On the Angular Update and Hyperparameter Tuning of a Scale-Invariant Network

Juseung Yun[1⋆], Janghyeon Lee[2], Hyounguk Shon[1], Eojindl Yi[1], Seung Hwan Kim[2], and Junmo Kim[1]

[1] Korea Advanced Institute of Science and Technology
[2] LG AI Research
{juseung_yun,hyounguk.shon,djwld93,junmo.kim}@kaist.ac.kr
{janghyeon.lee,sh.kim}@lgresearch.ai

Abstract. Modern deep neural networks are equipped with normalization layers such as batch normalization or layer normalization to enhance and stabilize training dynamics. If a network contains such normalization layers, the optimization objective is invariant to the scale of the neural network parameters. The scale-invariance induces the neural network's output to be only affected by the weights' direction and not the weights' scale. We first find a common feature of good hyperparameter combinations on such a scale-invariant network, including learning rate, weight decay, number of data samples, and batch size. Then we observe that hyperparameter setups that lead to good performance show similar degrees of angular update during one epoch. Using a stochastic differential equation, we analyze the angular update and show how each hyperparameter affects it. With this relationship, we can derive a simple hyperparameter tuning method and apply it to the efficient hyperparameter search.

Keywords: scale-invariant network, normalization, effective learning rate, angular update, hyperparameter tuning

1 Introduction

Many recent deep neural network architectures are equipped with normalization layers such as batch normalization (BN) [12], layer normalization (LN) [1], group normalization (GN) [31], or instance normalization (IN) [27]. Such a normalization layer stabilizes deep neural networks' training and boosts generalization performance. These normalization layers make the optimization objective scale-invariant to its parameter, *i.e.*, the weight magnitude does not affect the output of the neural network; only direction does. In stochastic gradient descent (SGD) of a weight $\boldsymbol{x}$ and learning rate η, the direction update of $\boldsymbol{x}$ is proportional to the effective learning rate $\eta/\|\boldsymbol{x}\|^2$ [6,35,28]. Similarly, Wan *et al.* [29] also proposed a concept of angular update, which is the angle between the current and previous weights. Intriguingly, unlike the traditional concept of weight decay (WD), which regularizes the neural network's capacity by preventing the

⋆ Work done during an internship at LG AI Research.

growth of weight norm, scale-invariant networks have the same expressive power regardless of the weight norm. However, weight decay still implicitly regulates the training dynamics by controlling the effective learning rate [6,35,28].

Recent studies have examined the relationship between hyperparameters and SGD dynamics on a scale-invariant network [3,25,24,18,19,29,15,7]. Li *et al.* [18] showed that SGD with fixed learning rate and weight decay is equivalent to the exponentially increasing learning rate schedule without weight decay. However, as the learning rate exponentially increases, the weight norms gradually overflow in calculations, making practical use difficult. Some studies have investigated the relationship between batch size and learning rate [3,25,24]. Goyal *et al.* [3] proposed a simple linear scaling rule, that is, when the minibatch size is multiplied by α, the learning rate is multiplied by α. Several studies [25,24,23,19] adopted stochastic differential equation (SDE), which captures the stochastic gradient noise to derive this linear scaling rule. However, this rule only considers learning rate, momentum, and batch size, disregarding factors such as weight decay and the number of training data samples. Yun *et al.* [33] proposed a weight decay scheduling method according to the number of data samples but without analyzing other hyperparameters.

Contribution. This study investigates the relationship between hyperparameters including learning rate η, momentum μ, weight decay λ, batch size B, and the number of data samples N of SGD, especially on a scale-invariant network with a fixed epoch budget.

Thus far, we have attempted to find a common feature of hyperparameter combinations with good performance in Section 3. Specifically, we tune the learning rate and weight decay to find the optimal combination that yields the best performance. Then, we observe three factors: effective learning rate, effective learning rate per epoch, and angular update per epoch. Although N, B, η, and λ are different, well-tuned hyperparameter combinations show similar angular updates during one epoch (See Figure 2a).

Based on the novel findings, we propose a simple hyperparameter tuning rule for SGD. We aim to find the relationship between the angular update per epoch and hyperparameters. Then, conversely, if we keep the relation between hyperparameters, the neural network would show similar angular updates and also show good performance. Here, we adopt SDE to derive the angular update formula. The result shows that we should keep the tuning factor $\frac{N\eta\lambda}{B(1-\mu)}$ for optimum performance. For instance, when B is multiplied by α, multiply λ by α. Or, when N is multiplied by α, we divide λ by α. We apply the tuning factor for efficient hyperparameter search and show that we can find near optimum hyperparameters even with a small portion of training samples. Although the hyperparameter tuning rule might not be the precise optimal policy and is overly simplified, it is expected to provide valuable insight on how to tune the hyperparameters efficiently.

Scope of this work. Our goal is not to prove why the angular update per epoch is important for tuning but to provide empirical evidence for the interesting observation on the angular update, and simultaneously highlight some insight into our observation. This work only considers training with SGD and a fixed epoch budget. We also assume that SDE can approximate SGD. The theoretical justification for this approximation is provided in [16,20]

2 Preliminaries

SGD and SDE approximation. In SGD, the update of the neural network parameter $\boldsymbol{x}_k \in \mathbb{R}^d$ for a randomly sampled mini-batch $\mathcal{B}_k$ at k-th step is

$$\boldsymbol{x}_{k+1} \leftarrow \boldsymbol{x}_k - \eta \nabla \mathcal{L}(\boldsymbol{x}_k; \mathcal{B}_k), \tag{1}$$

where η is the learning rate and $\mathcal{L}\left(\boldsymbol{x}_k; \mathcal{B}_k\right)$ is the averaged mini-batch loss. Previous studies [2,16,19,17,30,25] used SDE as a surrogate for SGD with a continuous time limit. Assume that the gradient of each sample is independent and follows a short-tailed distribution. Then, the gradient noise, which is the difference between expected gradient $\mathcal{L}\left(\boldsymbol{x}\right) = \mathbb{E}_{\mathcal{B}}\left[\mathcal{L}\left(\boldsymbol{x}; \mathcal{B}\right)\right]$ and mini-batch gradient $\mathcal{L}\left(\boldsymbol{x}; \mathcal{B}\right)$, can be modeled by Gaussian noise with zero mean and covariance $\boldsymbol{\Sigma}(\boldsymbol{x}) := \mathbb{E}_{\mathcal{B}}[(\nabla \mathcal{L}(\boldsymbol{x}; \mathcal{B}) - \nabla \mathcal{L}(\boldsymbol{x}))(\nabla \mathcal{L}(\boldsymbol{x}; \mathcal{B}) - \nabla \mathcal{L}(\boldsymbol{x}))^\top]$. The corresponding SDE for the Equation 1 is

$$d\boldsymbol{X}_t = -\nabla \mathcal{L}(\boldsymbol{X}_t) dt + \left(\eta \boldsymbol{\Sigma}(\boldsymbol{X}_t)\right)^{\frac{1}{2}} d\boldsymbol{W}_t, \tag{2}$$

where the time correspondence is $t = k\eta$, *i.e.*, $\boldsymbol{X}_t \approx \boldsymbol{x}_k$, and $\boldsymbol{W}_t \in \mathbb{R}^d$ is the standard Wiener process on interval $t \in [0, T]$, which satisfies $\boldsymbol{W}_t - \boldsymbol{W}_s \sim \mathcal{N}(\boldsymbol{0}, (t-s)\boldsymbol{I}_d)$. Here, $\mathcal{N}(\boldsymbol{0}, \boldsymbol{I}_d)$ is a normal distribution with zero mean and unit variance. Li *et al.* [17] rigorously proved this continuous time limit approximation holds for infinitesimal learning rate. Li *et al.* [20] showed that the approximation is also valid for finite learning rate and theoretically analyzed the conditions for the approximation to hold. Smith *et al.* [25,23] used the SDE to derive the linear scaling rule [3].

From now on, with an abuse of notation, we will omit the mini-batch $\mathcal{B}$ in the loss function $\mathcal{L}(\boldsymbol{x}; \mathcal{B})$ for brevity, if the mini-batch dependency is clear from the context. The stochastic gradient descent with momentum (SGDM) can be written as the following updates

$$\begin{aligned} \boldsymbol{v}_{k+1} &= \mu \boldsymbol{v}_k - \eta \nabla \mathcal{L}(\boldsymbol{x}_k) \\ \boldsymbol{x}_{k+1} &= \boldsymbol{x}_k + \boldsymbol{v}_{k+1}, \end{aligned} \tag{3}$$

where μ is the momentum parameter, and $\boldsymbol{v}$ is the velocity. Similar to the SGD case, the following SDE is an *order-1 weak approximation* of SGDM [16,17]:

$$\begin{aligned} d\boldsymbol{V}_t &= \left(-\eta^{-1}(1-\mu)\boldsymbol{V}_t - \nabla \mathcal{L}\left(\boldsymbol{X}_t\right)\right) dt + \left(\eta \boldsymbol{\Sigma}\left(\boldsymbol{X}_t\right)\right)^{\frac{1}{2}} d\boldsymbol{W}_t \\ d\boldsymbol{X}_t &= \eta^{-1} \boldsymbol{V}_t dt. \end{aligned} \tag{4}$$

Scale-invariance. Let $\mathcal{L}(\boldsymbol{x}) : \mathbb{R}^d \to \mathbb{R}$ be the loss function of a neural network parameterized by $\boldsymbol{x}$. $\mathcal{L}(\boldsymbol{x})$ is said to be a *scale-invariant function* if it satisfies $\mathcal{L}(\alpha\boldsymbol{x}) = \mathcal{L}(\boldsymbol{x})$ for any $\boldsymbol{x} \in \mathbb{R}^d$ and any $\alpha > 0$, and the neural network is said to be a *scale-invariant neural network*. For a scale-invariant function $\mathcal{L}(\boldsymbol{x})$, the following properties hold:

$$1. \quad \nabla\mathcal{L}(\alpha\boldsymbol{x}) = \frac{1}{\alpha}\nabla\mathcal{L}(\boldsymbol{x}) \tag{5}$$

$$2. \quad \langle \boldsymbol{x}, \nabla\mathcal{L}(\boldsymbol{x}) \rangle = 0 \tag{6}$$

$$3. \quad \|\boldsymbol{\Sigma}(\boldsymbol{x})^{1/2}\boldsymbol{x}\| = \sqrt{\boldsymbol{x}^\top \boldsymbol{\Sigma}(\boldsymbol{x})\boldsymbol{x}} = 0, \tag{7}$$

where $\|\cdot\|$ denots the L_2 norm of a vector. These properties are previously discussed [6,35,28,18,19], but we also derive them in the Appendix.

Effective learning rate. For a scale-invariant network parameterized by weight $\boldsymbol{x}$, the magnitude of $\boldsymbol{x}$ does not affect the output, and only the direction does. Defining $\boldsymbol{y} := \frac{\boldsymbol{x}}{\|\boldsymbol{x}\|}$, then SGD update of unit vector $\boldsymbol{y}$ can be approximated as follows:

$$\boldsymbol{y}_{k+1} \approx \boldsymbol{y}_k - \frac{\eta}{\|\boldsymbol{x}_k\|^2}\nabla\mathcal{L}\left(\boldsymbol{y}_k\right). \tag{8}$$

The result shows that the update of $\boldsymbol{y}$ is proportional to $\eta/\|\boldsymbol{x}\|^2$, or the *effective learning rate* [28,6,35].

3 Common Feature of Good Hyperparameter Combinations

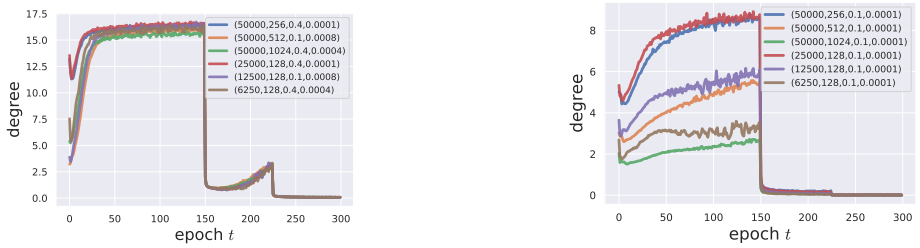
This section finds a common property of hyperparameter combinations that performs well, and demonstrates novel observations on an update of weight direction. These observations serve as a key intuition for our simple hyperparameter tuning rule. We first describe the architectural modification to make a convolutional neural network scale-invariant, and define novel indices – *effective learning rate per epoch* and *angular update per epoch* – in Section 3.1. Then, we provide experimental setups and observation results in Section 3.2.

3.1 Measuring Updates of a Scale-Invariant Network

Network modification. We first describe how to convert a neural network scale-invariant. Many convolutional neural networks (CNNs) are equipped with normalization layers and consist of several *convolution→normalization→activation function* sequences. Before output there is a global average pooling and a fully connected layer [5,34,11,4,32,9]. Such a network is invariant to all convolutional parameters followed by a normalization layer. However, a normalization layer (*e.g.*, BN) also has an affine parameter and the last fully connected layer has

(a) (b)

Fig. 1: (a) Effective learning rate and (b) effective learning rate per epoch of well-tuned hyperparameter combinations.



(a) (b)

Fig. 2: (a) Angular update per epoch of well-tuned hyperparameter combinations and (b) poorly tuned hyperparameter combinations.

no followed normalization layer. To increase the reliability of the observation in scale-invariant networks, we modify the ResNet [5] to make the training objective scale-invariant to all trainable parameters. We add normalization without affine transform after global average pooling and fix the parameters of the FC layer as done in [8,18,19]. We use ghost batch normalization (GBN) [7] with a ghost batch size of 64 as the normalization layers. GBN is often used in large batch training [24,23] to ensure the mean gradient is independent of batch size [25]. We name this modified architecture as *scale-invariant* ResNet. However, we also observe the result on unmodified ResNet in Section 5.2.

Measure for directional updates. As described in Section 2, the *effective learning rate* is defined as

$$\eta_{eff} := \eta / \|\boldsymbol{x}\|^2. \tag{9}$$

We define a novel index, *effective learning rate per epoch* η_{epoch} as

$$\eta_{epoch} := \eta_{eff} \frac{N}{B} = \frac{\eta N}{\|\boldsymbol{x}\|^2 B}. \tag{10}$$

The number of iterations during one epoch is N/B, and thus η_{epoch} means η_{eff} multiplied by the number of updates per epoch. We also propose another new index, *angular update per epoch* $\Delta\theta$ for measuring directional change of a vector. At e-th epoch, $\Delta\theta$ is defined as

$$\Delta\theta := \measuredangle\left(\boldsymbol{x}_{eN/B}, \boldsymbol{x}_{(e-1)N/B}\right) = \arccos\left(\frac{\left\langle \boldsymbol{x}_{eN/B}, \boldsymbol{x}_{(e-1)N/B}\right\rangle}{\|\boldsymbol{x}_{eN/B}\| \cdot \|\boldsymbol{x}_{(e-1)N/B}\|}\right). \tag{11}$$

3.2 Observation

Experimental setup. This experiment trains a scale-invariant ResNet-18 [5] on the CIFAR-10 dataset [13]. We train for 300 epochs regardless of the number of training samples using SGD with a momentum coefficient of 0.9. For all setups, the learning rate is divided by 10 at epochs 150 and 225. We use standard augmentation settings such as resizing, cropping, and flipping. We tune the learning rate and weight decay for six different hyperparameter combinations (N, B, η, λ):

1. $(50000, 256, \eta, 0.0001)$
2. $(50000, 256, 0.1, \lambda)$
3. $(50000, 1024, \eta, \lambda)$
4. $(25000, 128, \eta, 0.0001)$
5. $(12500, 128, 0.1, \lambda)$
6. $(6250, 128, \eta, \lambda)$.

Values written in Arabic numerals are fixed, and hyperparameters written in symbols are searched (although N is typically not regarded as a hyperparameter, for convenience, we include it in the combination notation). When tuning only η, we search $\eta \in \{0.1, 0.2, 0.4, 0.8, 1.6, 3.2\}$. When tuning only λ, we search $\lambda \in \{0.0001, 0.0002, 0.0004, 0.00008, 0.0016, 0.0032\}$. When tuning both η and λ, we search from $(\eta, \lambda) = (0.1, 0.0001)$ to $(0.8, 0.0008)$ while multiplying both with a factor of $\sqrt{2}$. We evaluate the average performance of three runs for tuning.

Result. The searched values exhibiting the optimal test accuracy are shown in legends of Figure 1 and Figure 2a. Figure 1 represents the effective learning rate η_{eff} of the found hyperparameter combinations. In all cases, η_{eff} initially increases and converges to a certain value until the first learning rate decays. It implies $\|\boldsymbol{x}\|$ also converges to a certain value. However, we cannot find a common feature of the converging value of η_{eff}. Figure 1b represents the effective learning rate per epoch η_{epoch} of the found hyperparameter combinations. For cases with the same number of data, η_{epoch} showed similar values as training proceeded (blue, orange, and green lines), although they differed at initial transient phases. If the number of iterations is α times higher, a well-tuned hyperparameter combination has α times smaller η_{eff}. However, for the cases with a different number of data (red, purple, and brown lines), even η_{epoch} showed different values. Figure 2a shows the angular update per epoch $\Delta\theta$ of well-tuned hyperparameter combinations and Figure 2b shows the result of the poorly tuned combinations.

Although N, B, η, and λ were different, hyperparameter combinations resulting in good performance showed similar angular updates during one epoch. In contrast, poorly tuned hyperparameter combinations showed different values of η_{epoch}.

4 Angular Update

This section adopts SDE to describe the angular update according to the hyperparameters. Then we propose a simple hyperparameter tuning rule.

Angular update of SGD. Similar to Equation 2, we obtain the following SDE by considering weight decay:

$$d\boldsymbol{X}_t = -\nabla\mathcal{L}(\boldsymbol{X}_t)dt - \lambda\boldsymbol{X}_t dt + (\eta\boldsymbol{\Sigma}(\boldsymbol{X}_t))^{\frac{1}{2}}\, d\boldsymbol{W}_t, \tag{12}$$

where the time correspondence is $t = k\eta$, *i.e.*, $\boldsymbol{X}_t \approx \boldsymbol{x}_k$. In such a SDE, however, the time scale does not include information on how many iterations the neural network is trained. Assuming that the network is trained for the same number of epochs, the number of entire training iterations is proportional to the ratio of the number of data to batch size N/B. To make the time correspondence inversely proportional to N/B, we rescale the time by $\frac{B}{N}$. We rescale the above SDE by considering $\tilde{\boldsymbol{X}}_t = \boldsymbol{X}_{\frac{N}{B}t}$; then, we have

$$\begin{aligned} d\tilde{\boldsymbol{X}}_t &= d\boldsymbol{X}_{\frac{N}{B}t} && (13)\\ &= -\nabla\mathcal{L}(\boldsymbol{X}_{\frac{N}{B}t})d\left(\frac{N}{B}t\right) - \lambda\boldsymbol{X}_{\frac{N}{B}t}d\left(\frac{N}{B}t\right) + \left(\eta\boldsymbol{\Sigma}(\boldsymbol{X}_{\frac{N}{B}t})\right)^{\frac{1}{2}} d\boldsymbol{W}_{\frac{N}{B}t} && (14)\\ &= -\frac{N}{B}\nabla\mathcal{L}(\tilde{\boldsymbol{X}}_t)dt - \frac{N\lambda}{B}\tilde{\boldsymbol{X}}_t dt + \left(\eta\boldsymbol{\Sigma}(\tilde{\boldsymbol{X}}_t)\right)^{\frac{1}{2}} d\boldsymbol{W}_{\frac{N}{B}t}, && (15) \end{aligned}$$

where the time correspondence is $t = \frac{B}{N}k\eta$, *i.e.*, evolving for η time with the above SDE approximates $\Omega(N/B)$ steps of SGD. Similar to deriving the linear scaling rule [25,24,23,3], we also assume that during a small time interval η, the parameters do not move far enough for the gradients to significantly change, *i.e.*, $\|\tilde{\boldsymbol{X}}_t\| \approx \|\tilde{\boldsymbol{X}}_{t+\eta}\|$ and $\nabla\mathcal{L}(\tilde{\boldsymbol{X}})$ are nearly constant. Then, we obtain the update as follows:

$$\tilde{\boldsymbol{X}}_{t+\eta} \approx \tilde{\boldsymbol{X}}_t - \frac{N\eta}{B}\nabla\mathcal{L}(\tilde{\boldsymbol{X}}_t) - \frac{N\eta\lambda}{B}\tilde{\boldsymbol{X}}_t + \left(\eta\boldsymbol{\Sigma}(\tilde{\boldsymbol{X}}_t)\right)^{\frac{1}{2}} \boldsymbol{W}_{\frac{N}{B}\eta}. \tag{16}$$

Define $\boldsymbol{Y} = \frac{\tilde{\boldsymbol{X}}}{\|\tilde{\boldsymbol{X}}\|}$, then we get

$$\boldsymbol{Y}_{t+\eta} \approx \left(1 - \frac{N\eta\lambda}{B}\right)\boldsymbol{Y}_t - \frac{N\eta}{B\|\tilde{\boldsymbol{X}}\|^2}\nabla\mathcal{L}(\boldsymbol{Y}_t) + \frac{1}{\|\tilde{\boldsymbol{X}}\|}\left(\eta\boldsymbol{\Sigma}(\boldsymbol{Y}_t)\right)^{\frac{1}{2}} \boldsymbol{W}_{\frac{N}{B}\eta}. \tag{17}$$

The angle between $\tilde{\boldsymbol{X}}_t$ and $\tilde{\boldsymbol{X}}_{t+\eta}$ is $\arccos\langle \boldsymbol{Y}_t, \boldsymbol{Y}_{t+\eta}\rangle$. By Equation 6 and Equation 7, $\langle \boldsymbol{Y}_t, \nabla\mathcal{L}(\boldsymbol{Y}_t)\rangle = 0, \boldsymbol{Y}_t^\top \boldsymbol{\Sigma}(\boldsymbol{Y}_t)^{\frac{1}{2}} = 0$. Then, we have

$$\langle \boldsymbol{Y}_t, \boldsymbol{Y}_{t+\eta}\rangle \approx 1 - \frac{N\eta\lambda}{B}. \tag{18}$$

The result suggests that for $\Theta(N/B)$ steps, the angular update is proportional to $1 - \frac{N\eta\lambda}{B}$. Although we assumed that the parameters do not move far during the update, we observed that such a relationship still holds for one epoch interval (Section 5). For instance, when the number of data is multiplied by α, multiplying η by $1/\alpha$ can maintain the angular update per epoch. The result also coincides with that of the linear scaling rule [3,24].

Angular update of SGDM. Similar to Equation 3, by considering weight decay, we obtain the following combined SDE for SGDM

$$d\boldsymbol{V}_t = \left(-\eta^{-1}(1-\mu)\boldsymbol{V}_t - \nabla\mathcal{L}(\boldsymbol{X}_t) - \lambda\boldsymbol{X}_t\right)dt + \left(\eta\boldsymbol{\Sigma}(\boldsymbol{X}_t)\right)^{\frac{1}{2}} d\boldsymbol{W}_t, \tag{19}$$

$$d\boldsymbol{X}_t = \eta^{-1}\boldsymbol{V}_t dt. \tag{20}$$

To make time correspondence inversely proportional to N/B, we rescale the above SDE by considering $\tilde{\boldsymbol{X}}_t = \boldsymbol{X}_{\frac{N}{B}t}$ and $\tilde{\boldsymbol{V}}_t = \boldsymbol{V}_{\frac{N}{B}t}$, which yields

$$d\tilde{\boldsymbol{V}}_t = -\frac{N}{B}\left(\frac{1-\mu}{\eta}\tilde{\boldsymbol{V}}_t + \nabla\mathcal{L}(\tilde{\boldsymbol{X}}_t) + \lambda\tilde{\boldsymbol{X}}_t\right)dt + \left(\eta\boldsymbol{\Sigma}(\tilde{\boldsymbol{X}}_t)\right)^{\frac{1}{2}} d\boldsymbol{W}_{\frac{N}{B}t}, \tag{21}$$

$$d\tilde{\boldsymbol{X}}_t = \frac{N}{B\eta}\tilde{\boldsymbol{V}}_t dt. \tag{22}$$

Using Equation 21, we can rewrite the right hand side of Equation 22 as

$$\begin{aligned}\frac{N}{B\eta}\tilde{\boldsymbol{V}}_t dt = -\frac{1}{1-\mu}d\tilde{\boldsymbol{V}}_t - \frac{N}{B(1-\mu)}\left(\nabla\mathcal{L}(\tilde{\boldsymbol{X}}_t) + \lambda\tilde{\boldsymbol{X}}_t\right)dt \\ + \frac{1}{1-\mu}\left(\eta\boldsymbol{\Sigma}(\tilde{\boldsymbol{X}}_t)\right)^{\frac{1}{2}} d\boldsymbol{W}_{\frac{N}{B}t}.\end{aligned} \tag{23}$$

We also assume that during a short time interval η, the parameters do not move far enough for the gradients and velocity to change significantly. Then, by integrating Equation 22 from t to $t+\eta$, we get

$$\begin{aligned}\tilde{\boldsymbol{X}}_{t+\eta} \approx \tilde{\boldsymbol{X}}_t - \frac{\tilde{\boldsymbol{V}}_{t+\eta} - \tilde{\boldsymbol{V}}_t}{1-\mu} - \frac{N\eta}{B(1-\mu)}\left(\nabla\mathcal{L}(\tilde{\boldsymbol{X}}_t) + \lambda\tilde{\boldsymbol{X}}_t\right) \\ + \frac{1}{1-\mu}\left(\eta\boldsymbol{\Sigma}(\tilde{\boldsymbol{X}}_t)\right)^{\frac{1}{2}} \boldsymbol{W}_{\frac{N}{B}\eta}.\end{aligned} \tag{24}$$

We define $\boldsymbol{Y} = \frac{\tilde{\boldsymbol{X}}}{\|\tilde{\boldsymbol{X}}\|}$; futher, the direction update is

$$\begin{aligned}\boldsymbol{Y}_{t+\eta} \approx \boldsymbol{Y}_t - \frac{\tilde{\boldsymbol{V}}_{t+\eta} - \tilde{\boldsymbol{V}}_t}{(1-\mu)\|\tilde{\boldsymbol{X}}_t\|} - \frac{N\eta}{B(1-\mu)\|\tilde{\boldsymbol{X}}_t\|}\nabla\mathcal{L}(\boldsymbol{Y}_t) - \frac{N\eta\lambda}{B(1-\mu)}\boldsymbol{Y}_t \\ - \frac{1}{(1-\mu)\|\tilde{\boldsymbol{X}}_t\|}\left(\eta\boldsymbol{\Sigma}(\boldsymbol{Y}_t)\right)^{\frac{1}{2}} \boldsymbol{W}_{\frac{N}{B}\eta}.\end{aligned} \tag{25}$$

Thus, the angle between $\boldsymbol{Y}_t$ and $\boldsymbol{Y}_{t+\eta}$ is $\arccos\langle \boldsymbol{Y}_t, \boldsymbol{Y}_{t+\eta}\rangle$ and we get

$$\langle \boldsymbol{Y}_t, \boldsymbol{Y}_{t+\eta}\rangle \approx 1 - \frac{N\eta\lambda}{B(1-\mu)} - \frac{\left\langle \boldsymbol{Y}_t, \tilde{\boldsymbol{V}}_{t+\eta} - \tilde{\boldsymbol{V}}_t \right\rangle}{(1-\mu)\|\tilde{\boldsymbol{X}}_t\|} \tag{26}$$

$$\approx 1 - \frac{N\eta\lambda}{B(1-\mu)}. \tag{27}$$

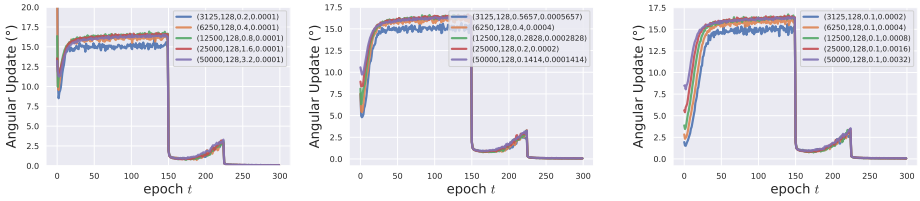
We refer to $\frac{N\eta\lambda}{B(1-\mu)}$ as the *tuning factor*, which determines how much the angular update occurs during a single epoch of training. Thus, based on the observations in Section 3 and Equation 27, we argue that rather than searching for a hyperparameter individually, one should search for the tuning factor $\frac{N\eta\lambda}{B(1-\mu)}$ for an efficient tuning. Finding the tuning factor is sufficient for obtaining near optimum performance, *i.e.*, even if any hyperparameter is changed, a good performance can still be obtained by maintaining the tuning factor. With an intuition for the range of appropriate learning rate and momentum coefficient (*e.g.*, 0.1 for initial learning rate and 0.9 for momentum coefficient is used in many architectures) and by choosing a moderate batch size considering GPU memory or computation budget, we only need to search for a weight decay.

5 Experiments

In this section, we demonstrate that hyperparameter combinations with good performance show similar angular updates per epoch. With the same tuning factor, we can make these angular updates per epoch similar. We first show the experimental results on a modified scale-invariant architecture in Section 5.1. Next, we show the results on an unmodified architecture that is equipped with BN but has scale-variant parameters in Section 5.2.

5.1 Scale-Invariant Network

Experimental setup. For clarity, in the main text, we only report experiments using ResNet-18 [5] (or scale-invariant ResNet-18) on CIFAR-10 [13]; however, we provide additional experiments using DenseNet-100 [11] on CIFAR-100 [13], and ResNet-18 on Tiny ImageNet [14] in the appendix. We train for 300 epochs regardless of the number of training samples, and the learning rate is divided by 10 at epochs 150 and 225. For SGDM, we set the base values as LR=0.1 and WD=0.0001, search them by multiplying the factor of 2 or $\sqrt{2}$, and set the momentum coefficient as 0.9. For SGD, we set the base value as LR=1, which makes the value $\frac{N\eta\lambda}{B(1-\mu)}$ the same as LR=0.1 for the SGDM setting. We use the standard augmentation setting such as resizing, cropping, and flipping. Because the tuning factor comprises five components $(N, B, \eta, \lambda, \mu)$, there are many possible combinations. To validate the tuning factor, we categorize the combinations into three: fixed B, fixed N, and fixed $\eta\lambda$. We report the average performance of three runs. However, if there is divergence among the three runs, we exclude the run when calculating the average; if all runs diverge, we leave the result table blank.



(a) (b) (c)

Fig. 3: Angular update per epoch of scale-invariant network with the batch size 128: (a) tuning LR, (b) tuning WD, and (c) tuning both LR and WD.

Table 1: Scale-invariant ResNet-18 on CIFAR-10 with SGDM and B=128.

LR	WD	3125	6250	12500	25000	50000
6.4	0.0001	76.99	81.71	83.07	81.77	80.28
3.2	0.0001	**78.28**	83.70	86.93	89.25	89.92
1.6	0.0001	77.87	84.36	88.75	91.60	93.38
0.8	0.0001	76.73	**84.69**	88.93	92.75	94.79
0.4	0.0001	74.78	83.90	**89.06**	**92.81**	**95.13**
0.2	0.0001	73.17	83.34	88.70	92.76	95.08
0.1	0.0001	71.62	81.61	87.79	92.07	94.86

(a) Tuning LR

LR	WD	3125	6250	12500	25000	50000
0.1	0.0064	77.31	81.71	82.94	83.79	80.18
0.1	0.0032	**78.59**	83.83	87.44	89.43	90.55
0.1	0.0016	77.57	**85.22**	89.06	91.94	93.39
0.1	0.0008	76.68	85.11	**89.47**	**93.10**	94.89
0.1	0.0004	74.14	84.06	89.31	92.90	**95.13**
0.1	0.0002	73.14	82.66	88.56	92.74	95.00
0.1	0.0001	71.62	81.61	87.79	92.07	94.86

(b) Tuning WD

LR	WD	3125	6250	12500	25000	50000
0.8	0.0008	76.93	81.78	83.19	82.95	79.39
0.5657	0.0005657	**78.48**	84.00	87.18	89.21	90.63
0.4	0.0004	78.39	**84.71**	88.92	92.07	93.16
0.2828	0.0002828	76.67	84.80	**89.63**	92.86	94.95
0.2	0.0002	74.64	83.85	89.12	**92.96**	**95.26**
0.1414	0.0001414	73.66	82.93	88.62	92.44	95.09
0.1	0.0001	71.62	81.61	87.79	92.07	94.86

(c) Tuning LR,WD

Table 2: Scale-invariant ResNet-18 on CIFAR-10 with SGD and B=128.

LR	WD	3125	6250	12500	25000	50000
64	0.0001	76.75	80.58	79.50	78.98	
32	0.0001	**78.85**	84.36	88.07	90.27	91.43
16	0.0001	78.24	84.34	89.10	92.15	93.58
8	0.0001	77.08	**84.72**	**89.19**	92.53	94.80
4	0.0001	75.90	83.96	89.05	**92.79**	**95.23**
2	0.0001	74.90	83.46	88.49	92.65	95.14
1	0.0001	73.24	82.18	88.12	92.26	94.83

(a) Tuning LR

LR	WD	3125	6250	12500	25000	50000
1	0.0064	77.28	81.53	78.34	78.76	78.95
1	0.0032	**79.42**	84.67	88.40	91.11	91.60
1	0.0016	79.06	**85.14**	89.63	92.52	94.02
1	0.0008	77.71	84.94	**89.67**	93.15	95.02
1	0.0004	75.86	84.22	89.34	**92.96**	**95.10**
1	0.0002	74.40	83.53	88.67	92.53	95.05
1	0.0001	73.24	82.18	88.12	92.26	94.83

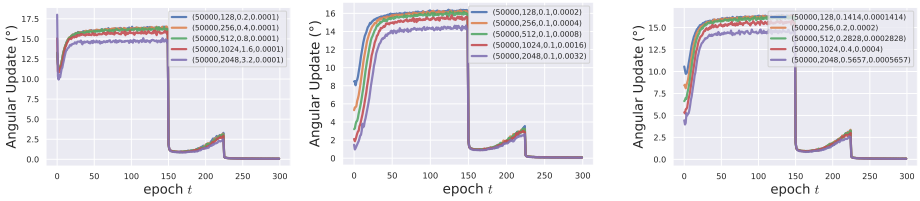
(b) Tuning WD

LR	WD	3125	6250	12500	25000	50000
8	0.0008	77.48	81.22	79.81	79.40	78.84
5.657	0.0005657	**79.62**	84.30	88.03	90.50	91.72
4	0.0004	78.75	84.68	89.19	92.32	94.03
2.828	0.0002828	77.19	**85.08**	**89.25**	92.70	94.96
2	0.0002	75.97	84.28	89.18	**92.98**	**95.24**
1.414	0.0001414	74.17	83.17	88.87	92.66	95.12
1	0.0001	73.24	82.18	88.12	92.26	94.83

(c) Tuning LR,WD

Experiment on fixed B. In these experiments, we fixed B as 128 and tuned LR or (and) WD for each number of data samples. Table 1 and Table 2 show the results on SGDM and SGD, respectively. Yellow-colored cells satisfy the tuning factor $\frac{N\eta\lambda}{B(1-\mu)} = \frac{10}{128}$, and bold values represent the highest accuracy for each column. For all columns, the best performance or the second best accuracy was in the yellow cell. Figure 3 shows the angular update per epoch of the yellow cells of Table 1. They show distinct aspects in an initial transient phase where the weight changes rapidly. However, after 50 epochs, they show very similar angular updates, indicating the tuning factor's validity.

Experiment on fixed N. In these experiments, we fixed N at 50k and tuned LR or (and) WD for each B. Table 3 and Table 4 show the results on SGDM and SGD, respectively. Yellow color indicates the cells that satisfy the tuning factor $\frac{N\eta\lambda}{B(1-\mu)} = \frac{10}{128}$, and bold values represent the highest accuracy for each column. For all columns of Table 3, the best performance or the second-best accuracy



(a) (b) (c)

Fig. 4: Angular update per epoch of scale-invariant network with the number of data being 50k. (a) Tuning the learning rate, (b) tuning the weight decay, and (c) tuning both the learning rate and the weight decay.

Table 3: Scale-invariant ResNet-18 on CIFAR-10 with SGDM and 50k data samples.

LR	WD	2048	1024	512	256	128
6.4	0.0001	94.52	93.89	91.90	88.55	80.28
3.2	0.0001	**94.87**	**94.92**	94.31	92.80	89.92
1.6	0.0001	94.74	94.91	95.03	94.66	93.38
0.8	0.0001	94.21	94.83	**95.04**	94.99	94.79
0.4	0.0001	93.71	94.44	94.87	**95.13**	**95.13**
0.2	0.0001	92.96	93.93	94.45	94.85	95.08
0.1	0.0001	92.36	93.31	93.85	94.50	94.86

(a) Tuning LR

LR	WD	2048	1024	512	256	128
0.1	0.0064	**94.87**	94.23	92.41	89.27	80.18
0.1	0.0032	94.82	95.01	94.37	93.17	90.55
0.1	0.0016	94.74	**95.05**	95.02	94.78	93.39
0.1	0.0008	94.16	94.71	**95.16**	**95.30**	94.89
0.1	0.0004	93.46	94.17	94.76	95.19	**95.13**
0.1	0.0002	92.69	93.75	94.52	94.95	95.00
0.1	0.0001	92.36	93.31	93.85	94.50	94.86

(b) Tuning WD

LR	WD	2048	1024	512	256	128
0.8	0.0008	94.69	94.22	92.44	89.43	79.39
0.5657	0.0005657	**94.98**	95.02	94.39	93.12	90.63
0.4	0.0004	94.76	**95.05**	**95.13**	94.79	93.16
0.2828	0.0002828	94.19	94.70	95.01	**95.24**	94.95
0.2	0.0002	93.75	94.49	94.76	95.11	**95.26**
0.1414	0.0001414	93.09	93.70	94.40	94.91	95.09
0.1	0.0001	92.36	93.31	93.85	94.50	94.86

(c) Tuning LR,WD

Table 4: Scale-invariant ResNet-18 on CIFAR-10 with SGD and 50k data samples.

LR	WD	2048	1024	512	256	128
64	0.0001	86.40	83.72	82.38	82.70	
32	0.0001	91.25	93.62	94.31	93.17	91.43
16	0.0001	93.13	94.63	95.03	94.70	93.58
8	0.0001	**93.50**	**94.69**	**95.15**	**95.06**	94.80
4	0.0001	93.18	94.54	95.04	95.01	**95.23**
2	0.0001	93.06	93.76	94.33	94.76	95.14
1	0.0001	92.30	93.20	93.97	94.57	94.83

(a) Tuning LR

LR	WD	2048	1024	512	256	128
1	0.0064	86.48	82.41	82.81	81.44	78.95
1	0.0032	92.63	93.66	94.46	93.95	91.60
1	0.0016	93.66	94.60	**95.17**	94.79	94.02
1	0.0008	**93.79**	**94.66**	95.00	95.22	95.02
1	0.0004	93.47	94.36	94.90	**95.22**	**95.10**
1	0.0002	92.63	93.62	94.31	94.85	95.05
1	0.0001	92.30	93.20	93.97	94.57	94.83

(b) Tuning WD

LR	WD	2048	1024	512	256	128
8	0.0008	86.08	84.91	83.29	81.19	78.84
5.657	0.00056569	92.95	93.94	94.68	93.31	91.72
4	0.0004	93.28	**94.75**	**95.16**	94.92	94.03
2.828	0.00028284	**93.72**	94.68	95.10	**95.21**	94.96
2	0.0002	93.28	94.32	94.96	95.17	**95.24**
1.414	0.00014142	92.77	93.69	94.56	94.91	95.12
1	0.0001	92.30	93.20	93.97	94.57	94.83

(c) Tuning LR,WD

was in the yellow cell. An interesting point here is not only the LR linear scaling rule but also a WD linear scaling rule is possible as batch size changes. For SGD (Table 4), there exists a case in which even the second-best value is not in yellows cells for B bigger than 512. This may because the excessively large LR has caused unstable training. Figure 4 shows the angular update per epoch of the yellow cells of Table 3. At the initial transient phase, where the weight changes rapidly, they showed different aspects. However, after 50 epoch, they showed very similar angular update, and it shows the validity of tuning factor.

Experiment on fixed LR×WD. In these experiments, we fixed the LR as 0.1 and WD as 0.0016, and then tuned B for each number of data samples. Table 5 shows the results on SGDM. We also used yellow to indicate cells that satisfy the

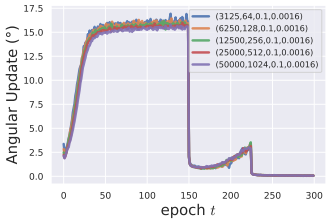
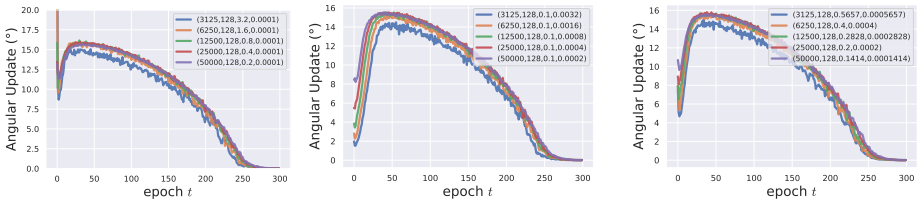


Fig. 5: Angular update per epoch of scale-invariant network with η=0.1 and λ=0.0016.

B	LR	WD	3125	6250	12500	25000	50000
32	0.1	0.0016	78.64	83.10	85.30	87.08	87.44
64	0.1	0.0016	**79.25**	84.30	87.83	89.52	91.01
128	0.1	0.0016	77.57	**85.22**	89.06	91.94	93.39
256	0.1	0.0016	75.65	84.60	**89.31**	**92.84**	94.78
512	0.1	0.0016	72.02	82.90	89.10	92.78	95.02
1024	0.1	0.0016	68.19	79.62	87.90	92.58	**95.05**
2048	0.1	0.0016	51.51	74.71	85.90	91.41	94.74

Table 5: Scale-invariant ResNet-18 on CIFAR-10 with SGDM, η=0.1, and λ=0.0016.



(a) (b) (c)

Fig. 6: Angular update per epoch of scale-invariant network with the number of data = 50k and using cosine LR scheduling. (a) Tuning LR, (b) tuning WD, and (c) tuning both LR rate and WD.

tuning factor $\frac{N\eta\lambda}{B(1-\mu)} = \frac{10}{128}$; the bold values represent the highest accuracy for each column. For all columns of Table 5, the best performance or the second-best accuracy was in the yellow cell. Figure 5 shows the angular update per epoch of the yellow cells of Table 5. They showed remarkably similar angular updates, indicating the validity of the tuning factor.

Experiment on cosine LR scheduling. In these experiments, we fixed N as 50k and tuned LR or (and) WD for each B, but used cosine LR scheduling. Table 6 shows the results on SGDM. We also used yellow color to indidate cells that satisfy the tuning factor $\frac{N\eta\lambda}{B(1-\mu)} = \frac{10}{128}$; bold values represent the highest accuracy for each column. Figure 6 shows the angular update per epoch of the yellow cells of Table 6. The result shows that the tuning factor remains valid for cosine LR scheduling.

5.2 Unmodified Network

Previously, we demonstrated the validity of the tuning factor on a scale-invariant network. We now demonstrate that the tuning factor is still valid on a ResNet-18 which is not modified, as described in Section 3.1. In the main text, we only

Table 6: Scale-invariant ResNet-18 on CIFAR-10 with SGDM, B=128, and cosine learning rate scheduling.

LR	WD	3125	6250	12500	25000	50000
6.4	0.0001	78.36	83.08	85.96	87.98	87.36
3.2	0.0001	**79.02**	84.51	88.21	90.89	92.78
1.6	0.0001	78.35	**84.97**	89.32	92.52	94.30
0.8	0.0001	76.80	84.88	**89.48**	**93.00**	95.08
0.4	0.0001	75.18	84.09	88.99	92.89	**95.32**
0.2	0.0001	72.60	83.01	88.79	92.81	95.14
0.1	0.0001	71.12	82.15	87.89	92.12	94.92

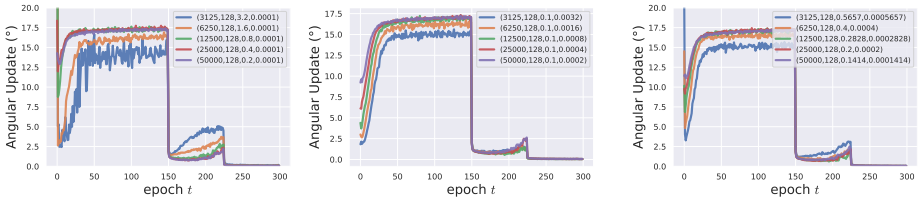
(a) Tuning LR

LR	WD	3125	6250	12500	25000	50000
0.1	0.0064	78.69	83.37	85.96	87.70	87.26
0.1	0.0032	**79.42**	84.74	88.40	91.17	92.95
0.1	0.0016	78.88	**85.06**	**89.67**	92.52	94.87
0.1	0.0008	76.13	84.94	89.64	**93.18**	95.39
0.1	0.0004	74.02	84.16	89.17	92.95	**95.51**
0.1	0.0002	71.53	82.05	88.82	92.74	95.25
0.1	0.0001	71.12	82.15	87.89	92.12	94.92

(b) Tuning WD

LR	WD	3125	6250	12500	25000	50000
0.8	0.0008	78.92	83.06	85.98	87.92	87.39
0.5657	0.0005657	**79.33**	84.61	88.21	90.94	92.81
0.4	0.0004	78.53	**85.20**	89.57	92.32	94.69
0.2828	0.0002828	76.53	85.00	**89.77**	**93.16**	95.21
0.2	0.0002	74.61	84.21	89.29	92.83	**95.37**
0.1414	0.0001414	72.68	82.76	88.67	92.60	95.11
0.1	0.0001	71.12	82.15	87.89	92.12	94.92

(c) Tuning LR,WD



(a) (b) (c)

Fig. 7: Angular update per epoch of ResNet-18 with SGDM, and $B = 128$. (a) Tuning LR, (b) tuning WD, and (c) tuning both LR and WD.

Table 7: ResNet-18 on CIFAR-10 with SGDM, B=128.

LR	WD	3125	6250	12500	25000	50000
6.4	0.0001					
3.2	0.0001	73.58	77.20	82.51	83.25	79.93
1.6	0.0001	77.23	83.23	87.45	91.16	92.86
0.8	0.0001	**77.35**	**84.26**	**89.01**	92.62	94.69
0.4	0.0001	73.75	83.22	88.66	**92.69**	**95.18**
0.2	0.0001	71.86	82.69	88.48	92.58	95.09
0.1	0.0001	72.32	81.71	88.02	91.96	94.87

(a) Tuning LR

LR	WD	3125	6250	12500	25000	50000
0.1	0.0064	77.43	80.63	81.95	80.19	81.90
0.1	0.0032	79.06	83.74	87.23	88.80	89.85
0.1	0.0016	**79.20**	**84.88**	89.38	92.38	93.90
0.1	0.0008	77.78	84.83	**89.80**	**93.33**	95.14
0.1	0.0004	75.23	83.75	89.43	93.06	**95.37**
0.1	0.0002	72.83	83.31	88.80	92.67	95.18
0.1	0.0001	72.32	81.71	88.02	91.96	94.87

(b) Tuning WD

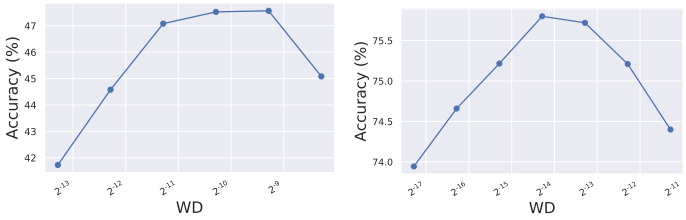
LR	WD	3125	6250	12500	25000	50000
0.8	0.0008	76.39	80.29	80.94	78.81	51.82
0.5657	0.0005657	77.98	83.49	87.12	89.10	89.71
0.4	0.0004	**78.05**	**84.81**	88.70	91.68	93.63
0.2828	0.0002828	75.89	84.67	**89.48**	92.99	94.86
0.2	0.0002	73.19	84.03	89.19	**93.04**	**95.42**
0.1414	0.0001414	72.98	83.07	88.68	92.64	95.06
0.1	0.0001	72.32	81.71	88.02	91.96	94.87

(c) Tuning LR,WD

report experiments with fixed batch size $B = 128$. We report other results in the Appendix. Table 7 shows the results. Figure 7 shows the angular update per epoch of the yellow cells of Table 7. Here, the angular update per epoch was obtained with only scale-invariant weights.

6 Efficient Hyperparameter Search

In this section, we discuss a practical application for efficient hyperparameter search. Finding the tuning factor is sufficient for obtaining near optimum performance, *i.e.*, even if any hyperparameter is changed, a good performance can be obtained if the tuning factor is maintained. This motivates the question: can we find a near optimal hyperparameter with a small portion of data samples? Here, we show that it is not necessary to use all the training data to find



(a) (b)

Fig. 8: Classification accuracy of EfficientNet-B0 on ImageNet according to λ: (a) $N = 80$k, and (b) $N = 1.28$M

the hyperparameter. We search the hyperparameter of training EfficientNet-B0 [26] on ImageNet dataset [22] which comprisis 1.28M samples. EfficientNet has scale-variant parts, *e.g.*, squeeze and excitation module [10] and SiLU activation function [21]. We show that even in the presence of such scale-variant parts, the tuning factor still works well as long as the network has a normalization layer.

We first find the tuning factor with 80k randomly sampled data, which is 1/16 of the entire sample. We apply data augmentation, including random horizontal flip and resizing ratio between 0.08 and 1 and aspect ratio between 3/4 and 4/3. We train for 200 epochs using SGD with batch size 512, LR 0.1, momentum coefficient 0.9; the LR is divided by 10 at 100, 150, 180 epochs (50%, 75%, and 90% of the entire duration) and only the WD is tuned. Such a LR and momentum values are typically used in many architectures [5,4,11] which are trained using SGDM. The original EfficientNet study used an RMSprop optimizer with WD 1e-5 and an initial LR of 0.256 that decays by 0.97 every 2.4 epoch. Figure 8 (a) shows the result. The best result is obtained when $\lambda = 0.0016$; we obtain the tuning factor $\frac{80\text{k}\cdot 0.1\cdot 0.0016}{512\cdot(1-0.9)}$. Thus, using the factor, we can expect that the optimum is around $\lambda =$ 1e-04 when trained with the entire data sample. Figure 8 (b) shows the result for the entire dataset. The best result can be seen when $\lambda =$ 5e-05; but the second best result is seen when $\lambda =$ 1e-04. This shows that we can find near optimal hyperparameters with significantly fewer iterations.

7 Conclusion

This study observed that if hyperparameters are well tuned, the scale-invariant network shows a similar degree of angular update during an epoch. We derived the relation between hyperparameters and angular update per epoch based on this novel observation by adopting SDE. We proposed the concept of a tuning factor, and performed rigorous hyperparameter tuning to show the validity. We also proposed an efficient hyperparameter search method only using a small portion of training samples.

Acknowledgements. This research was supported by the Engineering Research Center Program through the National Research Foundation of Korea (NRF) funded by the Korean Government MSIT (NRF-2018R1A5A1059921).

References

1. Ba, J.L., Kiros, J.R., Hinton, G.E.: Layer normalization. arXiv preprint arXiv:1607.06450 (2016)
2. Gardiner, C.W., et al.: Handbook of stochastic methods, vol. 3. springer Berlin (1985)
3. Goyal, P., Dollár, P., Girshick, R., Noordhuis, P., Wesolowski, L., Kyrola, A., Tulloch, A., Jia, Y., He, K.: Accurate, large minibatch sgd: Training imagenet in 1 hour. arXiv preprint arXiv:1706.02677 (2017)
4. Han, D., Kim, J., Kim, J.: Deep pyramidal residual networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 5927–5935 (2017)
5. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 770–778 (2016)
6. Hoffer, E., Banner, R., Golan, I., Soudry, D.: Norm matters: efficient and accurate normalization schemes in deep networks. In: Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., Garnett, R. (eds.) Advances in Neural Information Processing Systems. vol. 31. Curran Associates, Inc. (2018), https://proceedings.neurips.cc/paper/2018/file/a0160709701140704575d499c997b6ca-Paper.pdf
7. Hoffer, E., Hubara, I., Soudry, D.: Train longer, generalize better: closing the generalization gap in large batch training of neural networks. arXiv preprint arXiv:1705.08741 (2017)
8. Hoffer, E., Hubara, I., Soudry, D.: Fix your classifier: the marginal value of training the last weight layer (2018)
9. Howard, A., Sandler, M., Chu, G., Chen, L.C., Chen, B., Tan, M., Wang, W., Zhu, Y., Pang, R., Vasudevan, V., et al.: Searching for mobilenetv3. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 1314–1324 (2019)
10. Hu, J., Shen, L., Sun, G.: Squeeze-and-excitation networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 7132–7141 (2018)
11. Huang, G., Liu, Z., Van Der Maaten, L., Weinberger, K.Q.: Densely connected convolutional networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 4700–4708 (2017)
12. Ioffe, S., Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: International conference on machine learning. pp. 448–456. PMLR (2015)
13. Krizhevsky, A., Hinton, G., et al.: Learning multiple layers of features from tiny images (2009)
14. Le, Y., Yang, X.: Tiny imagenet visual recognition challenge. CS 231N **7**(7), 3 (2015)
15. Lewkowycz, A., Gur-Ari, G.: On the training dynamics of deep networks with l_2 regularization. arXiv preprint arXiv:2006.08643 (2020)
16. Li, Q., Tai, C., Weinan, E.: Stochastic modified equations and adaptive stochastic gradient algorithms. In: International Conference on Machine Learning. pp. 2101–2110. PMLR (2017)
17. Li, Q., Tai, C., Weinan, E.: Stochastic modified equations and dynamics of stochastic gradient algorithms i: Mathematical foundations. The Journal of Machine Learning Research **20**(1), 1474–1520 (2019)

18. Li, Z., Arora, S.: An exponential learning rate schedule for deep learning. In: International Conference on Learning Representations (2020), https://openreview.net/forum?id=rJg8TeSFDH
19. Li, Z., Lyu, K., Arora, S.: Reconciling modern deep learning with traditional optimization analyses: The intrinsic learning rate. Advances in Neural Information Processing Systems **33** (2020)
20. Li, Z., Malladi, S., Arora, S.: On the validity of modeling sgd with stochastic differential equations (sdes). arXiv preprint arXiv:2102.12470 (2021)
21. Ramachandran, P., Zoph, B., Le, Q.V.: Searching for activation functions. arXiv preprint arXiv:1710.05941 (2017)
22. Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A.C., Fei-Fei, L.: ImageNet Large Scale Visual Recognition Challenge. International Journal of Computer Vision (IJCV) **115**(3), 211–252 (2015). https://doi.org/10.1007/s11263-015-0816-y
23. Smith, S., Elsen, E., De, S.: On the generalization benefit of noise in stochastic gradient descent. In: International Conference on Machine Learning. pp. 9058–9067. PMLR (2020)
24. Smith, S.L., Kindermans, P.J., Ying, C., Le, Q.V.: Don't decay the learning rate, increase the batch size. arXiv preprint arXiv:1711.00489 (2017)
25. Smith, S.L., Le, Q.V.: A bayesian perspective on generalization and stochastic gradient descent. arXiv preprint arXiv:1710.06451 (2017)
26. Tan, M., Le, Q.: Efficientnet: Rethinking model scaling for convolutional neural networks. In: International Conference on Machine Learning. pp. 6105–6114. PMLR (2019)
27. Ulyanov, D., Vedaldi, A., Lempitsky, V.: Instance normalization: The missing ingredient for fast stylization. arXiv preprint arXiv:1607.08022 (2016)
28. Van Laarhoven, T.: L2 regularization versus batch and weight normalization. arXiv preprint arXiv:1706.05350 (2017)
29. Wan, R., Zhu, Z., Zhang, X., Sun, J.: Spherical motion dynamics: Learning dynamics of normalized neural network using sgd and weight decay. Advances in Neural Information Processing Systems **34** (2021)
30. Welling, M., Teh, Y.W.: Bayesian learning via stochastic gradient langevin dynamics. In: Proceedings of the 28th international conference on machine learning (ICML-11). pp. 681–688. Citeseer (2011)
31. Wu, Y., He, K.: Group normalization. In: Proceedings of the European conference on computer vision (ECCV). pp. 3–19 (2018)
32. Xie, S., Girshick, R., Dollár, P., Tu, Z., He, K.: Aggregated residual transformations for deep neural networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 1492–1500 (2017)
33. Yun, J., Kim, B., Kim, J.: Weight decay scheduling and knowledge distillation for active learning. In: European Conference on Computer Vision. pp. 431–447. Springer (2020)
34. Zagoruyko, S., Komodakis, N.: Wide residual networks. In: British Machine Vision Conference 2016. British Machine Vision Association (2016)
35. Zhang, G., Wang, C., Xu, B., Grosse, R.: Three mechanisms of weight decay regularization. In: International Conference on Learning Representations (2019), https://openreview.net/forum?id=B1lz-3Rct7