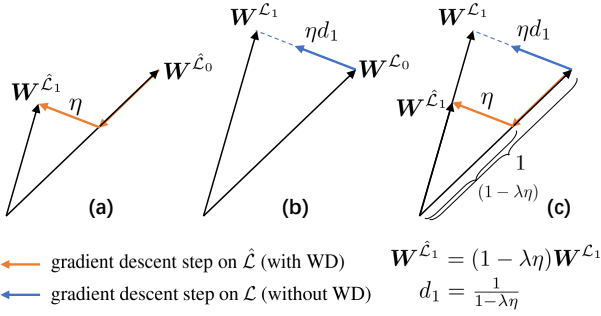


Fig. 3. Illustration of proving that weight decay can be entirely replaced by modulating the learning rate (under the setting of  $p_0 = 1$  in this example). (a) denotes one update with weight decay. (b) denotes one update without weight decay. (c) shows the similar triangle relationship between these two updates, where the equations  $\mathbf{W}^{\hat{\mathcal{L}}_1} = (1 - \lambda\eta)\mathbf{W}^{\mathcal{L}_1}$  and  $d_1 = \frac{1}{1 - \lambda\eta}$  can be easily derived.

performing one gradient descent step:

$$\begin{aligned} \mathbf{W}^{\hat{\mathcal{L}}_{q+1}} &= (1 - \lambda\eta)\mathbf{W}^{\hat{\mathcal{L}}_q} - \frac{\eta}{\|\mathbf{W}^{\hat{\mathcal{L}}_q}\|} \left(1 - \frac{\mathbf{W}^{\hat{\mathcal{L}}_q} * \mathbf{W}^{\hat{\mathcal{L}}_q}}{\|\mathbf{W}^{\hat{\mathcal{L}}_q}\|^2}\right) * \frac{\partial \mathcal{L}}{\partial \mathbf{A}^{\hat{\mathcal{L}}_q}} \\ &= (1 - \lambda\eta)p_q \mathbf{W}^{\mathcal{L}_q} - \frac{\eta}{p_q \|\mathbf{W}^{\mathcal{L}_q}\|} \left(1 - \frac{\mathbf{W}^{\mathcal{L}_q} * \mathbf{W}^{\mathcal{L}_q}}{\|\mathbf{W}^{\mathcal{L}_q}\|^2}\right) * \frac{\partial \mathcal{L}}{\partial \mathbf{A}^{\mathcal{L}_q}}, \end{aligned} \quad (17)$$

and

$$\mathbf{W}^{\mathcal{L}_{q+1}} = \mathbf{W}^{\mathcal{L}_q} - \frac{\eta d_q}{\|\mathbf{W}^{\mathcal{L}_q}\|} \left(1 - \frac{\mathbf{W}^{\mathcal{L}_q} * \mathbf{W}^{\mathcal{L}_q}}{\|\mathbf{W}^{\mathcal{L}_q}\|^2}\right) * \frac{\partial \mathcal{L}}{\partial \mathbf{A}^{\mathcal{L}_q}}. \quad (18)$$

Therefore, it is very obvious to deduce from Eq. (17) and (18) that when  $d_q = \frac{1}{p_q^2(1 - \lambda\eta)}$ , we can have

$$\mathbf{W}^{\hat{\mathcal{L}}_{q+1}} = (1 - \lambda\eta)p_q \mathbf{W}^{\mathcal{L}_{q+1}}, \quad (19)$$

which consequently leads to  $\mathbf{A}^{\hat{\mathcal{L}}_{q+1}} = \mathbf{A}^{\mathcal{L}_{q+1}}$  and thereby it completes the proof. At the same time, we can also derive the recursive formula for  $p$ :

$$p_{q+1} = (1 - \lambda\eta)p_q. \quad (20)$$

Given the sequence of  $p$  generated from Eq. (20), the resulted sequence  $\{d_1, \dots, d_T\}$  of  $d$  then becomes  $\{\frac{1}{p_1^2(1 - \lambda\eta)}, \dots, \frac{1}{p_T^2(1 - \lambda\eta)}\}$  finally. The similar deductions can be carried out for the case of WS. To give a better illustration of the proof, we let  $p_0 = 1$  and demonstrate the first gradient descent update of  $\hat{\mathcal{L}}$  and  $\mathcal{L}$  in Fig. 3. It is easy to see that  $\mathbf{W}^{\hat{\mathcal{L}}_1}$  and  $\mathbf{W}^{\mathcal{L}_1}$  form a similar triangle relationship and their scale factor is only determined by the hyper-parameter  $\lambda$  and  $\eta$ .  $\square$

According to the above analyses, we conclude that for networks with weight normalization family, weight decay only takes effect in modulating effective learning rate, and theoretically, we can replace it simply by adjusting the learning rate in each iteration with a calculated ratio which is only related to the hyper-parameter  $\lambda$  and  $\eta$ .

To show how weight decay regulates the effective learning rate, we plot the mean effective learning rate of all the filters from the first layer of WN-ResNet-50 and WS-ResNet-

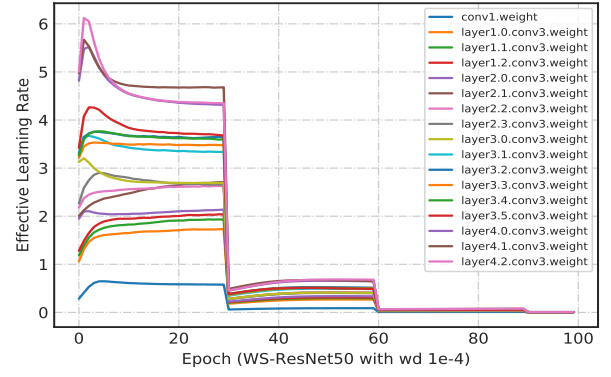


Fig. 4. The curves of mean effective learning rate of multiple WS-equipped convolutional layers. We observe an interesting warmup phenomenon in the initial training stage. In the form of “layer $a.b$ .conv $c$ ”, “ $a$ ” denotes the stage number, “ $b$ ” denotes the bottleneck number, “ $c$ ” represents the order of convolutional layer inside this bottleneck. For reference, the first “conv1” means the first convolutional layer, which is exactly the same with the blue curve in WS-ResNet-50 of Fig. 2.

50 throughout the training process in Fig. 2, where 0 and  $1e-4$  weight decay ratios are applied to the corresponding convolutional layers respectively. It is observed that the effective learning rate is appropriately controlled in a relatively large range with weight decay applied, which leads to a better optimized solution.

One more interesting empirical observation is that the control of effective learning rate by weight decay in the early stage is quite similar to a warmup process, and the effectiveness of warmup has been generally confirmed in [9], [14], [28], [50]. As shown in the Fig. 4, we sample and investigate a set of convolutional layers, and observe that almost all of the layers show an increase in the effective learning rate of several epochs at the beginning of training. The effect of implicit warmup may explain why networks equipped with WS can have slightly improved performance [36]. To investigate this deeper, we make additional experiments by explicitly adding warmup process (i.e., linearly increasing the learning rate from 0 to 0.1 during the first 5 epochs) into the training of ResNet-50, and thus partially confirm this conclusion in Table 1. Further, the effective learning rate of the first convolutional layer for training WS-ResNet-50 and ResNet-50 with warmup are depicted in Fig. 5. Note that the definition of effective learning rate for ResNet-50 follows [15], in order to keep them in a similar magnitude. We suprisingly find that the two curves are very closely matched, which implies that training networks with weight normalization family and weight decay can implicitly have certain benefits of warmup technique.

#### 4 PROBLEMS VIA INTRODUCING WEIGHT DECAY IN WEIGHT NORMALIZATION FAMILY

Despite the certain practical success and benefits of applying traditional weight decay to control the effective learning rate, there are still several serious problems in essence, which are rarely revealed or noticed before our work. In this section, we discuss about these *problems* of introducing weight decay term in the loss objective for weight normalization family in details.

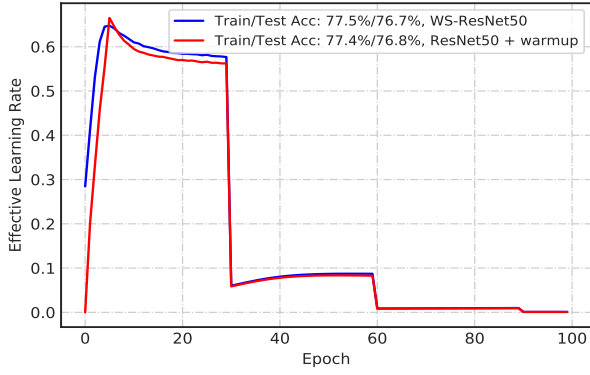


Fig. 5. The comparisons of effective learning rate of the first convolutional layer for training WS-ResNet-50 and ResNet-50 with warmup at each epoch during the optimization. The training of WS-ResNet-50 can implicitly simulate the process of warmup to some extent.

#### 4.1 No Guarantee of Global Minimum

We first consider WN. If we introduce weight decay term of  $\mathbf{W}$  to the final loss objective, i.e., Eq. (4), we can prove that for  $\mathbf{W}$ , the entire loss function does not theoretically guarantee a global minimum. Here we use the proof by contradiction:

If there exists a global minimum  $\mathbf{W}^*$  such that the objective (Eq. (4)) is minimized, then we have the smallest loss  $\hat{\mathcal{L}}(\mathbf{W}^*)$  as

$$\hat{\mathcal{L}}(\mathbf{W}^*) = \sum_i^N L(f(\mathbf{x}_i; \frac{\mathbf{W}^*}{\|\mathbf{W}^*\|}, y_i) + \frac{1}{2}\lambda\|\mathbf{W}^*\|^2. \quad (21)$$

Let's take a real number  $\alpha (0 < \alpha < 1)$  and form a new solution  $\mathbf{W}^\# = \alpha\mathbf{W}^*$ . Then we have:

$$\begin{aligned} \hat{\mathcal{L}}(\mathbf{W}^\#) &= \sum_i^N L(f(\mathbf{x}_i; \frac{\mathbf{W}^\#}{\|\mathbf{W}^\#\|}, y_i) + \frac{1}{2}\lambda\|\mathbf{W}^\#\|^2 \\ &= \sum_i^N L(f(\mathbf{x}_i; \frac{\mathbf{W}^*}{\|\mathbf{W}^*\|}, y_i) + \frac{1}{2}\lambda\alpha^2\|\mathbf{W}^*\|^2 \\ &< \sum_i^N L(f(\mathbf{x}_i; \frac{\mathbf{W}^*}{\|\mathbf{W}^*\|}, y_i) + \frac{1}{2}\lambda\|\mathbf{W}^*\|^2 = \hat{\mathcal{L}}(\mathbf{W}^*), \end{aligned} \quad (22)$$

which leads to the contradiction with the assumption that  $\hat{\mathcal{L}}(\mathbf{W}^*)$  is smallest. The similar conclusion can be found with WS.

#### 4.2 Training Instability

When given enough training iterations, the objective (Eq. (10)) will continuously push the length of the weight (i.e.,  $k$ ) to 0. The effective learning rate is *inversely proportional* to the weight length, which is much easier to cause the floating point overflow and lead to a failed training.

Specifically, we find that improper selection of  $\lambda$  would actually cause the instability in training. When we choose a slightly larger  $\lambda$  for an optimizer, some of the weights in the network will quickly converge to 0, making the effective learning rate close to infinity. Thereby the numerical gradient updates are beyond the representation of float in the computational resource, resulting in a training failure. Table 2 shows the impact of  $\lambda$  on network training with

TABLE 1

Top-1/5 Accuracy (%) via single  $224 \times$  crop on ImageNet validation set of ResNet-50 with warmup and WS-ResNet-50.

Type	Top-1/5 Acc (%)
ResNet-50	76.54/93.07
ResNet-50 + warmup	76.81/93.20
WS-ResNet-50	76.74/93.28

TABLE 2

Top-1 Accuracy (%) via single  $224 \times$  crop on ImageNet validation set of different weight decay  $\lambda$  settings for convolution in ResNet-50, WN-convolution in WN-ResNet-50, and WS-convolution in WS-ResNet-50.  $\lambda$  of other parts (BN and fc layers) is kept with  $1e-4$ . We demonstrate the results of two widely used optimizers: SGD (with momentum) and Adam. “—” denotes a failed training due to the gradient float overflow.

Top-1 Acc (%) w.r.t. $\lambda$		1e-2	1e-3	1e-4	1e-5	0
ResNet-50	SGD	47.67	74.12	76.54	74.80	72.65
	Adam	19.42	35.68	52.97	63.46	72.50
WN-ResNet-50	SGD	—	—	76.44	74.65	72.86
	Adam	—	—	—	—	72.34
WS-ResNet-50	SGD	—	—	76.74	74.70	72.92
	Adam	—	—	—	—	72.85

two widely used optimizers SGD [41] with momentum and Adam [22], where it is much easier to have a failed training for networks with weights normalized. Moreover, the adaptive gradient method (e.g., Adam) even fail to have a successful training unless we discard the weight decay by setting  $\lambda = 0$ . Since Adam will calculate the square of the gradient during the optimization process, it is more likely to encounter the risk of floating point overflow.

To better illustrate the gradient float overflow risks, we demonstrate the maximal  $\frac{1}{\|\mathbf{W}\|}$  for WN-ResNet-50 and  $\frac{1}{\sqrt{\frac{1}{n}\|\mathbf{W}-\bar{\mathbf{W}}\|^2}}$  for WS-ResNet-50 in the case of  $\lambda = 1e-3$  (for all corresponding convolutional layers) during SGD optimization in Fig. 6, where the maximal  $\frac{1}{\|\mathbf{W}\|}$  (or  $\frac{1}{\sqrt{\frac{1}{n}\|\mathbf{W}-\bar{\mathbf{W}}\|^2}}$ ) goes exponentially large and eventually leads to the gradient float overflow after about 50k iterations.

One may argue that the practical implementation of WS [36] already considers the risk of float overflow in the original paper by adding a positive constant  $\epsilon$  in the denominator of standardization:

$$\mathbf{W}' = \frac{\mathbf{W} - \bar{\mathbf{W}}}{\sqrt{\frac{1}{n}\|\mathbf{W} - \bar{\mathbf{W}}\|^2} + \epsilon}, \quad (23)$$

here we clarify that only adding  $\epsilon$  in the standardization part is definitely not enough for preventing the gradient float overflow problem. To explain, we can derive the gradient of  $W'_i$  w.r.t.  $W_i$  according to Eq. (23):

$$\frac{1}{\sqrt{\frac{1}{n}\|\mathbf{W}^t - \bar{\mathbf{W}}^t\|^2} + \epsilon} \left( 1 - \frac{1}{n} - \frac{\frac{1}{n}(W_i^t - \bar{W}^t)^2}{(\sqrt{\frac{1}{n}\|\mathbf{W}^t - \bar{\mathbf{W}}^t\|^2} + \epsilon)\sqrt{\frac{1}{n}\|\mathbf{W}^t - \bar{\mathbf{W}}^t\|^2}} \right), \quad (24)$$

where the individual standard deviation term  $\sqrt{\frac{1}{n}\|\mathbf{W}^t - \bar{\mathbf{W}}^t\|^2}$  still appears in the denominator and the gradient float overflow can still take place consequently. Therefore, it is necessary to propose a different approach to address the problem.

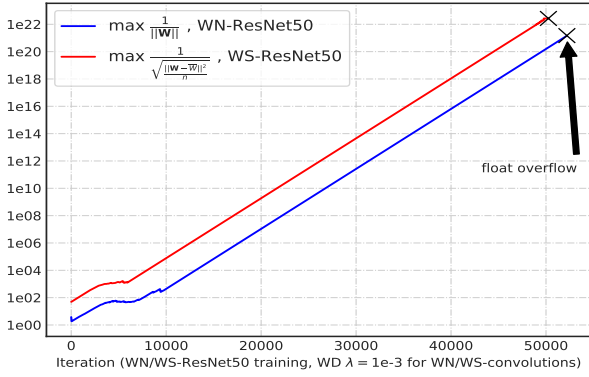


Fig. 6. Illustration of gradient float overflow problem over training iterations of WN-ResNet-50 and WS-ResNet-50. We use the maximal reciprocal length (or standard deviation) of weights as statistics.

## 5 METHODS

This section describes our proposed  $\epsilon$ -shifted  $L_2$  Regularizer in order to address the above problems.

### 5.1 $\epsilon$ -shifted $L_2$ Regularizer

As stated in Sec 3.2, when training weight normalization family with weight decay term, it is suggested to design a mechanism which can successfully prevent the network weights from being extremely small towards 0 in any case of hyper-parameter or optimizer. Given such an insight, we start from investigating the lack of global minimum problem, where Eq. (10) is again reviewed:

$$\begin{aligned} \min_{\mathbf{A}, k} \hat{\mathcal{L}}(\mathbf{A}, k) &= \sum_i^N L(f(\mathbf{x}_i; \mathbf{A}), y_i) + \frac{1}{2} \lambda k^2, \\ \text{s.t. } \|\mathbf{A}\| &= 1, k > 0. \end{aligned}$$

We notice that the central reason for the missing of global minimum is the regular  $L_2$  term:  $\frac{1}{2} \lambda k^2$  ( $k > 0$ ). During optimization, this term will have a large chance to continuously drive  $k$  ( $k = \|\mathbf{W}\|$ ) infinitely close to 0, making the gradient  $\frac{\partial \mathcal{L}}{\partial W_i} = \frac{\eta}{\|\mathbf{W}\|} (1 - \frac{(W_i^t)^2}{\sum_j^n (W_j^t)^2}) \frac{\partial \mathcal{L}}{\partial A_i}$  to infinity and thus leading to training failures. To avoid such risks, we propose the  $\epsilon$ -shifted  $L_2$  Regularizer, which constrains  $k$  from being too small by a positive constant  $\epsilon$ :

$$\begin{aligned} \min_{\mathbf{A}, k} \hat{\mathcal{L}}(\mathbf{A}, k) &= \sum_i^N L(f(\mathbf{x}_i; \mathbf{A}), y_i) + \frac{1}{2} \lambda (k - \epsilon)^2, \\ \text{s.t. } \|\mathbf{A}\| &= 1, k > 0, \end{aligned} \quad (25)$$

For the case of WS, given the standard deviation  $v > 0$ , by modifying Eq. (13) we have:

$$\begin{aligned} \min_{\mathbf{B}, m, v} \hat{\mathcal{L}}(\mathbf{B}, m, v) &= \sum_i^N L(f(\mathbf{x}_i; \mathbf{B}), y_i) + \frac{1}{2} \lambda n (m^2 + (v - \epsilon)^2), \\ \text{s.t. } \|\mathbf{B}\| &= n, v > 0. \end{aligned} \quad (26)$$

### 5.2 Guarantee of Global Minimum

Thanks to the introduction of  $\epsilon$ -shifted  $L_2$  Regularizer, the modified loss objective (i.e., Eq. (25) and Eq. (26)) now can guarantee the existence of global minimum. For WN, suppose

$\mathbf{A}^*$  is the optimal solution to  $\min_{\mathbf{A}} \sum_i^N L(f(\mathbf{x}_i; \mathbf{A}), y_i)$ , then the global minimal solution of  $\hat{\mathcal{L}}(\mathbf{A}, k)$  is  $\mathbf{A} = \mathbf{A}^*, k = \epsilon$ . Therefore, the minimized objective is equal to  $\sum_i^N L(f(\mathbf{x}_i; \mathbf{A}^*), y_i)$ , where the additional  $\epsilon$ -shifted  $L_2$  Regularizer  $\frac{1}{2} \lambda (k - \epsilon)^2$  is utilized during optimization for mainly *two* important purposes: 1) controlling the effective learning rate to help networks converge, and 2) preventing the magnitude of weights from being too small and thus avoiding the gradient float overflow and training failures.

### 5.3 Dynamic Decay Mechanism

In addition, we find that  $\epsilon$ -shifted  $L_2$  Regularizer has the function of dynamically adjusting the decay coefficient according to the current magnitude of training weights during optimization. In the case of WN, the  $\epsilon$ -shifted  $L_2$  Regularizer term is  $\frac{1}{2} \lambda (\sqrt{\sum_j^n W_j^2} - \epsilon)^2$ , whose gradient formula for  $W_i$  is:

$$\lambda (1 - \frac{\epsilon}{\sqrt{\sum_j^n W_j^2}}) W_i = \lambda (1 - \frac{\epsilon}{\|\mathbf{W}\|}) W_i. \quad (27)$$

It can also be regarded as an adaptive version of traditional weight decay (i.e.,  $\lambda W_i$ ), which uses the dynamic magnitude of training weights  $\|\mathbf{W}\|$  to slightly adjust the decay ratio:  $\lambda \rightarrow \lambda (1 - \frac{\epsilon}{\|\mathbf{W}\|})$ . When the  $\|\mathbf{W}\|$  is relatively large,  $(1 - \frac{\epsilon}{\|\mathbf{W}\|})$  will also be relatively large, meaning that we will use a larger factor to shrink the larger weights, and it is reasonably intuitive. It probably explains why applying  $\epsilon$ -shifted  $L_2$  Regularizer can have slight improvements in our experiments.

For the case of WS, the gradient formula of  $\frac{1}{2} \lambda n (m^2 + (v - \epsilon)^2)$  w.r.t.  $W_i$  is:

$$\lambda ((1 - \frac{\epsilon}{\sqrt{\frac{\|\mathbf{W} - \bar{\mathbf{W}}\|^2}{n}}}) W_i + \frac{\epsilon}{\sqrt{\frac{\|\mathbf{W} - \bar{\mathbf{W}}\|^2}{n}}} \bar{W}), \quad (28)$$

where the similar analysis can be conducted.

## 6 EXPERIMENTS

In this section, we conduct extensive experiments on both classification and detection tasks to validate the effectiveness of the proposed  $\epsilon$ -shifted  $L_2$  Regularizer.

### 6.1 Experimental Settings

To validate the effectiveness of the proposed  $\epsilon$ -shifted  $L_2$  Regularizer, we conduct comprehensive experiments on the ImageNet [5]/CIFAR-100 [23] classification dataset and COCO [27] detection dataset accordingly. For fair comparisons, all the experiments are run under a unified pytorch [35] framework, including results of every baseline model. More details can be referred in our public code base: <https://github.com/implus/PytorchInsight>. We mainly conduct experiments based on the state-of-the-art Weight Normalization (WN) [39] and Weight Standardization (WS) [36] from the weight normalization family.

**ImageNet classification:** The ILSVRC 2012 classification dataset [5] contains 1.2 million images for training, and 50K for validation, from 1K classes. The training settings for large models are kept similar with [25], except that we set the weight decay ratio  $\lambda$  to 0 for all the bias part in

TABLE 3

Top-1 Accuracy (%) via single  $224 \times$  crop on ImageNet validation set of different weight decay  $\lambda$  settings for WN-convolution in WN-ResNet-50 and WS-convolution in WS-ResNet-50.  $\lambda$  of other parts (BN and fc layers) is kept with  $1e-4$ . We demonstrate the results of two widely used optimizers: SGD (with momentum) and Adam. “+  $\epsilon$ ” denotes the use of  $\epsilon$ -shifted  $L_2$  regularizer.

Top-1 Acc (%) w.r.t. $\lambda$		1e-2	1e-3	1e-4	1e-5
WN-ResNet-50 + $\epsilon$	SGD	71.86	75.31	76.52	74.63
	Adam	64.31	64.47	65.92	68.23
WS-ResNet-50 + $\epsilon$	SGD	72.15	75.68	76.86	74.99
	Adam	64.56	64.71	66.17	68.45

networks [14], which generally improves about 0.2% over all the baselines in this paper. We train networks on the training set and report the Top-1 (and Top-5) accuracies on the validation set with single  $224 \times 224$  central crop. For data augmentation, we follow the standard practice [42] and perform the random-size cropping and random horizontal flipping. All networks are trained with naive softmax cross entropy without label-smoothing regularization [43]. We train all the architectures from scratch by SGD [41] or Adam [22], [31]. SGD is with weight decay 0.0001 and momentum 0.9 for 100 epochs, starting from learning rate 0.1 and decreasing it by a factor of 10 every 30 epochs. Adam keeps the default settings with learning rate 0.001,  $\beta_1 = 0.9$ ,  $\beta_2 = 0.999$ . The total batch size is set as 256 and 8 GPUs (32 images per GPU) are utilized for training. The default weight initialization strategy is in [11], where we use a ‘fan\_out’ mode specifically. The training settings for small models (i.e., ShuffleNet [33], [53] and MobileNet [16], [40]) are slightly different according to the references of their original papers [16], [53]: the default weight decay is  $4e-5$  with a number of total epochs 300. Warmup [9], cosine learning rate decay [29], label smoothing [43] and no weight decay on all depthwise convolutional/BN layers [21] are as well applied. One should notice that small models are more difficult to train with higher accuracy, so in many papers of small models [16], [53], the authors usually take these training tricks as described above. Also importantly, the weight normalization family should not be applied on the *depthwise convolution* (common in those small architectures) in practice since the number of parameters in each normalized group is too small, otherwise we will observe severe performance degradations. In order to make a fair comparison, especially to make the performance of our reimplemented baseline reach the accuracy of the reported ones in the referenced paper, we use these tips in the training of all small models. Note that in our experiments, only those normalized weights are trained with  $\epsilon$ -shifted  $L_2$  Regularizer, others (BN and fc layer weights) are kept with traditional weight decay term (if it exists) since they donot suffer from these problems.

**CIFAR-100 classification:** The CIFAR-100 dataset [23] consists of colored natural images with  $32 \times 32$  pixels, where the images are drawn from 100 classes. The training and test set contain 50K and 10K images, respectively. Apart from the standard data augmentation scheme that is widely used in the dataset [13], [18], [46], we also add two recent popular methodologies namely cutout [6] and mixup [52] to further reduce the overfitting risks, where we keep their respective

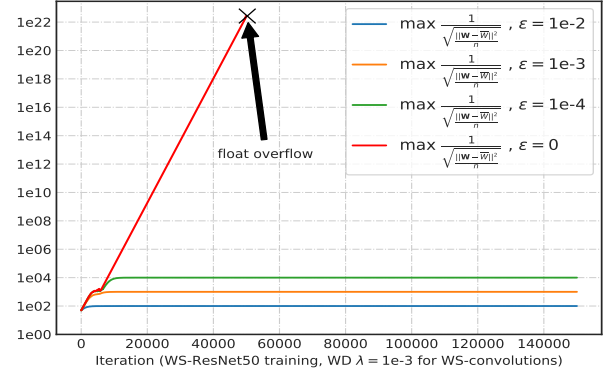


Fig. 7. Illustration of training stability over training iterations of WS-ResNet-50 under  $\lambda = 1e-3$ . The shifted  $\epsilon$  in fact limits the range of gradient float and thus greatly ensures the training stability. “ $\epsilon = 0$ ” denotes the training of WS-ResNet-50 with traditional weight decay.

TABLE 4

Top-1 Accuracy (%) via single  $224 \times$  crop on ImageNet validation set of different  $\epsilon$  settings for WS-convolution in WS-ResNet-50 under  $\lambda = 1e-4$ .  $\epsilon = 0$  denotes the baseline without the use of  $\epsilon$ -shifted  $L_2$  regularizer.

Top-1 Acc (%) w.r.t. $\epsilon$	0	1e-2	1e-3	1e-4	1e-5
WS-ResNet-50	76.74	76.60	<b>76.86</b>	<b>76.84</b>	76.71

default hyper-parameters in training. For preprocessing, we normalize the data using the channel means and standard deviations. The networks are trained with batch size 64 on one GPU. The training is with weight decay 0.0005 and momentum 0.9 for 300 epochs, starting from learning rate 0.05, which is decreased at 150th and 225th epoch by a factor of 10.

**COCO detection:** The COCO 2017 [27] dataset is comprised of 118k images in train set, and 5k images in validation set. We follow the standard setting [10] of evaluating object detection via the standard mean Average-Precision (AP) and mean Average-Recall (AR) [37] scores at different box IoUs or object scales, respectively. We use the standard configuration of Cascade R-CNN [3] with FPN [26] and ResNet as the backbone architecture. The input images are resized such that their shorter side is of 800 pixels. We trained on 8 GPUs with 2 images per GPU. The backbones of all models are pretrained on ImageNet classification, then all layers except for c1 and c2 are jointly finetuned with detection heads. The end-to-end training introduced in [37] is adopted for our implementation, which yields better results. We utilize the conventional finetuning setting [37] by fixing the learning parameters of BN layers. All models are trained for 20 epochs using Synchronized SGD with a weight decay of  $1e-4$  and momentum of 0.9. The learning rate is initialized to 0.02, and decays by a factor of 10 at the 16th and 19th epochs. The choice of hyper-parameters also follows the latest release of the Cascade R-CNN benchmark [4]. For more experiments with other detector frameworks, e.g., Faster R-CNN [37] and Mask R-CNN [10], we exactly follow the official settings of the baselines described in [4].

## 6.2 Training Stability

The proposed  $\epsilon$ -shifted  $L_2$  Regularizer ensures the global minimal solution of the magnitude of weights, which is also



TABLE 5

Accuracy via single  $224 \times$  crop on ImageNet validation set of different backbones using SGD.  $\lambda = 1e-4$  denotes the results of large models and  $\lambda = 4e-5$  denotes the results of small models. The “WS” denotes the conventional weight decay training of WS-equipped networks. “WS +  $\epsilon$ ” denotes the introduction of  $\epsilon$ -shifted  $L_2$  regularizer as objectives for training the WS-equipped networks.

Top-1 Acc (%) $\lambda = 1e-4$	baseline	WS	WS + $\epsilon$
ResNet-50 [12]	76.54	76.74	<b>76.86</b>
ResNet-101 [12]	78.17	78.07	<b>78.29</b>
ResNeXt-50 [49]	77.64	77.76	<b>77.88</b>
ResNeXt-101 [49]	78.71	78.68	<b>78.80</b>
SE-ResNet-101 [17]	78.43	78.65	<b>78.75</b>
DenseNet-201 [18]	77.54	77.56	<b>77.59</b>
Top-1 Acc (%) $\lambda = 4e-5$	baseline	WS	WS + $\epsilon$
ShuffleNetV1 1x (g=8) [53]	67.62	67.84	<b>68.09</b>
ShuffleNetV2 1x [33]	69.64	69.66	<b>69.70</b>
MobileNetV1 1x [16]	73.55	73.56	<b>73.60</b>
MobileNetV2 1x [40]	73.14	73.17	<b>73.22</b>

able to prevent the weights from being too small and thus avoids the gradient float overflow risks. To verify this, we traverse the hyper-parameter  $\lambda$  in a large scope and employ  $\epsilon$ -shifted  $L_2$  Regularizer to train networks with weight normalization family (namely, WN and WS), yielding the results in Table 3. In comparison with Table 2, we find that  $\epsilon$ -shifted  $L_2$  Regularizer not only greatly improves the stability of training, i.e., no matter how the hyper-parameter  $\lambda$  changes, the optimization can finally converge to a good solution with no cases of training failures for any type of optimizer, but also slightly boosts the generalization performance in those comparable cases of  $\lambda = 1e-4$  and  $1e-5$ .

In our experiments, we empirically find that, with  $\epsilon$ -shifted  $L_2$  Regularizer applied, the magnitude of the training weights will actually be controlled in the range of greater than or equal to  $\epsilon$  during the entire optimization. To give a better illustration, we depict the maximal  $\frac{1}{\sqrt{\|w - \bar{w}\|^2}}$  for WS-ResNet-50 with  $\epsilon$ -shifted  $L_2$  Regularizer and  $\lambda = 1e-3$  over its training iterations, and vary  $\epsilon = 1e-2, 1e-3, 1e-4$ . For a better comparison, we also plot the curve without  $\epsilon$ -shifted  $L_2$  Regularizer (i.e.,  $\epsilon = 0$ ), which is exactly the red curve in Fig. 6. As can be seen in Fig. 7, the shifted  $\epsilon$  limits the range of gradient float in fact and thus successfully prevents the training failures.

### 6.3 Parameter Sensitivity

In this subsection, we are interested in the selection of  $\epsilon$  as it is the only parameter of the proposed regularizer. To investigate the sensitivity of the hyper-parameter  $\epsilon$ , we traverse its range in  $[1e-2, 1e-3, 1e-4, 1e-5]$  for training WS-ResNet-50 under  $\lambda = 1e-4$ , shown in Table 4. It is demonstrated that the choice of  $\epsilon$  is robust to the final generalization performance, where more appropriate selections (i.e.,  $\epsilon = 1e-3$  and  $1e-4$ ) can consistently bring slight improvements of accuracy.

### 6.4 Extension to More Architectures

Further, we apply  $\epsilon$ -shifted  $L_2$  regularizer to more state-of-the-art network structures [12], [16], [17], [18], [33], [40], [49],

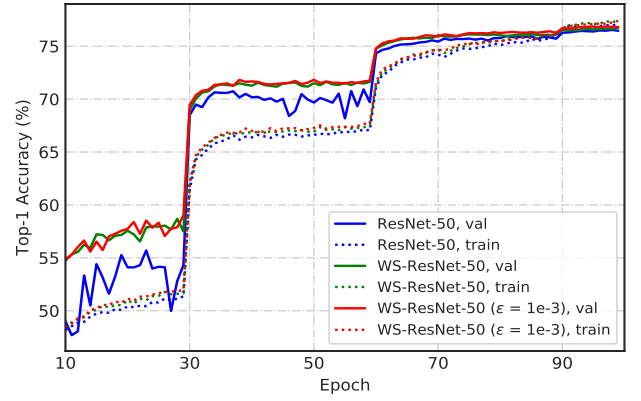


Fig. 8. The training and validation curves of (WS-)ResNet-50 on ImageNet dataset. It is observed that the  $\epsilon$ -shifted  $L_2$  regularizer maintains the property of faster convergence.

TABLE 6

Top-1 Accuracy (%) on CIFAR-100 validation set. For the type, the “WS” denotes the conventional weight decay training of WS-equipped networks. “WS +  $\epsilon$ ” denotes the introduction of  $\epsilon$ -shifted  $L_2$  regularizer as objectives for training the WS-equipped networks. The numbers in parentheses represent the absolute promotion of baseline.

Type	Backbone	Top-1 Acc (%)
baseline	ResNeXt29-16x64d [49]	83.68
WS	WS-ResNeXt29-16x64d [49]	83.48
WS + $\epsilon$	WS-ResNeXt29-16x64d [49]	<b>84.39 (+0.71)</b>
baseline	SE-ResNeXt29-16x64d [17]	84.66
WS	WS-SE-ResNeXt29-16x64d [17]	84.55
WS + $\epsilon$	WS-SE-ResNeXt29-16x64d [17]	<b>84.87 (+0.21)</b>

[53] and compare it to the original baseline and traditional WS version using SGD. For the WS-equipped networks, we set hyper-parameters  $\lambda = 1e-4$  and search  $\epsilon \in \{1e-3, 1e-5, 1e-8\}$  to report our results. As can be seen from Table 5, while keeping excellent training stability,  $\epsilon$ -shifted  $L_2$  regularizer also achieves very competitive results, both for large and small models. We also empirically demonstrate that the convergence can still be speeded up over the original baseline by  $\epsilon$ -shifted  $L_2$  regularizer in Fig. 8.

### 6.5 Extension to Other Datasets

We further verify whether the effectiveness of  $\epsilon$ -shifted  $L_2$  regularizer can generalise to datasets beyond ImageNet. Here we choose the widely used CIFAR-100, and mainly validate it based on two strong backbones: ResNeXt29-16x64d [49] and SE-ResNeXt29-16x64d [17]. In the experiments, we typically find that  $\epsilon = 1e-2$  would bring consistent gains for  $\epsilon$ -shifted  $L_2$  regularizer. From the results in Table 6, we are surprised to discover that in CIFAR-100 for (SE-)ResNeXt29-16x64d, the performance of WS-equipped networks *without*  $\epsilon$ -shifted  $L_2$  regularizer has slightly declined when compared to the baseline. Instead,  $\epsilon$ -shifted  $L_2$  regularizer can still improve the recognition accuracy over the baseline models, which demonstrates its high potentials for practice. Furthermore, the training and validation curves of  $\epsilon$ -shifted  $L_2$  regularizer (i.e., WS +  $\epsilon$ ) can converge significantly faster than baseline and WS, which is depicted in Fig. 9.

TABLE 7

Detection results on COCO 2017 [27] validation set using Cascade R-CNN [3] and FPN [26] with (WS-)ResNet-50 and (WS-)ResNet-101 as backbones. The “WS” denotes the conventional weight decay training of WS-equipped networks. “WS +  $\epsilon$ ” denotes the introduction of  $\epsilon$ -shifted  $L_2$  regularizer as objectives for training the WS-equipped networks. The numbers in parentheses represent the absolute promotion of baseline.

Cascade R-CNN [3]	Backbone	AP	AP <sub>.5</sub>	AP <sub>.75</sub>	AP <sub>s</sub>	AP <sub>m</sub>	AP <sub>l</sub>	AR <sub>s</sub>	AR <sub>m</sub>	AR <sub>l</sub>
baseline	ResNet-50	41.1	59.3	44.8	22.6	44.5	54.9	33.2	58.8	70.7
WS	WS-ResNet-50	41.6	60.1	45.2	<b>23.4</b>	44.7	<b>55.6</b>	<b>34.2</b>	58.2	71.0
WS + $\epsilon$	WS-ResNet-50	<b>41.8 (+ 0.7)</b>	<b>60.2</b>	<b>45.5</b>	<b>23.4</b>	<b>45.0</b>	55.4	33.9	<b>58.9</b>	<b>71.8</b>
baseline	ResNet-101	42.6	60.9	46.4	23.7	46.1	56.9	34.5	59.8	72.0
WS	WS-ResNet-101	43.2	61.6	47.2	<b>24.8</b>	46.7	57.8	<b>34.8</b>	59.7	72.2
WS + $\epsilon$	WS-ResNet-101	<b>43.5 (+ 0.9)</b>	<b>61.7</b>	<b>47.5</b>	23.9	<b>47.1</b>	<b>58.4</b>	33.4	<b>60.2</b>	<b>72.4</b>

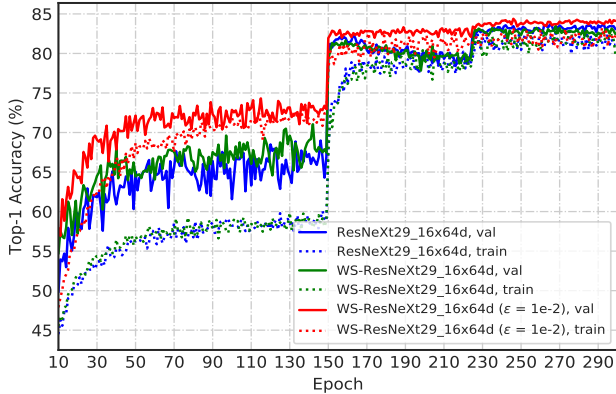


Fig. 9. The training and validation curves of different training types of (WS-)ResNeXt29-16x64d on CIFAR-100 dataset. The  $\epsilon$ -shifted  $L_2$  regularizer significantly speeds up the convergence of WS-ResNeXt29-16x64d.

## 6.6 Extension to Detection Tasks

We are also interested that whether  $\epsilon$ -shifted  $L_2$  regularizer can still work in some downstream tasks beyond image classification, e.g., object detection. Here we choose one of the most advanced object detector: Cascade R-CNN [3] for evaluation and conduct comprehensive experiments on COCO datasets [27]. The pretrained models with best performance are utilized for initialization of detector backbones. From Table 7, it is observed that  $\epsilon$ -shifted  $L_2$  regularizer has the potential to significantly boost the overall performance of the detectors, especially for large backbone ResNet-101 model. Specifically, it purely improves nearly absolute 1% AP based on the original baseline, and outperforms the baseline in all aspects of other metrics, i.e., AP/AR with different object scales. We also conduct experiments on other state-of-the-art detectors, and observe the consistent improvements in Table 8, which demonstrates its wide usage.

## 6.7 Important Practices for Weight Normalization Family

In the original papers of weight normalization family [36], [39], the authors rarely discuss where to use WN or WS in deep neural networks. Their default mode is to place WN or WS on all conventional convolutional layers, while the BN and fc layers will not participate in WN/WS operations. However, in our practice, it is not always the best option. For depthwise convolutions or group convolutions with very few parameters in each group, using WN or WS can result

TABLE 8

Detection results for Faster R-CNN [37] and Mask R-CNN [10] with FPN [26] on COCO 2017 [27] validation set. “WS +  $\epsilon$ ” denotes the introduction of  $\epsilon$ -shifted  $L_2$  regularizer as objectives for training the WS-equipped networks.

Faster R-CNN [37]	Backbone	AP	AP <sub>.5</sub>	AP <sub>.75</sub>
baseline	ResNet-50	37.7	59.3	<b>41.1</b>
WS + $\epsilon$	WS-ResNet-50	<b>37.9</b>	<b>59.7</b>	40.9
baseline	ResNet-101	39.4	60.7	43.0
WS + $\epsilon$	WS-ResNet-101	<b>39.8</b>	<b>60.8</b>	<b>43.5</b>
Mask R-CNN [10]	Backbone	AP	AP <sub>.5</sub>	AP <sub>.75</sub>
baseline	ResNet-50	38.6	60.0	41.9
WS + $\epsilon$	WS-ResNet-50	<b>38.9</b>	<b>60.1</b>	<b>42.2</b>
baseline	ResNet-101	40.4	61.6	44.2
WS + $\epsilon$	WS-ResNet-101	<b>41.1</b>	<b>62.2</b>	<b>45.0</b>

TABLE 9

Top-1/5 accuracy (%) via single  $224 \times$  crop on ImageNet validation set of using or not using WS on the depthwise convolutions. “w/” denotes using WS and “wo/” denotes not using WS. All other parts are kept the same.

backbone	w/	wo/
WS-ShuffleNetV2 1x [33]	63.79/84.63	<b>69.66/88.76</b>
WS-MobileNetV2 1x [40]	69.74/89.18	<b>73.17/91.05</b>

in a severe degradation of performance both in train and test set. We speculate that when normalizing only a few parameters, since the set of parameters itself has very few degrees of freedom, normalization or standardization will further reduce the degrees of freedom, leading to extremely limited representation ability. One example is the learnable parameter of BN. It is essentially equivalent to a  $1 \times 1$  depthwise convolution, where each parameter group only contains one variable. If we normalize it, it then becomes a fixed constant (in case of WN), which definitely can not learn the effect of scaling features.

The experiments which we have conducted above mainly avoid these risks. For example, in the small models like ShuffleNetV2, MobileNetV1 and MobileNetV2, we do not apply WS on the depthwise convolutions. And for ShuffleNetV1, it is suggested not to equip the group convolutions with WS. To be specific, we list the results of using or not using WS on the depthwise convolutions in Table 9. It can be observed that whether to use WS on the depthwise convolution can result in a very large performance gap.

## 7 CONCLUSIONS

In this paper, we first review the disharmony between weight normalization family and weight decay, i.e., the counter-intuitive under-fitting risk caused by weight decay on the normalized weights. Then, we theoretically answer this question by two evidences: 1) weight decay doesn't change the optimization goal and 2) it ensures the appropriate effective learning rate for better convergence. After that, we expose the detailed problems via introducing fixed weight decay term in the loss objective, including missing of global minimum and training instability. Finally, to solve these potential problems, we propose  $\epsilon$ -shifted  $L_2$  regularizer that shifts the  $L_2$  objective by a positive constant  $\epsilon$ . The shifted  $\epsilon$  prevents network weights from being too small, where the gradient float overflow risks can be avoided directly. Comprehensive analyses demonstrate that the proposed  $\epsilon$ -shifted  $L_2$  regularizer successfully guarantees the global minimum and significantly improves the training stability, whilst maintaining superior performance.

## REFERENCES

- [1] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.
- [2] Léon Bottou. Large-scale machine learning with stochastic gradient descent. In *COMPSTAT*. 2010.
- [3] Zhaowei Cai and Nuno Vasconcelos. Cascade r-cnn: Delving into high quality object detection. In *CVPR*, 2018.
- [4] Kai Chen, Jiaqi Wang, Jiangmiao Pang, Yuhang Cao, Yu Xiong, Xiaoxiao Li, Shuyang Sun, Wansen Feng, Ziwei Liu, Jiarui Xu, Zheng Zhang, Dazhi Cheng, Chenchen Zhu, Tianheng Cheng, Qijie Zhao, Buyu Li, Xin Lu, Rui Zhu, Yue Wu, Jifeng Dai, Jingdong Wang, Jianping Shi, Wanli Ouyang, Chen Change Loy, and Dahua Lin. MMDetection: Open mmlab detection toolbox and benchmark. *arXiv preprint arXiv:1906.07155*, 2019.
- [5] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009.
- [6] Terrance DeVries and Graham W Taylor. Improved regularization of convolutional neural networks with cutout. *arXiv preprint arXiv:1708.04552*, 2017.
- [7] Igor Gitman and Boris Ginsburg. Comparison of batch normalization and weight normalization algorithms for the large-scale image classification. *arXiv preprint arXiv:1709.08145*, 2017.
- [8] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *NeurIPS*, 2014.
- [9] Priya Goyal, Piotr Dollár, Ross Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. Accurate, large minibatch sgd: Training imagenet in 1 hour. *arXiv preprint arXiv:1706.02677*, 2017.
- [10] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *ICCV*, 2017.
- [11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *ICCV*, 2015.
- [12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016.
- [13] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *ECCV*, 2016.
- [14] Tong He, Zhi Zhang, Hang Zhang, Zhongyue Zhang, Junyuan Xie, and Mu Li. Bag of tricks for image classification with convolutional neural networks. In *CVPR*, 2019.
- [15] Elad Hoffer, Ron Banner, Itay Golan, and Daniel Soudry. Norm matters: efficient and accurate normalization schemes in deep networks. In *NeurIPS*, 2018.
- [16] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.
- [17] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *CVPR*, 2018.
- [18] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *CVPR*, 2017.
- [19] Lei Huang, Xianglong Liu, Yang Liu, Bo Lang, and Dacheng Tao. Centered weight normalization in accelerating training of deep neural networks. In *ICCV*, 2017.
- [20] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.
- [21] Xianyan Jia, Shutao Song, Wei He, Yangzihao Wang, Haidong Rong, Feihu Zhou, Liqiang Xie, Zhenyu Guo, Yuanzhou Yang, Liwei Yu, et al. Highly scalable deep learning training system with mixed-precision: Training imagenet in four minutes. *arXiv preprint arXiv:1807.11205*, 2018.
- [22] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.
- [23] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. Technical report, 2009.
- [24] Anders Krogh and John A Hertz. A simple weight decay can improve generalization. In *NeurIPS*, 1992.
- [25] Xiang Li, Xiaolin Hu, and Jian Yang. Spatial group-wise enhance: Improving semantic feature learning in convolutional networks. *arXiv preprint arXiv:1905.09646*, 2019.
- [26] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *CVPR*, 2017.
- [27] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *ECCV*, 2014.
- [28] Liyuan Liu, Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Jiawei Han. On the variance of the adaptive learning rate and beyond. *arXiv preprint arXiv:1908.03265*, 2019.
- [29] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016.
- [30] Ilya Loshchilov and Frank Hutter. Fixing weight decay regularization in adam. *arXiv preprint arXiv:1711.05101*, 2017.
- [31] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. 2019.
- [32] Ping Luo, Jiamin Ren, Zhanglin Peng, Ruimao Zhang, and Jingyu Li. Differentiable learning-to-normalize via switchable normalization. *arXiv preprint arXiv:1806.10779*, 2018.
- [33] Ningning Ma, Xiangyu Zhang, Hai-Tao Zheng, and Jian Sun. ShuffleNet v2: Practical guidelines for efficient cnn architecture design. In *ECCV*, 2018.
- [34] Andrew Y Ng. Feature selection,  $l_1$  vs.  $l_2$  regularization, and rotational invariance. In *ICML*, 2004.
- [35] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017.
- [36] Siyuan Qiao, Huiyu Wang, Chenxi Liu, Wei Shen, and Alan Yuille. Weight standardization. *arXiv preprint arXiv:1903.10520*, 2019.
- [37] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *NeurIPS*, 2015.
- [38] Sebastian Ruder. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*, 2016.
- [39] Tim Salimans and Durk P Kingma. Weight normalization: A simple reparameterization to accelerate training of deep neural networks. In *NeurIPS*, 2016.
- [40] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *CVPR*, 2018.
- [41] Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. On the importance of initialization and momentum in deep learning. In *ICML*, 2013.
- [42] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *CVPR*, 2015.
- [43] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *CVPR*, 2016.

- [44] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Instance normalization: The missing ingredient for fast stylization. *arXiv preprint arXiv:1607.08022*, 2016.
- [45] Twan Van Laarhoven. L2 regularization versus batch and weight normalization. *arXiv preprint arXiv:1706.05350*, 2017.
- [46] Wenhai Wang, Xiang Li, Jian Yang, and Tong Lu. Mixed link networks. *arXiv preprint arXiv:1802.01808*, 2018.
- [47] Yuxin Wu and Kaiming He. Group normalization. In *ECCV*, 2018.
- [48] Sitao Xiang and Hao Li. On the effects of batch and weight normalization in generative adversarial networks. *arXiv preprint arXiv:1704.03971*, 2017.
- [49] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *CVPR*, 2017.
- [50] Yang You, Igor Gitman, and Boris Ginsburg. Large batch training of convolutional networks. *arXiv preprint arXiv:1708.03888*, 2017.
- [51] Guodong Zhang, Chaoqi Wang, Bowen Xu, and Roger Grosse. Three mechanisms of weight decay regularization. *arXiv preprint arXiv:1810.12281*, 2018.
- [52] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*, 2017.
- [53] Xiangyu Zhang, Xinyu Zhou, Mengxiao Lin, and Jian Sun. Shufflenet: An extremely efficient convolutional neural network for mobile devices. In *CVPR*, 2018.