Understanding the Disharmony between Weight Normalization Family and Weight Decay

Xiang Li,[1,2] **Shuo Chen,**[1] **Jian Yang**[1*]
[1]DeepInsight@PCALab, Nanjing University of Science and Technology
[2]Momenta

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 $\boldsymbol{W}$ to $\boldsymbol{W}'$, which makes $\boldsymbol{W}'$ independent to the magnitude of $\boldsymbol{W}$. Surprisingly, $\boldsymbol{W}$ must be decayed during gradient descent, otherwise we will observe a severe underfitting 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) $\boldsymbol{W}' = \frac{\boldsymbol{W}}{||\boldsymbol{W}||}$ in the original loss function $\sum_i L(f(\boldsymbol{x}_i; \boldsymbol{W}'), y_i) + \frac{1}{2}\lambda||\boldsymbol{W}'||^2$, it is observed that the regularization term $\frac{1}{2}\lambda||\boldsymbol{W}'||^2$ will be canceled as a constant $\frac{1}{2}\lambda$ in the optimization objective. Therefore, to decay $\boldsymbol{W}$, we need to explicitly append: $\frac{1}{2}\lambda||\boldsymbol{W}||^2$. In this paper, we *theoretically* prove that $\frac{1}{2}\lambda||\boldsymbol{W}||^2$ improves optimization only by modulating the effective learning rate and fairly has no influence on generalization when the weight normalization family is compositely employed. Furthermore, we also expose several serious problems when introducing weight decay term to weight normalization family, including the missing of global minimum, training instability and sensitivity of initialization. To address these problems, we propose an Adaptive Weight Shrink (AWS) scheme, which gradually shrinks the weights during optimization by a dynamic coefficient proportional to the magnitude of the parameter. This simple yet effective method appropriately controls the effective learning rate, which significantly improves the training stability and makes optimization more robust to initialization.

Introduction

The normalization methodologies on *features* have made great progress in recent years, with the introduction of BN (Ioffe and Szegedy 2015), IN (Ulyanov, Vedaldi, and Lempitsky 2016), LN (Ba, Kiros, and Hinton 2016), GN (Wu and He 2018) and SN (Luo et al. 2018). 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 (He et al. 2016a; 2016b; Huang et al. 2017a; Wang et al. 2018) much easier to optimize, leading to robust solutions with favorable generalization performance.

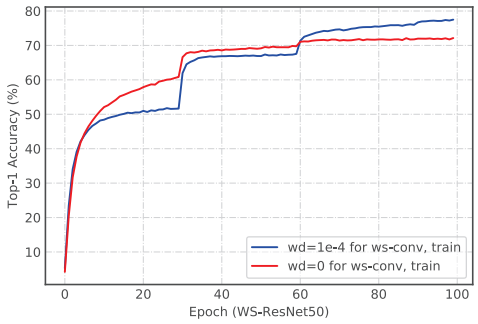

Figure 1: Counter-intuitive performance degradation (underfitting) by setting weight decay parameter λ to 0 for WS-equipped convolutions. Top-1 Accuracy via single 224× crop on the ImageNet training set is plotted.

Beyond *feature* normalization, there is an increasing interest on the normalization of network *weights*. Weight Normalization (WN) (Salimans and Kingma 2016) 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) (Goodfellow et al. 2014), Weight Normalization with Translated ReLU (Xiang and Li 2017) is shown to achieve superior results. Later, Centered Weight Normalization (CWN) (Huang et al. 2017b) further powers WN by additionally centering their input weights, ulteriorly improving the conditioning and convergence speed. Recently, very similar to CWN, Weight Standardization (WS) (Qiao et al. 2019) aims to standardize the weights with zero mean and unit variance. On the large-scale tasks (ImageNet (Deng et al. 2009) clas-

*Xiang Li, Shuo Chen and Jian Yang are with PCA Lab, Key Lab of Intelligent Perception and Systems for High-Dimensional Information of Ministry of Education, and Jiangsu Key Lab of Image and Video Understanding for Social Security, School of Computer Science and Engineering, Nanjing University of Science and Technology. (E-mail: (xiang.li.implus, shuochen, csjyang)@njust.edu.cn). Xiang Li is also a visiting scholar at Momenta. (Corresponding author: Jian Yang)

sification/COCO (Lin et al. 2014) 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 (Krogh and Hertz 1992). Specifically, we consider training a single-layer (note that the following conclusions can be easily extended to the multiple-layer one) neural network $f(\boldsymbol{x}; \boldsymbol{W}')$, where $\boldsymbol{x}, \boldsymbol{W}' \in \mathcal{R}^n$ with the following loss function to be optimized:

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

where $\hat{\mathcal{L}}$ contains task-related loss $L(\cdot, \cdot)$, input/label pair $(\boldsymbol{x}_i, y_i)$, and the regularization term $\frac{1}{2}\lambda||\boldsymbol{W}'||^2$ with a constant λ to balance against $L(\cdot, \cdot)$. For simplicity, we use weight normalization to re-parameterize $\boldsymbol{W}'$, regardless the learning of its length. By substituting $\boldsymbol{W}' = \frac{\boldsymbol{W}}{||\boldsymbol{W}||}$ in Eq. (1), we can get $\sum_i L(f(\boldsymbol{x}_i; \frac{\boldsymbol{W}}{||\boldsymbol{W}||}), y_i) + \frac{1}{2}\lambda$, which is equivalent to minimize the following function

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

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

$$\sum_i L(f(\boldsymbol{x}_i; \frac{\boldsymbol{W}-\overline{W}}{\sqrt{\frac{||\boldsymbol{W}-\overline{W}||^2}{n}}}), y_i), \quad (3)$$

where $\overline{W} = \frac{\sum_i W_i}{n}$. 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 ResNet50 (WS-ResNet50) on ImageNet dataset, i.e., setting weight decay ratio λ to 0 for all WS-equipped convolutions, we observe a severe degradation with significant performance drop in training set (Figure 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 $\boldsymbol{W}$'s weight decay term does not change the optimization goal. Therefore, weight decay loses its original role that finds a better generalized solution by introducing a different loss part against the task-related one. At the same time, basing on the derivation of the gradient formula of $\boldsymbol{W}$, 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, and empirically demonstrate how it adjusts the effective learning rate.

The current common and default operation (Salimans and Kingma 2016; Qiao et al. 2019) to optimize networks with weight normalization family is to continue to preserve the traditional decay term of $\boldsymbol{W}$ for better convergence that comes from ensuring the stable effective learning rate, i.e., to explicitly add $\frac{1}{2}\lambda||\boldsymbol{W}||^2$ on Eq. (2):

$$\sum_i L(f(\boldsymbol{x}_i; \frac{\boldsymbol{W}}{||\boldsymbol{W}||}), y_i) + \frac{1}{2}\lambda||\boldsymbol{W}||^2. \quad (4)$$

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 λ will easily lead to training failures due to gradient float overflow. Finally, Eq. (4) is also very sensitive to the initialization of parameter $\boldsymbol{W}$.

To address these problems, we propose a very simple yet effective Adaptive Weight Shrink (AWS) scheme, which gradually shrinks the weights during optimization by a dynamic coefficient proportional to the length of the parameter. By doing so, we are able to *discard* the fixed weight decay term in the loss expression and avoids a series of existing problems, which greatly improves the training stability and makes it robust to initialization. The effectiveness of our method is demonstrated by experiments on the large-scale ImageNet dataset.

To summarize our contributions:

- We thoroughly analyze the disharmony between weight normalization family and weight decay, and expose the serious problems caused by the optimization of weight decay term in the final loss objective.
- 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. The detailed mathematical expression of how to adjust the effective learning rate is also given.
- We propose a simple yet effective Adaptive Weight Shrink (AWS) scheme to overcome the disharmony between weight decay and weight normalization family, which improves the training stability and robustness to initialization whilst maintaining competitive performance.

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.

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 tools of variable decomposition. Specifically, we choose two representative methods from weight normalization family, namely WN and WS, and discuss each in turn.

In WN, we aim to prove that optimizing

$$\hat{\mathcal{L}}(\boldsymbol{W}) = \sum_i L(f(\boldsymbol{x}_i; \frac{\boldsymbol{W}}{||\boldsymbol{W}||}), y_i) + \frac{1}{2}\lambda||\boldsymbol{W}||^2, \quad (5)$$

is equal to optimizing

$$\mathcal{L}(\boldsymbol{W}) = \sum_i L(f(\boldsymbol{x}_i; \frac{\boldsymbol{W}}{||\boldsymbol{W}||}), y_i). \quad (6)$$

Specifically, let $\boldsymbol{A} = \frac{\boldsymbol{W}}{||\boldsymbol{W}||}$ and $k = ||\boldsymbol{W}||(k > 0)$, we can decompose the direction and length of $\boldsymbol{W}$ as two independent variables. Then the objectives of Eq. (5) and (6) can be rewritten as

$$\min_{\boldsymbol{A},k} \hat{\mathcal{L}}(\boldsymbol{A}, k) = \sum_i L(f(\boldsymbol{x}_i; \boldsymbol{A}), y_i) + \frac{1}{2}\lambda k^2,$$
$$s.t. \ \ ||\boldsymbol{A}|| = 1, k > 0, \quad (7)$$

and

$$\min_{\boldsymbol{A}} \mathcal{L}(\boldsymbol{A}) = \sum_i L(f(\boldsymbol{x}_i; \boldsymbol{A}), y_i), \ \ s.t. \ \ ||\boldsymbol{A}|| = 1. \quad (8)$$

Since $\boldsymbol{A}$ and k are two independent variables, we have

$$\min_{\boldsymbol{A},k} \hat{\mathcal{L}}(\boldsymbol{A}, k) = \min_{\boldsymbol{A}} \sum_i L(f(\boldsymbol{x}_i; \boldsymbol{A}), y_i) + \min_k \frac{1}{2}\lambda k^2$$
$$= \min_{\boldsymbol{A}} \mathcal{L}(\boldsymbol{A}) + \min_k \frac{1}{2}\lambda k^2. \quad (9)$$

Eq. (9) shows that minimizing $\hat{\mathcal{L}}$ actually contains the task of minimizing $\mathcal{L}$, which completes the proof. Similarly, in WS we can further decompose the mean and variance of $\boldsymbol{W}$ by letting $\boldsymbol{B} = \frac{\boldsymbol{W}-\overline{W}}{\sqrt{\frac{||\boldsymbol{W}-\overline{W}||^2}{n}}}, m = \overline{W}$ and $v = \sqrt{\frac{||\boldsymbol{W}-\overline{W}||^2}{n}}(v > 0)$, where $\boldsymbol{B}, m, v$ are also mutually independent. Again we can have

$$\min_{\boldsymbol{B},m,v} \hat{\mathcal{L}}(\boldsymbol{B}, m, v) = \min_{\boldsymbol{B}} \mathcal{L}(\boldsymbol{B}) + \min_{m,v} \frac{1}{2}\lambda n(m^2 + v^2). \quad (10)$$

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

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 $\boldsymbol{W}$ as (the deviate to Eq. (6)):

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{W}} = \frac{\partial \mathcal{L}}{\partial \boldsymbol{A}} * (\frac{\boldsymbol{1}}{||\boldsymbol{W}||} - \frac{\boldsymbol{W} * \boldsymbol{W}}{||\boldsymbol{W}||^3}), \quad (11)$$

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

$$W_i^{t+1} = W_i^t - \frac{\eta}{||\boldsymbol{W}^t||}(1 - \frac{(W_i^t)^2}{\sum_j (W_j^t)^2})\frac{\partial \mathcal{L}}{\partial A_i}. \quad (12)$$

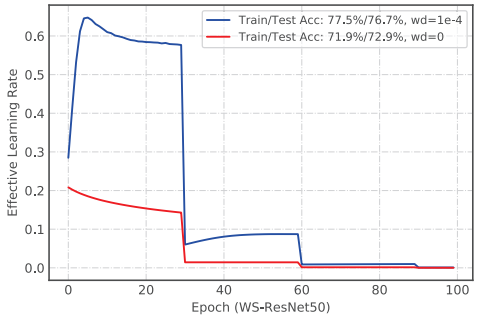

Figure 2: The comparisons of effective learning rate for different weight decay ratios. The blue curve ensures a larger effective learning rate which helps the network converge.

Suppose we fix the direction of $\boldsymbol{W}$ and vary the length $||\boldsymbol{W}||$, then $\boldsymbol{A} = \frac{\boldsymbol{W}}{||\boldsymbol{W}||}$ remains unchanged, thereby $\frac{\partial \mathcal{L}}{\partial A_i}$ is also fixed. Meanwhile, $(1 - \frac{(W_i^t)^2}{\sum_j (W_j^t)^2})$ term is as well fixed within a stable range $[0, 1]$. Consequently, the entire update step size can be determined by $\frac{\eta}{||\boldsymbol{W}||}$, which exactly controls the effective learning rate (Hoffer et al. 2018). *If we do not limit $||\boldsymbol{W}||$ during the update process, the weights can grow unbounded $||\boldsymbol{W}|| \to \infty$, and the effective learning rate goes to 0 ($\frac{\eta}{||\boldsymbol{W}||} \to 0$).* The similar analysis can be conducted in the case of WS, where one update step is:

$$W_i^{t+1} = W_i^t - \frac{\eta}{\sqrt{\frac{||\boldsymbol{W}^t-\overline{W^t}||^2}{n}}}(1 - \frac{1}{n} - \frac{(W_i^t-\overline{W^t})^2}{\sum_j (W_j^t-\overline{W^t})^2})\frac{\partial \mathcal{L}}{\partial B_i}, \quad (13)$$

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

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 WS-ResNet50 throughout the training process in Figure 2, which represents the applications of 0 and 1e-4 weight decay to the corresponding convolutional layers respectively.

Further, we can theoretically prove that, using SGD (Bottou 2010; Sutskever et al. 2013), the training trajectory of $\hat{\mathcal{L}}$ (Eq. (5) with weight decay) can be completely reproduced by simply scaling the learning rate at each step when optimizing $\mathcal{L}$ (Eq. (6) without weight decay). Here we focus on the normalized variables to represent the training trajectory, e.g., $\boldsymbol{A}$ (or $\boldsymbol{B}$), as they are the ultimate weights with which networks use to operate. In the case of WN, we suppose optimizing $\hat{\mathcal{L}}(\boldsymbol{A}, k)$ and $\mathcal{L}(A)$ take T steps in total respectively, 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., $\boldsymbol{W}^{\hat{\mathcal{L}}_t}$. Such notations are kept similarly in optimizing $\mathcal{L}$, e.g., $\boldsymbol{W}^{\mathcal{L}_t}$. We further assume that two optimization processes start from the same initial weights $\boldsymbol{A}^{\hat{\mathcal{L}}_0} = \boldsymbol{A}^{\mathcal{L}_0}$, which also means $\boldsymbol{W}^{\hat{\mathcal{L}}_0} = p_0 \boldsymbol{W}^{\mathcal{L}_0}$, where p_t is a scale be-

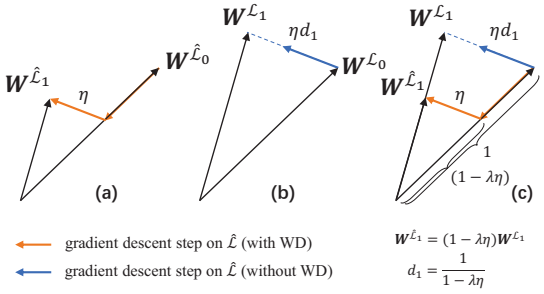

Figure 3: Illustration of proving that weight decay can be entirely replaced by modulating the learning rate (under the setting of $p_0 = 1$ in this example). **(a)** denotes one update with weight decay. **(b)** denotes one update without weight decay. **(c)** shows the similar triangle relationship between these two updates, where the scale ratio $1 - \lambda\eta$ can be easily derived.

tween $\boldsymbol{W}^{\hat{\mathcal{L}}_t}$ and $\boldsymbol{W}^{\mathcal{L}_t}$ (if p_t exists). Specifically, we aim to prove that there exists a sequence of $\{d_1, ..., d_T\}$ as multipliers (note that d must be independent of the training process) for learning rate η during optimizing $\mathcal{L}$, and it ensures $\boldsymbol{A}^{\hat{\mathcal{L}}_t} = \boldsymbol{A}^{\mathcal{L}_t}$ for any step t from $\{1, ..., T\}$. We take the standard SGD (Ruder 2016) for analysis via mathematical induction:

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

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

$$\begin{aligned}
&\boldsymbol{W}^{\hat{\mathcal{L}}_{q+1}} \\
&= (1-\lambda\eta)\boldsymbol{W}^{\hat{\mathcal{L}}_q} - \frac{\eta}{||\boldsymbol{W}^{\hat{\mathcal{L}}_q}||}(\mathbf{1} - \frac{\boldsymbol{W}^{\hat{\mathcal{L}}_q} * \boldsymbol{W}^{\hat{\mathcal{L}}_q}}{||\boldsymbol{W}^{\hat{\mathcal{L}}_q}||^2}) * \frac{\partial\mathcal{L}}{\partial\boldsymbol{A}^{\hat{\mathcal{L}}_q}} \\
&= (1-\lambda\eta)p_q\boldsymbol{W}^{\mathcal{L}_q} - \frac{\eta}{p_q||\boldsymbol{W}^{\mathcal{L}_q}||}(\mathbf{1} - \frac{\boldsymbol{W}^{\mathcal{L}_q} * \boldsymbol{W}^{\mathcal{L}_q}}{||\boldsymbol{W}^{\mathcal{L}_q}||^2}) * \frac{\partial\mathcal{L}}{\partial\boldsymbol{A}^{\mathcal{L}_q}}, \quad (14)
\end{aligned}$$

and

$$\boldsymbol{W}^{\mathcal{L}_{q+1}} = \boldsymbol{W}^{\mathcal{L}_q} - \frac{\eta d_q}{||\boldsymbol{W}^{\mathcal{L}_q}||}(\mathbf{1} - \frac{\boldsymbol{W}^{\mathcal{L}_q} * \boldsymbol{W}^{\mathcal{L}_q}}{||\boldsymbol{W}^{\mathcal{L}_q}||^2}) * \frac{\partial\mathcal{L}}{\partial\boldsymbol{A}^{\mathcal{L}_q}}. \quad (15)$$

Therefore, it is very obvious to deduce from Eq. (14) and (15) that when $d_q = \frac{1}{p_q^2(1-\lambda\eta)}$, we can have

$$\boldsymbol{W}^{\hat{\mathcal{L}}_{q+1}} = (1-\lambda\eta)p_q\boldsymbol{W}^{\mathcal{L}_{q+1}}, \quad (16)$$

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

$$p_{q+1} = (1-\lambda\eta)p_q. \quad (17)$$

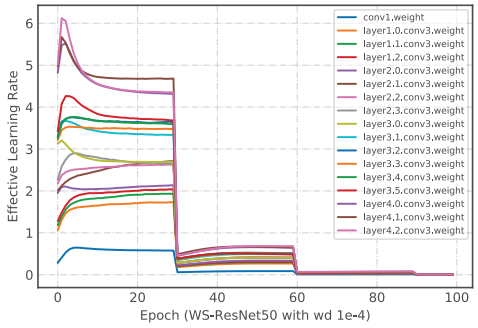

Figure 4: The curves of mean effective learning rate of multiple WS-equipped convolutional layers. We observe an interesting warmup phenomenon in the initial training stage. In the form of "layer$a$.b.conv$c$", "a" denotes the stage number, "b" denotes the bottleneck number, "c" represents the order 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 Figure 2.

Given the sequence of p generated from Eq. (17), the resulted sequence $\{d_1, ..., d_T\}$ of d then becomes $\{\frac{1}{p_1^2(1-\lambda\eta)}, ..., \frac{1}{p_T^2(1-\lambda\eta)}\}$ 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 Figure 3. It is easy to see that $\boldsymbol{W}^{\hat{\mathcal{L}}_1}$ and $\boldsymbol{W}^{\mathcal{L}_1}$ form a similar triangle relationship and their scale factor is only determined by the hyper-parameter λ and η.

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 λ and η.

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 (He et al. 2019; You, Gitman, and Ginsburg 2017; Goyal et al. 2017; Liu et al. 2019). As shown in the Figure 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.

Problems via Introducing Weight Decay

Despite the certain practical success of applying traditional weight decay to control the effective learning rate, there are still several serious problems in essence. In this section, we discuss about these *problems* of introducing weight decay term in the loss objective for weight normalization family.

No Garantee of Global Minimum and Training Instability

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

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

$$\hat{\mathcal{L}}(\boldsymbol{W}^*) = \sum_i L(f(\boldsymbol{x}_i; \frac{\boldsymbol{W}^*}{||\boldsymbol{W}^*||}), y_i) + \frac{1}{2}\lambda||\boldsymbol{W}^*||^2. \quad (18)$$

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

$$\begin{aligned}\hat{\mathcal{L}}(\boldsymbol{W}^\#) &= \sum_i L(f(\boldsymbol{x}_i; \frac{\boldsymbol{W}^\#}{||\boldsymbol{W}^\#||}), y_i) + \frac{1}{2}\lambda||\boldsymbol{W}^\#||^2 \\ &= \sum_i L(f(\boldsymbol{x}_i; \frac{\boldsymbol{W}^*}{||\boldsymbol{W}^*||}), y_i) + \frac{1}{2}\lambda\alpha^2||\boldsymbol{W}^*||^2 \\ &< \sum_i L(f(\boldsymbol{x}_i; \frac{\boldsymbol{W}^*}{||\boldsymbol{W}^*||}), y_i) + \frac{1}{2}\lambda||\boldsymbol{W}^*||^2 = \hat{\mathcal{L}}(\boldsymbol{W}^*),\end{aligned} \quad (19)$$

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

The lack of the global minimum will make the training of the network ill-posed, which means that the model cannot be optimized by any learning parameters. Moreover, the objective will continuously push the length of the weight to 0. The effective learning rate is inversely proportional to the weight length, which is easier to cause the floating point overflow and lead to a failed training.

Specifically, we find that improper selection of λ would actually cause the instability in training. When we choose a slightly larger λ, 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 1 shows the impact of λ on network training with two widely used optimizers SGD (Sutskever et al. 2013) with momentum and Adam (Kingma and Ba 2014), where it is much easier to have a failed training for networks with weights normalized.

Initialization Sensitivity

When the initialized weight $\boldsymbol{W}$ is with a larger magnitude, the training process will spend a lot of iterations and time on minimizing the regularization term, so that it cannot converge efficiently to the optimal solution of the task-related objective. That is, the introduction of the fixed weight decay term makes the networks more sensitive to initialization. To verify this, we plot the accuracy rate curves for the WS-ResNet50 on the ImageNet training and validation set by multiplying the initialization weights by 1, 20, and 100, respectively. From Figure 5 we can see that as the initial length of $\boldsymbol{W}$ increases, the performance of the network gradually degrades, and both the training and validation accuracy drop simultaneously.

Table 1: Accuracy via single 224× crop on ImageNet validation set of different Weight Decay (WD) settings for WS-convolution in WS-ResNet50. WD 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 float overflow. For reference, we list the SGD results of ResNet50 without WS.

λ (WD)	SGD (w/ WS)	Adam (w/ WS)	SGD (w/o WS)
	Top-1/5 Acc (%)		
1e-2	–	–	47.67/72.53
1e-3	–	–	74.12/91.88
1e-4	76.74/93.28	–	76.54/93.07
1e-5	74.70/92.08	–	74.80/92.16

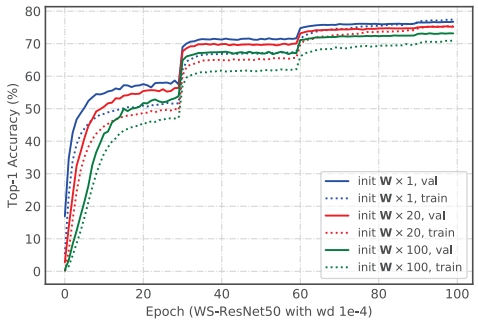

Figure 5: Top-1 Accuracy curves via single 224× crop on the ImageNet training and validation set with different initialization scales for training with weightk decay. The baseline initialization strategy ($\boldsymbol{W}\times1$) is from (He et al. 2015). The traditional fixed weight decay term is sensitive to initialization with various scales.

Method

This section describes our proposed Adaptive Weight Shrink (AWS) method to address the above problems.

Adaptive Weight Shrink

From the analysis of these problems, we understand that a fixed regularization term will bring about training failure and low training efficiency. In order to improve the efficiency and make the statistics of the network stable within a reasonable range, we propose dynamic shrink mechanism during the training process, where the weight is shrunken according to the length of the current weight (or standard deviation for the case of WS), whilst the original fixed weight decay term in the loss objective is discarded. Specifically, for WN, we keep the loss objective as $\mathcal{L}(\boldsymbol{W}) = \sum_i L(f(\boldsymbol{x}_i; \frac{\boldsymbol{W}}{||\boldsymbol{W}||}), y_i)$ and use a dynamic factor:

$$\gamma = \alpha||\boldsymbol{W}||, \quad (20)$$

Table 2: Accuracy via single 224× crop on ImageNet validation set of different α AWS settings for WS-convolution in WS-ResNet50 and WN-convolution in WN-ResNet50, trained with SGD and Adam respectively. WD of other parts (BN and fc layers) is kept with 1e-4. AWS greatly improves the training stability.

α (AWS)	w/ WS		w/ WN	
	SGD	Adam	SGD	Adam
	Top-1 Acc (%)			
1e-1	72.15	71.02	70.64	70.15
1e-2	76.35	71.85	72.02	71.63
1e-3	75.93	72.54	75.76	72.31
1e-4	74.37	72.55	76.63	72.34
1e-5	73.23	72.65	75.48	72.40

to shrink the corresponding weight in each step during the gradient update (e.g., for W_i):

$$\begin{aligned} W_i^{t+1} &= (1-\eta\gamma)W_i^t - \frac{\eta}{||\boldsymbol{W}^t||}(1-\frac{(W_i^t)^2}{\sum_j (W_j^t)^2})\frac{\partial \mathcal{L}}{\partial A_i} \\ &= (1-\eta\alpha||\boldsymbol{W}^t||)W_i^t - \frac{\eta}{||\boldsymbol{W}^t||}(1-\frac{(W_i^t)^2}{\sum_j (W_j^t)^2})\frac{\partial \mathcal{L}}{\partial A_i}. \end{aligned} \tag{21}$$

α is a hyper-parameter for adjusting the intensity of shrinking. For the case of WS, the γ becomes $\alpha\sqrt{\frac{||\boldsymbol{W}-\overline{W}||^2}{n}}$.

Experimental Setting

To validate the effectiveness of the proposed AWS, we conduct comprehensive experiments on the ImageNet (Deng et al. 2009) dataset accordingly. The training settings are kept similar with (Li, Hu, and Yang 2019), except that we set the weight decay ratio λ to 0 for all the bias part in networks (He et al. 2019), which generally improves about 0.2% over the baselines here. We train networks on the training set and report the Top-1 and Top-5 accuracies on the validation set with single 224 × 224 central crop. For data augmentation, we follow the standard practice (Szegedy et al. 2015) and perform the random-size cropping and random horizontal flipping. All networks are trained with naive softmax cross entropy without label-smoothing regularization (Szegedy et al. 2016). We train all the architectures from scratch by SGD (Sutskever et al. 2013) or Adam (Kingma and Ba 2014; Loshchilov and Hutter 2019). 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 (He et al. 2015). Note that in our experiments, only those normalized weights are trained with AWS, others (BN and fc layer weights) are kept with traditional weight decay term since they donot suffer from these problems. We mainly conduct experiments based on the state-of-the-art Weight Standardization (WS) from the weight normalization family. For a fair comparison, all experiments are run under a unified pytorch (Paszke et al. 2017) framework.

Table 3: Accuracy via single 224× crop on ImageNet validation set of different backbones using SGD.

Top-1/5 Acc (%)	baseline	WS + WD	WS + AWS
ResNet-50	76.54/93.07	76.74/93.28	76.59/93.18
ResNet-101	78.17/93.98	78.07/94.02	78.14/94.05
ResNeXt-50	77.64/93.70	77.76/93.76	77.71/93.52
ResNeXt-101	78.71/94.28	78.68/94.17	78.74/94.33
SE-ResNet-50	77.55/93.81	77.78/93.84	77.68/93.77
SE-ResNet-101	78.43/94.15	78.65/94.33	78.50/94.23

Dynamic Balance for Training Stability

Intuitively for WN, AWS will work since the dynamic shrinking ratio γ considers the presence of $||\boldsymbol{W}||$ in the effective learning rate (elr):

$$\gamma \propto ||\boldsymbol{W}||, \quad elr \propto \frac{1}{||\boldsymbol{W}||}. \tag{22}$$

During the optimization, if $||\boldsymbol{W}||$ is too large, γ will increase in proportion, and $||\boldsymbol{W}||$ will decay at a very fast speed; if $||\boldsymbol{W}||$ is too small, γ will be sharply reduced, which means nearly no shrinking will take effect. Meanwhile, the elr greatly increases, and the network will quickly increase the $||\boldsymbol{W}||$ when optimizing the task-related loss. Through the above mechanism, the network can efficiently achieve a stable state of numerical statistics during training, thereby avoiding training failures and inefficient optimizations.

To verify this, analogous to Table 1, we traverse AWS's hyper-parameter α in a large scope, yielding the results in Table 2. In comparison with Table 1, we find that AWS greatly improves the stability of training, i.e., no matter how the hyper-parameter α changes, the optimization can finally converge to a good solution with no cases of training failures for any type of optimizer.

Further, we apply AWS to more network structures (He et al. 2016a; Xie et al. 2017; Hu, Shen, and Sun 2018) and compare it to the original baseline and WS+WD training using SGD. For the WS-equipped networks, we search for the optimal hyper-parameters λ or α to report our results (typically λ = 1e-4 and α = 2.5e-3). As can be seen from Table 3, while keeping excellent training stability, AWS also achieves very competitive results.

Robust to Initialization

Different from the fixed weight decay training, AWS ensures efficient optimization by calculating the dynamic statistics of the training weights. Therefore it is much robust to the initialization. To validate this, similar to Figure 5, we let α = 2.5e-3 and vary the initialization weight by multiplying 1, 20, and 100, respectively. The accuracy curves are depicted in Figure 6, where different training curves quickly converge to similar training and validation errors during the first training phase. For other choices of hyper-parameter α, we can still observe similar results.

Related Work

Weight Normalization Family: Weight Normalization (WN) (Salimans and Kingma 2016) takes the first attempt to

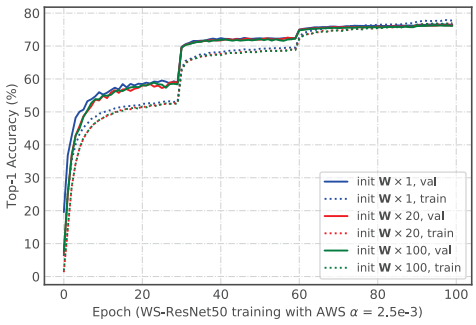

Figure 6: Top-1 Accuracy curves via single 224× crop on the ImageNet training and validation set with different initialization scales for training with AWS. AWS is much more robust to various initialization, compared with fixed weight decay in Figure 5.

reparameterize weights by the separation of direction $\frac{\boldsymbol{W}}{||\boldsymbol{W}||}$ and length g:

$$\boldsymbol{W}' = g\frac{\boldsymbol{W}}{||\boldsymbol{W}||}. \tag{23}$$

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, (Gitman and Ginsburg 2017) points out that in the large-scale ImageNet dataset, the final test accuracy of WN is significantly lower ($\sim 6\%$) than that of BN (Ioffe and Szegedy 2015). 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:

$$\boldsymbol{W}' = g\frac{\boldsymbol{W} - \overline{W}}{||\boldsymbol{W} - \overline{W}||}. \tag{24}$$

Recently, in order to alleviate the problem of degraded performance of GN (Wu and He 2018), Weight Standardization (WS) (Qiao et al. 2019) is proposed, which is very close to CWN but with the learning length g removed:

$$\boldsymbol{W}' = \frac{\boldsymbol{W} - \overline{W}}{\sqrt{\frac{||\boldsymbol{W} - \overline{W}||^2}{n}}}, \tag{25}$$

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. In this paper, we mainly focus on the weight normalization family and conduct a series of analyses on their properties.

Weight Decay: Weight Decay (WD) can be traced back to (Krogh and Hertz 1992), which is defined as multiplying each weight in the gradient descent at each epoch by a factor $\lambda (0 < \lambda < 1)$. It is known to be beneficial for the generalization of neural networks. In the Stochastic Gradient Descent (SGD) setting, WD is widely interpreted as a form of L_2 regularization because it can be derived from the gradient of the L_2 norm of the weights (Loshchilov and Hutter 2019). Recently, (Zhang et al. 2018) 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 (Van Laarhoven 2017; Hoffer et al. 2018) 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 WD 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 λ and η; 3) they fail to discover the problems by introducing WD into the loss objective with normalized weights, which is heavily revealed and discussed in this article; 4) although WD has several potential problems with normalization methods, they have not proposed a solution to replace WD. In contrast, our proposed AWS can guarantee training stability and become robust to initialization.

Conclusion

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 doesnot 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, training instability and initialization sensitivity. Finally, to solve these potential problems, we propose Adaptive Weight Shrink (AWS) that dynamicly shrinks the weight based on their variable magnitude, which significantly improves the training stability and robustness to initialization whilst maintaining competitive performance.

Acknowledgments

The authors would like to thank the editor and the anonymous reviewers for their critical and constructive comments and suggestions. This work was supported by the National Science Fund of China under Grant No. U1713208, Program for Changjiang Scholars, and "111" Program AH92005.

References

Ba, J. L.; Kiros, J. R.; and Hinton, G. E. 2016. Layer normalization. *arXiv preprint arXiv:1607.06450*.

Bottou, L. 2010. Large-scale machine learning with stochastic gradient descent. In *COMPSTAT*.

Deng, J.; Dong, W.; Socher, R.; Li, L.-J.; Li, K.; and Fei-Fei, L. 2009. Imagenet: A large-scale hierarchical image database. In *CVPR*.

Gitman, I., and Ginsburg, B. 2017. Comparison of batch normalization and weight normalization algorithms for the large-scale image classification. *arXiv preprint arXiv:1709.08145*.

Goodfellow, I.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; and Bengio, Y. 2014. Generative adversarial nets. In *NeurIPs*.

Goyal, P.; Dollár, P.; Girshick, R.; Noordhuis, P.; Wesolowski, L.; Kyrola, A.; Tulloch, A.; Jia, Y.; and He, K. 2017. Accurate, large minibatch sgd: Training imagenet in 1 hour. *arXiv preprint arXiv:1706.02677*.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2015. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *ICCV*.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016a. Deep residual learning for image recognition. In *CVPR*.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016b. Identity mappings in deep residual networks. In *ECCV*.

He, T.; Zhang, Z.; Zhang, H.; Zhang, Z.; Xie, J.; and Li, M. 2019. Bag of tricks for image classification with convolutional neural networks. In *CVPR*.

Hoffer, E.; Banner, R.; Golan, I.; and Soudry, D. 2018. Norm matters: efficient and accurate normalization schemes in deep networks. In *NeurIPs*.

Hu, J.; Shen, L.; and Sun, G. 2018. Squeeze-and-excitation networks. In *CVPR*.

Huang, G.; Liu, Z.; Van Der Maaten, L.; and Weinberger, K. Q. 2017a. Densely connected convolutional networks. In *CVPR*.

Huang, L.; Liu, X.; Liu, Y.; Lang, B.; and Tao, D. 2017b. Centered weight normalization in accelerating training of deep neural networks. In *ICCV*.

Ioffe, S., and Szegedy, C. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*.

Kingma, D. P., and Ba, J. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Krogh, A., and Hertz, J. A. 1992. A simple weight decay can improve generalization. In *NeurIPs*.

Li, X.; Hu, X.; and Yang, J. 2019. Spatial group-wise enhance: Improving semantic feature learning in convolutional networks. *arXiv preprint arXiv:1905.09646*.

Lin, T.-Y.; Maire, M.; Belongie, S.; Hays, J.; Perona, P.; Ramanan, D.; Dollár, P.; and Zitnick, C. L. 2014. Microsoft coco: Common objects in context. In *ECCV*.

Liu, L.; Jiang, H.; He, P.; Chen, W.; Liu, X.; Gao, J.; and Han, J. 2019. On the variance of the adaptive learning rate and beyond. *arXiv preprint arXiv:1908.03265*.

Loshchilov, I., and Hutter, F. 2019. Decoupled weight decay regularization.

Luo, P.; Ren, J.; Peng, Z.; Zhang, R.; and Li, J. 2018. Differentiable learning-to-normalize via switchable normalization. *arXiv preprint arXiv:1806.10779*.

Paszke, A.; Gross, S.; Chintala, S.; Chanan, G.; Yang, E.; DeVito, Z.; Lin, Z.; Desmaison, A.; Antiga, L.; and Lerer, A. 2017. Automatic differentiation in pytorch.

Qiao, S.; Wang, H.; Liu, C.; Shen, W.; and Yuille, A. 2019. Weight standardization. *arXiv preprint arXiv:1903.10520*.

Ruder, S. 2016. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*.

Salimans, T., and Kingma, D. P. 2016. Weight normalization: A simple reparameterization to accelerate training of deep neural networks. In *NeurIPs*.

Sutskever, I.; Martens, J.; Dahl, G.; and Hinton, G. 2013. On the importance of initialization and momentum in deep learning. In *ICML*.

Szegedy, C.; Liu, W.; Jia, Y.; Sermanet, P.; Reed, S.; Anguelov, D.; Erhan, D.; Vanhoucke, V.; and Rabinovich, A. 2015. Going deeper with convolutions. In *CVPR*.

Szegedy, C.; Vanhoucke, V.; Ioffe, S.; Shlens, J.; and Wojna, Z. 2016. Rethinking the inception architecture for computer vision. In *CVPR*.

Ulyanov, D.; Vedaldi, A.; and Lempitsky, V. 2016. Instance normalization: The missing ingredient for fast stylization. *arXiv preprint arXiv:1607.08022*.

Van Laarhoven, T. 2017. L2 regularization versus batch and weight normalization. *arXiv preprint arXiv:1706.05350*.

Wang, W.; Li, X.; Yang, J.; and Lu, T. 2018. Mixed link networks. *arXiv preprint arXiv:1802.01808*.

Wu, Y., and He, K. 2018. Group normalization. In *ECCV*.

Xiang, S., and Li, H. 2017. On the effects of batch and weight normalization in generative adversarial networks. *arXiv preprint arXiv:1704.03971*.

Xie, S.; Girshick, R.; Dollár, P.; Tu, Z.; and He, K. 2017. Aggregated residual transformations for deep neural networks. In *CVPR*.

You, Y.; Gitman, I.; and Ginsburg, B. 2017. Large batch training of convolutional networks. *arXiv preprint arXiv:1708.03888*.

Zhang, G.; Wang, C.; Xu, B.; and Grosse, R. 2018. Three mechanisms of weight decay regularization. *arXiv preprint arXiv:1810.12281*.