L₂ regularization for weights in neural networks is widely used as a standard training trick. In addition to weights, the use of batch normalization involves an additional trainable parameter γ, which acts as a scaling factor. However, L₂ regularization for γ remains an undiscussed mystery and is applied in different ways depending on the library and practitioner. In this article, we study whether L₂ regularization for γ is valid. To explore this issue, we consider two approaches: (1) variance control to make the residual network behave like an identity mapping and (2) stable optimization through the improvement of effective learning rate. Through two analyses, we specify the desirable and undesirable γ to apply L₂ regularization and propose four guidelines for managing them. In several experiments, we observed that applying L₂ regularization to applicable γ increased 1% to 4% classification accuracy, whereas applying L₂ regularization to inapplicable γ decreased 1% to 3% classification accuracy, which is consistent with our four guidelines. Our proposed guidelines were further validated through various tasks and architectures, including variants of residual networks and transformers.

CCS Concepts: • Computing methodologies → Neural networks; Regularization;

Additional Key Words and Phrases: L₂ regularization, weight decay, batch normalization, residual network, effective learning rate, deep learning

ACM Reference Format:
Bum Jun Kim, Hyeyeon Choi, Hyeonah Jang, and Sang Woo Kim. 2024. Guidelines for the Regularization of Gammas in Batch Normalization for Deep Residual Networks. ACM Trans. Intell. Syst. Technol. 15, 3, Article 44 (March 2024), 20 pages. https://doi.org/10.1145/3643860

1 INTRODUCTION

Deep neural networks have exhibited remarkable performance in various fields [19, 20, 25, 29, 31]. Previously, large neural networks with deep and wide architectures were considered difficult to train. However, various regularization techniques, such as L₂ regularization [36], batch normalization (BN) [16], and residual learning [11], have made it possible to train large neural networks, leading to successful performance.

Since the classic machine learning era, L₂ regularization has been applied as a restriction on the weights W of neural networks and has not been applied to bias b. However, trainable parameters
in modern neural networks are not limited to weights $W$ and bias $b$. BN outputs $y_i\hat{x}_i + \beta_i$ from normalized feature $\hat{x}$, where $\gamma$ and $\beta$ are also trainable parameters. Because $\beta$ plays a similar role to bias $b$, we ignore $L_2$ regularization of $\beta$. However, $\gamma$ controls the scale of the feature map and can be viewed as a special case of weights (Section 3). Despite the similar roles of $\gamma$ and weights, $L_2$ regularization of the $\gamma$ parameters of BN remains an undiscovered mystery. Moreover, each deep learning library and practice provides a different method. Here, we describe the $L_2$ regularization practice of major libraries in deep learning. Among the many libraries publicly available, we discuss three major libraries: TensorFlow, Keras, and PyTorch.

**TensorFlow and Keras.** To implement $L_2$ regularization, a regularizer option needs to be set for each layer. For example, `kernel_regularizer` can be applied to the convolution and fully connected layers to implement $L_2$ regularization on weight $W$. $L_2$ regularization of $\gamma$ is functionally supported through `gamma_regularizer`, which may be set to the BN layer. In practice, however, `gamma_regularizer` is rarely used. To our knowledge, in the TensorFlow and Keras official tutorials and code, practical use of `gamma_regularizer` does not exist. In other words, $L_2$ regularization in TensorFlow and Keras generally means using `kernel_regularizer` on weights, and $L_2$ regularization has not been applied to $\gamma$.

**PyTorch.** $L_2$ regularization is implemented by applying a weight decay at the optimizer level. However, the weight decay of PyTorch is applied to all trainable parameters, including all $W$, $b$, $\gamma$, and $\beta$. Although the validity of weight decay for $b$ and $\beta$ is also arguable, we focus on whether PyTorch’s practice of applying weight decay to $\gamma$ is valid.

**Practice and empirical observation.** In several works [9, 13, 17], $L_2$ regularization was not applied to $\gamma$ and $\beta$. However, Wu and He [30] and Yan et al. [33] used $L_2$ regularization on $\gamma$ and $\beta$, while Wu and He [30] turned it off during fine-tuning. Summers and Dinneen [24] empirically observed that $L_2$ regularization on $\gamma$ and $\beta$ often improves performance depending on the architecture.

Because these practices lack theoretical analysis, in this article, we explore whether it is desirable to apply $L_2$ regularization to the $\gamma$ parameter of BN. First, we claim that $\gamma$ plays a role in controlling variance in residual networks. We show that the variance of the features in the residual block is either accumulated or reset according to the layer arrangement. For better behavior in residual networks, we propose a strategy of making the accumulated variance small and the reset variance large (Section 2). Second, we present an analysis of the effective learning rate from the optimization perspective of $\gamma$ (Section 3). Through our theoretical analysis, we present four guidelines for managing $\gamma$ (Table 2).

The validity of the four guidelines is confirmed through several experiments (Section 4). We demonstrate a performance decrease due to incorrect $L_2$ regularization on $\gamma$ and a performance increase due to correct $L_2$ regularization on $\gamma$. We observed this phenomenon in various residual networks and transformers.

## 2 VARIANCE ANALYSIS IN RESIDUAL NETWORKS

In this article, we target deep residual networks because they are widely used as standard architectures and involve many variants. The deep residual networks in this study indicate common residual networks, such as ResNet-101 [11], Wide ResNet-50-2 [34], or ResNeXt-101-32x8d [32], considering their original and PreAct [12] versions. In residual networks, an input image is first passed through the early stage. Then, four stages are applied, and each stage is composed of several residual blocks. Each residual block consists of a skip connection and a residual branch, in which several weight, BN, and ReLU layers are placed in the residual branch [11].

ACM Trans. Intell. Syst. Technol., Vol. 15, No. 3, Article 44. Publication date: March 2024.
Table 1. List of Notations Used in This Article

| Notation | Meaning |
|----------|---------|
| $x_l$   | Feature at the $l$-th block. |
| $x_{l,i}$ | The $i$-th element of feature at the $l$-th block. |
| $x_s$ | The first input feature map for each stage of ResNet ($l = s$). |
| $f_l$ | Residual branch function at the $l$-th block. |
| $f_{l,i}$ | The $i$-th element of residual branch function at the $l$-th block. |
| $g_l$ | Skip branch function at the $l$-th downsampling block. |
| $g_{l,i}$ | The $i$-th element of skip branch function at the $l$-th downsampling block. |
| $B_{l,i}$ | The $i$-th element of the output of BN at the $l$-th block. |
| $y_{l,i}$ | The $i$-th $\gamma$ of BN at the $l$-th block. |
| $y_{last}$ | $\gamma$ in the last BN in the residual branch. |
| $y_{down}$ | $\gamma$ in the BN that affects the skip path in the downsampling block. |
| $y_0$ | $\gamma$ in the very first BN in the residual network. |
| $y_{others}$ | $\gamma$ in the BN that do not correspond to the above three categories. |
| $\Gamma_l$ | Diagonal matrix with a $\gamma$ element at the $l$-th block. |
| $(\Gamma_l)_{i,j}$ | $(i, j)$ element of diagonal matrix with a $\gamma$ element at the $l$-th block. |
| $\nabla\Gamma_l y$ | Gradient of $y$ with respect to $\Gamma_l$. |
| ReLU | Rectified linear unit, max($0, x$). |
| $\lambda$ | $L_2$ regularization coefficient. |
| $\eta$ | Learning rate of gradient descent optimization. |
| $N$ | Normal distribution. |
| $fan_{in}$ | The number of input features. |
| $W$ | Weight matrix. |

Table 2. Preview of Our Guidelines on $L_2$ Regularization

|          | TensorFlow | PyTorch | Proposed Guidelines |
|----------|------------|---------|---------------------|
| $W$      | Applied ✓  | Applied ✓ | Applicable ✓        |
| $y_{last}$ | Rarely applied x | Applied ✓ | Applicable ✓        |
| $y_{down}$ | Rarely applied x | Applied ✓ | Inapplicable x      |
| $y_0$    | Rarely applied x | Applied ✓ | Inapplicable x      |
| $y_{others}$ | Rarely applied x | Applied ✓ | Applicable ✓        |

We categorize $\gamma$ into four groups, where $y_{last}$ belongs to the last BN in the residual branch, $y_{down}$ to the BN that affects the skip path in the downsampling block, $y_0$ to the very first BN in the residual network, and $y_{others}$ to the BN that do not correspond to the above three categories.

Our assumption is that the condition when the skip connection dominates over the residual branch, which makes the residual block behave more like an identity mapping, is advantageous for residual network training. In fact, in the original ResNet paper, residual learning was introduced with the intention of modelling identity mappings [11]. Goyal et al. [9] reported that initializing $\gamma = 0$ in the last BN of the residual branch causes the residual block to behave like an identity function and eases optimization. De and Smith [5] found that a residual block behaves like an identity mapping because the variance of the skip connection is larger than the variance of the residual branch. In their paper, the $\gamma$ parameter of BN was assumed to be 1, but we show here that variance in the residual block varies according to $\gamma$.
We consider the bias parameter $b$ in the weight layers and $\beta$ in the BN layers to be 0. This is because ResNet employs convolutional layers without bias. Further, $\beta$ is initialized to zero and stays close to it during training. We conjecture that $\beta$ around zero is advantageous to preserve zero-centered ReLU input. Therefore, $L_2$ regularization of $b$ or $\beta$ is ignored in this article. In BN, each feature map is normalized to zero mean with unit variance and then rescaled using $\gamma$. In other words, the $i$-th element of BN with $\gamma_i$ outputs $B_i(x)$, where $E[B_i(x)] = 0$ and $Var[B_i(x)] = \gamma_i^2$.

The mean and variance in this study are computed along the sample axis to obtain the $i$-th mean and variance. Consider $x \in \mathbb{R}^{N \times D}$, where $N$ denotes mini-batch size and $D$ denotes the number of dimensions. Targeting the $i$-th channel element $x_i \in \mathbb{R}^N$, we define mean as $E[x_i] := \frac{1}{N} \sum_{n=1}^{N} x_{n,i}$. The same goes for variance as $Var[x_i] := \frac{1}{N} \sum_{n=1}^{N} (x_{n,i} - E[x_i])^2$. The term “weight layers” used below indicates convolution layers in ResNet but will be described by generalizing to fully connected layers. We assume that each element of the weights comes from the He initialization $N(0, 2/fan_{in})$ [10]. See Table 1 for a list of notations used in this study.

2.1 Case 1. Basic Block in PreActResNet

PreActResNet [12] is a modified version of the original ResNet [11] and is also called ResNetV2. The residual branch of PreActResNet consists of two or three [BN–ReLU–Weight] blocks (Figure 1, Left). The $l$-th residual block outputs $x_{l+1}$, which is the sum of the residual branch $f_l(x_l)$ and skip connection $x_l$, i.e., $x_{l+1} = x_l + f_l(x_l)$. As the residual branch $f_l(x_l)$ ends with a weight layer that has zero-mean weight, $f_l(x_l)$ becomes a weighted summation using zero-mean weights. Therefore, we obtain $E[f_l(x_l)] = 0$ regardless of $x_l$. Similarly, because $x_l f_l(x_l)$ becomes weights summation using zero-mean weights, we obtain $E[x_l f_l(x_l)] = 0$. Using these results, we obtain zero covariance between the residual branch and the skip connection $Cov[x_l, f_l(x_l)] = E[x_l f_l(x_l)] - E[x_l]E[f_l(x_l)] = 0$, which leads to $Var[x_{l+1}] = Var[x_l] + Var[f_l(x_l)]$ [3, 5].

Here, we examine the variance of the residual branch $Var[f_l(x_l)]$. Consider the residual branch with two [BN–ReLU–Weight] blocks. Because of the normalization in the second BN, the $\gamma$ of the
Fig. 2. Empirical validation of Equation (5). The variance grows by stacking blocks and is affected by \( \gamma \).

First BN and \( \|W_1\| \) do not affect the variance of the residual branch. Therefore, the variance of the residual branch is determined by the second [BN–ReLU–Weight] block. By the mean and variance of \( W \) from He initialization and BN, the variance of the \( i \)-th element of the residual branch is

\[
\text{Var}[f_{l,i}(x_l)] = 2 \cdot E[\text{ReLU}(B_{2,l,i}(x_l))^2] = E[B_{2,l,i}(x_l)^2] = \text{Var}[B_{2,l,i}(x_l)] = \gamma_{2,l,i}^2. \tag{3}
\]

The second equality holds because \( E[\text{ReLU}(X)^2] = E[X^2]/2 \) for normalized \( X \) [10]. Thus,

\[
\text{Var}[x_{l+1,i}] = \text{Var}[x_{l,i}] + \gamma_{2,l,i}^2. \tag{4}
\]

Starting from the \( s \)-th residual block, imagine that the variance of the residual branch is accumulated in a row. The variance of the \( l \)-th residual block is

\[
\text{Var}[x_{l,i}] = \text{Var}[x_{s,i}] + \sum_{m=s}^{l-1} \gamma_{2,m,i}^2, \tag{5}
\]

where \( x_s \) denotes the first input feature map for each stage of ResNet. As such, in the residual block, the variance of the residual branch is accumulated. The accumulation of variance can also be observed empirically (Figure 2), whose values matched the theoretical expectations (Equation (5)).

Here, we want to make the residual block satisfy \( \text{Var}[x_{l,i}] > \text{Var}[f_{l,i}(x_l)] \) so that the skip connection dominates over the residual branch. From Equation (3) and Equation (5), the inequality \( \text{Var}[x_{s,i}] + \sum_{m=s}^{l-1} \gamma_{2,m,i}^2 > \gamma_{2,l,i}^2 \) requires two conditions:

1. \( \gamma_{2,l,i}^2 \) should be small for all \( l \);
2. \( \text{Var}[x_{s,i}] \) should be large.
In fact, the dominance of skip connection can be achieved without requiring the two conditions, but explicitly satisfying the two conditions is more reliable. The variance at the stage starting point in the second condition is discussed later in Cases 3 and 4. According to the first condition, $\gamma$ in the second BN in all residual blocks should be small to reduce the variance of the residual branch. Similarly, when the residual branch consists of three [BN–ReLU–Weight] blocks, the $\gamma$ in the third BN should be small. Thus, in order for the skip connection to dominate over the residual branch, we should control $\gamma$ in the last BN. Therefore, we obtain \[ \text{Var}_{l+2}^2 f_l(x_{l+1}) = \gamma_{l+2}^2 (x_{l+2})^2 \] for the original ResNet, consecutive application of the ReLU operation makes $E[x_{l+1}]$ significantly smaller compared with other terms. From this property, we adopt $E[x_{l+1}] = 0$. Now we obtain

\[
\text{Var}[x_{l+1}, i] = \frac{1}{2} E \left[ (x_{l+1} + f_l(x_{l+1}))^2 \right] \\
= \frac{1}{2} \left( E[x_{l+1}^2] + E[f_l(x_{l+1})]^2 \right) \\
= \frac{1}{2} \left( \text{Var}[x_{l+1}] + \text{Var}[f_l(x_{l+1})] \right) \\
= \frac{1}{2} \left( \text{Var}[x_{l+1}] + \gamma_{l+1}^2 x_{l+1}^2 \right).
\]

Thus, the added variance is halved. Here, we introduce the following substitutions: $a_l = 2^l \text{Var}[x_{l+1}, i]$ and $b_l = 2^l \gamma_{l+1}^2 x_{l+1}^2$. Rewriting Equation (9), we have that $a_{l+1} = a_l + b_l$ and, thus, $a_l = a_s + \sum_{m=s}^{l-1} b_m$. Therefore, we obtain

\[
\text{Var}[x_{l+1}, i] = \frac{1}{2^{l-s}} \text{Var}[x_s, i] + \sum_{m=s}^{l-1} \frac{1}{2^{l-m}} \gamma_{l+1}^2 x_{l+1, i}^2.
\]

As such, the variance of the residual branch is accumulated here as well, but because the variance of the last ReLU is halved, the variance of the previous block decays. We call this half variance accumulation. For the skip connection to dominate, the condition $\text{Var}[x_{l+1}, i] > \text{Var}[f_l(x_{l+1})]$ requires that (1) $\gamma_{l+1}^2 x_{l+1}^2$ should be small and (2) $\text{Var}[x_s, i]$ should be large. Therefore, it is desirable to decay $\gamma_{l+1}$, and thus, Guideline 1 is valid for the original ResNet as well.
Fig. 3. Downsampling block in (left) PreActResNet and (right) original ResNet (Case 3). For the weight block, a rectangle shape represents convolution with stride 1, whereas a trapezoid shape indicates convolution with stride 2.

2.3 Case 3. Downsampling Block in ResNet

Convolutional neural networks (CNNs) downsample high-dimensional image features to low-dimensional ones using pooling layers [18]. For ResNet, downsampling is performed using strided convolution [23] as well as pooling. For the original ResNet, at the end of each stage, a downsampling block (Figure 3, right) is applied, where the strided convolution is used in both the first weight layers in the residual branch and the skip path. Here, the skip path of the downsampling block is composed of strided convolution and BN rather than identity. We denote the skip path output as $g_l(x_l) = B_{down,l}(W_{down,l}x_l)$. Because each branch ends with BN, the variance of each branch is determined by the last BN as follows:

$$Var[g_{l,i}(x_l)] = \gamma_{down,l,i}^2$$  \hspace{1cm} (11)

$$Var[f_{l,i}(x_l)] = \gamma_{last,l,i}^2$$  \hspace{1cm} (12)

From $x_{l+1} = ReLU(g_l(x_l) + f_l(x_l))$, the variance after the downsampling block is

$$Var[x_{l+1,i}] = \frac{1}{2}(Var[g_{l,i}(x_l)] + Var[f_{l,i}(x_l)])$$  \hspace{1cm} (13)

$$= \frac{1}{2}(\gamma_{down,l,i}^2 + \gamma_{last,l,i}^2).$$  \hspace{1cm} (14)

Note that the output variance of the downsampling block is not affected by the input variance and is newly determined by $\gamma_{down}$ and $\gamma_{last}$. We call this variance reset.
Again, it is desirable to decay $\gamma_{last}$ so that the residual branch does not dominate. However, it is not desirable to decay $\gamma_{down}$. There are two reasons for this. First, in order for the skip path in the downsampling block to dominate, we should decay $\gamma_{last}$ and should not decay $\gamma_{down}$. Second, it is desirable to have a large reset variance. The output variance of the downsampling block becomes the input variance of the successive $(l + 1)$-th residual block, which is the variance at the stage starting point $x_s$. In Cases 1 and 2, we confirmed that a large $\text{Var}[x_{s,i}]$ is favorable. Thus, a large output variance of the downsampling block is desirable. In other words, if we decay both $\gamma_{down}$ and $\gamma_{last}$, the reset variance decreases, which is undesirable. Therefore, we claim that $\gamma_{down}$ should not be subjected to $L_2$ regularization to ensure that the reset variance appears dominant afterward.

In this regard, we present the following guideline:

**Guideline 2.** Because the downsampling block plays the role of variance reset, we should not decay $\gamma_{down}$.

Now, we check whether Guideline 2 is also applicable to the downsampling block of PreActResNet. In the downsampling block of PreActResNet, before starting both the residual branch and skip path, block input $x_l$ is subjected to the first BN and ReLU (Figure 3, left). The first BN does not affect the variance of the residual branch but does affect the variance of the skip path. For skip path $g_l(x_l) = W_{down,l} \text{ReLU}(B_{1,l}(x_l))$, we have that $\text{Var}[g_{l,i}(x_l)] = 2 \cdot E[\text{ReLU}(B_{1,l,i}(x_l))^2] = \gamma_{1,l,i}^2$. For the residual branch, similar to Case 1, we have that $\text{Var}[f_{l,i}(x_l)] = \gamma_{2,l,i}^2$. Thus, the variance after the residual block is

$$\text{Var}[x_{l+1,i}] = \text{Var}[g_{l,i}(x_l)] + \text{Var}[f_{l,i}(x_l)]$$

$$= \gamma_{1,l,i}^2 + \gamma_{2,l,i}^2.$$  \hspace{1cm} (15)

Thus, the downsampling block of PreActResNet plays the role of variance reset as well. The variance reset can also be observed empirically (Figure 4), where the output variance of the downsampling block was determined by the two $\gamma$. To ensure that the reset variance is dominant afterward, we should not decay $\gamma_1$ at the first BN. For convenience, we also call this parameter $\gamma_{down}$. This guideline is also supported by empirical evidence, where applying $L_2$ regularization to $\gamma_{down}$ rather decreased performance (Figure 5).

### 2.4 Case 4. Early Stage in ResNet

The input variance of the first residual block is determined by the early stage of ResNet before the residual block starts. The early stage of ResNet, which is also called the stem, consists of convolution, BN, ReLU, and pooling layers without a skip connection in the original ResNet (Figure 6, right). Ignoring pooling, from the output of early stage $x_1 = \text{ReLU}(B_0(W_0x_0))$, we have that

$$\text{Var}[x_{1,i}] = \frac{1}{2} E[B_{0,i}(W_0x_0)]^2$$

$$= \frac{1}{2} \gamma_{0,i}^2.$$  \hspace{1cm} (17)

Therefore, $\gamma_0$ sets the output variance of the early stage. The output of the early stage becomes the input of the residual block of the first stage. Because a large variance at the stage starting point $x_s$ is favorable (Cases 1 and 2), $\gamma_0$ should be large. In this regard, we present the following guideline:

**Guideline 3.** Because the early stage plays the role of variance set, we should not decay $\gamma_0$.

In the case of PreActResNet, BN is not applied in the early stage (Figure 6, left). Thus, we do not have to consider $\gamma_0$. 

ACM Trans. Intell. Syst. Technol., Vol. 15, No. 3, Article 44. Publication date: March 2024.
Fig. 4. Empirical validation of Equation 16, \( \text{Var}[x_{l+1,i}] = \gamma_{1,l,i}^2 + \gamma_{2,l,i}^2 \), where the former \( \gamma_{1,l,i} \) is denoted as \( \gamma_{\text{down}} \), and \( \gamma \) in legend corresponds to the latter \( \gamma_{2,l,i} \). For example, when the latter is 1, we have a graph of \( y = x^2 + 1 \), which matches well with empirical results.

Fig. 5. Performance degradation caused by \( L_2 \) regularization on \( \gamma_{\text{down}} \) with various \( \lambda \) for ResNet-152 on the PET dataset. If \( L_2 \) regularization is applied to \( \gamma_{\text{down}} \), regardless of the choice of \( \lambda \), the test accuracy decreased compared with not applying \( L_2 \) regularization to it.
Finally, in the residual branch, additional $\gamma$ parameters exist in the BN before the last BN for both PreActResNet and the original ResNet. We call those parameters $\gamma_{\text{others}}$. Because they do not affect the variance of the residual branches, our variance analysis cannot explain their role. The $\gamma_{\text{others}}$ parameters are discussed again in Section 3. Here, we simply present the conclusion.

**Guideline 4.** Though other BNs do not determine the variance, $L_2$ regularization on $\gamma_{\text{others}}$ helps optimization by improving the effective learning rate. Thus, we should decay $\gamma_{\text{others}}$.

Further, using ResNet-50, we measured the variance of each feature map (Figure 7). First, the variance is (re)set at layer 1-0; then, the variance decreases at layer 1-1, which is expected from the half variance accumulation (Equations (9) and (10)). In layer 2-0, where the subsequent stage begins, the variance is reset to a larger value, which decreases again from layer 2-1 to layer 2-3. As such, both the variance reset to a larger value and the half variance accumulation with decreasing value appear in each residual block, which agrees with our claim. Note that through this tendency on stages one to three, one exception occurred in the last stage, where the variance rather increased by consecutive blocks within the stage. We conjecture that the features in the last stage have extraordinary characteristics due to the high-level pattern they contain [35], which requires an exception in our explanation.

### 2.5 Case 5. Transformer Block

The transformer is attracting attention for its high performance in various tasks, including computer vision [8] and natural language processing [7, 27]. The transformer uses a series of transformer blocks, which consists of residual branches with self-attention and multilayer perceptron blocks. In contrast to ResNet, the transformer uses LayerNorm (LN) [1]. However, LN also rescales normalized features using $\gamma$ and $\beta$. Therefore, our variance analysis can be generalized to transformers and can explain how the $\gamma$ parameter of LN affects the variance within the transformer block. See the Appendix for the mathematical details regarding the roles of $\gamma_1$ and $\gamma_2$ in the first and second LN blocks. Here, we simply present the following guideline for the transformer block.

**Guideline 1T.** Because a transformer block accumulates the variance, we should decay $\gamma_1$ and $\gamma_2$. 
Fig. 7. Variance of each feature map of ResNet-50 pre-trained on ImageNet. After the variance is (re)set to a larger value at the stage-starting point, it decreases by consecutive blocks within the stage.

Table 3. Our Observations that Vision Transformers (ViTs) Exhibited Variance Accumulation as Well

| Model   | 0  | 1  | 2  | 3  | 4  | 5  | 6  | 7  | 8  | 9  | 10 | 11 | 12 |
|---------|----|----|----|----|----|----|----|----|----|----|----|----|----|
| ViT-Ti/16 | 0.09 | 4.15 | 8.83 | 14.95 | 20.06 | 25.41 | 30.85 | 36.77 | 42.64 | 49.26 | 52.80 | 58.57 | 63.73 |
| ViT-S/16 | 0.10 | 3.76 | 8.49 | 13.22 | 17.87 | 22.99 | 27.83 | 33.81 | 38.70 | 45.28 | 52.65 | 58.78 | 64.73 |
| ViT-B/16 | 0.10 | 3.54 | 7.47 | 11.61 | 16.09 | 20.84 | 25.98 | 31.02 | 35.62 | 41.88 | 47.88 | 53.66 | 58.76 |

Ti, S, and B indicate the tiny, small, and base models, respectively.

Indeed, passing transformer blocks yielded variance accumulation (Table 3). Note that the transformer has no downsampling block (and, hence, there is no Guideline 2T). Further, some transformers, such as bidirectional encoder representations from transformers (BERT) [7] have LN in the early stage, whose $\gamma_0$ should not be subjected to the $L_2$ regularization (Guideline 3T). Finally, some transformers [27] have another LN between the final transformer block and the head layer. In the same context as Guideline 4, $L_2$ regularization on $\gamma_{other}$ helps optimization (Guideline 4T).

3 EFFECTIVE LEARNING RATE ANALYSIS

In this section, we explore why $\gamma_{other}$ parameters should be decayed even though their scale does not affect the BN output. Several studies [14, 26, 36] argued that in a neural network with BN, the weight scale does not affect the BN output. They introduced the effective learning rate to claim that a smaller weight scale is advantageous for optimization, which can be achieved through $L_2$ regularization on weights. In our article, we note that $\gamma$ is a trainable parameter in a neural network similar to weights. Extending their claim, we investigate the effective learning rate for the optimization of $\gamma$. 

ACM Trans. Intell. Syst. Technol., Vol. 15, No. 3, Article 44. Publication date: March 2024.
Fig. 8. Norm of the first update by the initial value of $γ$. A regression fit yielded a coefficient of $-1.9991$, which matches well with the theoretical expectation of $-2$.

For input feature $x$, BN is composed of two steps: (1) a normalization step $N(\cdot)$ to yield $\hat{x}$ and (2) a linear scaling step $L(\cdot)$ to yield $γi\hat{x}_i + β_i$. We denote the two steps of BN as $[N–L]$. From a series of $[\text{Weight–BN–ReLU}]$ blocks, we decompose BN into $[N–L]$ and investigate the intermediate $[L–\text{ReLU–Weight–N}]$ block. In the $l$-th intermediate block, weight layer holds $W_l$ and $L$ step holds $Γ_l$, where $Γ_l$ is a diagonal matrix with $(Γ_l)_{i,i} = γ_i$ and $(Γ_l)_{i,j} = 0$ for $i \neq j$. From the input feature map $x_l$, the intermediate block outputs $x_{l+1} = y(x_l; W_l, Γ_l) = N[W_l\text{ReLU}(Γ_lx_l)]$.

Here, we decompose the diagonal matrix into its norm and direction, i.e., $Γ_l = \|Γ_l\|^\hat{Γ}$. Then,

$$y(x_l; W_l, Γ_l) = N\left[W_l\text{ReLU}\left(\|Γ_l\|\hat{Γ}x_l\right)\right]$$

$$= N\left[W_l \|Γ_l\|\text{ReLU}(\hat{Γ}x_l)\right]$$

$$= N\left[W_l\text{ReLU}(\hat{Γ}x_l)\right].$$

Thus, the scale of $γ$ in BN does not affect the intermediate block output. If we investigate the gradient of $y(x_l; W_l, Γ_l)$ with respect to $Γ_l$, we obtain

$$\nabla_{Γ_l}y(x_l; W_l, Γ_l) = \frac{1}{\|Γ_l\|}\nabla_{Γ_l}y(x_l; W_l, \hat{Γ}).$$

This scale-variant gradient property was used in van Laarhoven [26] and Hoffer et al. [14] to claim that $L_2$ regularization on weights helps optimization even in neural networks with BN. Thus, effective learning rate analysis is expected to be applicable to $γ$ in BN. From Equation (22), we have that $\nabla_{Γ_l}L(Γ_l) = \frac{1}{\|Γ_l\|}\nabla_{Γ_l}L(\hat{Γ})$ for loss function $L$. First, gradient descent with respect to $Γ_l$ is in the form of

$$Γ_{l,t+1} = Γ_{l,t} - η\nabla_{Γ_{l,t}}L(Γ_{l,t}).$$
Table 4. List of Datasets We Used for Experiments

| Dataset      | # Data | Task                       | Reference                       |
|--------------|--------|----------------------------|---------------------------------|
| Oxford-IIIT PET | 7K     | Image Classification       | Parkhi et al. [22]              |
| NABirds      | 48K    | Image Classification       | Van Horn et al. [15]            |
| Food-101     | 101K   | Image Classification       | Bossard et al. [2]              |
| ImageNet     | 1.28M  | Image Classification       | Deng et al. [6]                 |
| IWSLT-14     | 160K   | German to English Machine Translation | Cettolo et al. [4] |
| GLUE SST-2   | 69K    | Sentiment Classification on Sentence | Wang et al. [28]               |

All these datasets are publicly available.

If we investigate gradient descent with respect to $\hat{\Gamma}_l$, we have that

$$\hat{\Gamma}_{l,t+1} = \hat{\Gamma}_{l,t} - (\eta \|\hat{\Gamma}_{l,t}\|^2) \nabla_{\hat{\Gamma}_{l,t}} L(\hat{\Gamma}_{l,t}).$$

(24)

Here, $\eta \|\Gamma_l\|^2$ is called the effective learning rate [14, 26]. If the scale of $\gamma$ in BN decreases, the effective learning rate increases, which prevents the optimization from saturating due to the large weight norm. Although we followed the derivation in van Laarhoven [26], the derivation in Hoffer et al. [14] shows similar results. Thus, although $L_2$ regularization on $\gamma_{\text{others}}$ does not affect the variance, it improves the effective learning rate and optimization. In this regard, we advocate applying $L_2$ regularization to $\gamma_{\text{others}}$.

Now, we empirically validate Equation (24). Using ResNet-18, we varied the initial value of $\gamma$ and measured the norm of the first update, $\|\hat{\Gamma}_{l,t} - \hat{\Gamma}_{l,0}\|$ (Figure 8). From their log values, a regression fit yielded a coefficient near $-2$, which agrees with $\|\hat{\Gamma}_{l,t+1} - \hat{\Gamma}_{l,t}\| \propto \|\Gamma_l\|^2$.

Note that this effective learning rate analysis is applicable to the $\gamma$ parameters of all four categories. Although applying $L_2$ regularization to $\gamma_{\text{down}}$ and $\gamma_0$ is harmful to variance control, it could be advantageous for improving optimization with respect to the effective learning rate. Nevertheless, due to the potential performance degradation, we advocate not applying $L_2$ regularization to $\gamma_{\text{down}}$ and $\gamma_0$ (Section 4).

4 EXPERIMENTS

We tested whether Guidelines 1 to 4 are valid in practical tasks. We targeted a variety of tasks, datasets, and residual networks for experiments. See Table 4 for a list of datasets we used for experiments.

4.1 Experimental Details

Here, we provide experimental details, including training setup and hyperparameters. For all experiments, we report the average result from three experiments, except for ImageNet. We used the NVIDIA RTX 3090 GPU. For $L_2$ regularization, we conducted a grid search from trials of 0.001, 0.002, 0.005, 0.01, 0.02, 0.05, and 0.1 and selected the $\lambda$ value that performed the best on the validation set. We did not use pre-trained models; we simply trained models from scratch to better observe the behavior of $\gamma$, except for the experiments with pre-trained BERT. All datasets are easily downloadable on their official webpages.

Experiments on PET dataset. For training, a stochastic gradient descent with momentum 0.9, learning rate 0.01, batch size 32, epochs 200, and cosine annealing schedule with 200 iterations were used. For data augmentation, a random resized crop with size 256, random rotation with degree 15, color jitter, random horizontal flip, center crop with size 224, and mean-std normalization were applied. The train/val/test set was split at a ratio of 70:15:15. Within 200 epochs, the model with the best validation accuracy was obtained and evaluated. We used ResNet and its variants from...
torchvision.models, which corresponds to the original ResNet in Figure 1 (right); the residual block is composed of two [Weight–BN–ReLU] blocks for ResNet-18, and three for ResNet-[50, 101, 152]. For Wide ResNet, we denote its size as Wide ResNet-[depth]-[width] and for ResNeXt, we denote its size as ResNeXt-[depth]-[width]x{cardinality}. Experiments on the NABirds and Food-101 datasets also followed this.

**Experiments on the ImageNet dataset.** For training, a stochastic gradient descent with momentum 0.9, learning rate 0.1, epochs 200, warm-up epochs 5, warm-up learning rate 0.0001, cosine annealing schedule, label smoothing 0.1, random erasing with probability 0.6, and RandAugment of magnitude 7 and noise-std 0.5 with increased severity (rand-m7-mstd0.5-inc1) were used. We used ResNet and its variants from pytorch-image-models (timm).

**Experiments on IWSLT-14.** For training, a stochastic gradient descent with momentum 0.9, learning rate 0.1, number of epochs 20, learning rate decay 0.2 at {6, 12, 16} epochs, and mini-batch size 1,500 were used. As in the original transformer paper, we used a warm-up step 4,000, label smoothing 0.1, and dropout probability 0.1. We used a vanilla transformer implemented with PyTorch with the number of layers 6, the hidden size 512, and the number of attention heads 8.

**Experiments on GLUE SST-2.** For training, AdamW with learning rate $10^{-5}$, linear learning rate schedule, mini-batch size 32, number of epochs 3, and maximum sequence length 128 were used. From Hugging Face, we fine-tuned the BERT called bert-base-cased, which is a case-sensitive model and was pre-trained by masked language objective. The model is base size with the number of layers 12, the hidden size 768, and the number of attention heads 12.

### 4.2 Image Classification Tasks

First, in the training task of ResNet and its variants [11, 12, 32, 34], we measured the performance of applying $L_2$ regularization to the weights and $\gamma$ of each group with $\lambda(\|W\|^2_2 + \|\gamma\|^2_2)/2$.

Table 5 shows image classification accuracy on the Oxford-IIIT PET dataset [22] for different $L_2$ regularization setups. First, when the $\gamma_{last}$ parameters were subjected to $L_2$ regularization, the test accuracy increased approximately 1% to 4% compared with the case in which $L_2$ regularization was not applied to them. This is consistent with Guideline 1, which states that we should decay the $\gamma_{last}$. However, when $L_2$ regularization was applied to $\gamma_{down}$, the test accuracy decreased approximately 1% to 3%. This is consistent with Guideline 2, which states that $L_2$ regularization for $\gamma_{down}$ is undesirable. Similarly, for $\gamma_0$ and $\gamma_{others}$, the experimental results agree with Guidelines 3 and 4.

However, for $\gamma_0$, an increase in performance was observed in some models. When the effect of improving the effective learning rate is more dominant than the variance control, it can result in performance improvement. Because the first stage is relatively shallow compared with other stages, the variance from $\gamma_0$ would have a minor effect and improvement of the effective learning rate may become dominant. The coexistence of variance control and improved effective learning rate requires further research.

We observed similar results on other datasets, including NABirds [15] and Food-101 [2] (Table 6). Furthermore, our proposed guidelines were also verified in the ImageNet dataset, a large-scale image classification dataset [6]. Following our guidelines consistently improved the top-1 accuracy as well (Table 7).

### 4.3 Natural Language Processing Tasks

We also tested whether our proposed guidelines are valid for transformers on natural language processing tasks. First, we visited a machine translation task using a transformer. The BLEU score [21] was measured for the German to English translation task using the IWSLT-14 dataset [4].
Table 5. Test Accuracy (%) for the PET Dataset

| Model             | W | W_{\gamma_{last}} | W_{\gamma_{down}} | W_{\gamma_0} | W_{\gamma_{others}} |
|-------------------|---|--------------------|--------------------|--------------|---------------------|
| ResNet-18 [11]    | 83.573 | 85.410 (-1.836) | 82.580 (-0.993) | 82.880 (-0.693) | 84.600 (+1.026) |
| ResNet-50 [11]    | 82.943 | 85.650 (+2.706) | 80.083 (-2.860) | 82.400 (-0.543) | 83.513 (+0.570) |
| ResNet-101 [11]   | 82.010 | 84.420 (+2.410) | 78.940 (-3.070) | 81.376 (+1.026) | 83.120 (+1.110) |
| ResNet-152 [11]   | 80.986 | 85.740 (+4.753) | 79.210 (-1.776) | 83.876 (+1.383) | 85.740 (+1.923) |
| ResNet-50-2 [34]  | 82.943 | 85.650 (+2.706) | 80.083 (-2.860) | 82.400 (-0.543) | 83.513 (+0.570) |
| ResNet-101-2 [34] | 82.460 | 85.140 (+2.680) | 81.200 (-1.260) | 83.453 (+0.993) | 85.740 (+2.170) |
| ResNeXt-50-32x4d  [32] | 84.146 | 85.620 (+1.203) | 84.113 (+0.303) | 84.600 (+1.833) | 87.740 (+1.323) |
| ResNeXt-101-32x8d [32] | 84.023 | 85.410 (+1.386) | 83.606 (-0.416) | 82.970 (-1.053) | 85.290 (+1.266) |

Table 6. Test Accuracy (%) for ResNet-18

| Dataset   | W | W_{\gamma_{last}} | W_{\gamma_{down}} | W_{\gamma_0} | W_{\gamma_{others}} |
|-----------|---|--------------------|--------------------|--------------|---------------------|
| NABirds   | 76.493 | 77.140 (+0.646) | 76.400 (-0.093) | 76.986 (+0.993) | 77.336 (+0.843) |
| Food-101  | 75.303 | 76.143 (+0.840) | 75.086 (-0.216) | 74.783 (-0.520) | 75.740 (+0.436) |
| Proposed Guidelines | − | Guideline 1 Applicable ✓ | Guideline 2 Inapplicable X | Guideline 3 Inapplicable X | Guideline 4 Applicable ✓ |

Table 7. Top-1 Accuracy (%) for the ImageNet Dataset

| Model             | W | W_{\gamma_{last}, \gamma_{others}} | Difference |
|-------------------|---|------------------------------------|------------|
| ResNet-50         | 77.878 | 78.002 (+0.124)                  |
| ResNet-101        | 79.488 | 79.922 (+0.434)                  |
| Wide ResNet-50-2  | 78.924 | 79.266 (+0.342)                  |
| Wide ResNet-101-2 | 80.244 | 80.474 (+0.230)                  |
| ResNeXt-50-32x4d  | 77.534 | 78.274 (+0.740)                  |
| ResNeXt-101-32x8d | 79.888 | 80.106 (+0.218)                  |

measured the performance before and after applying $L_2$ regularization to $\gamma_1$ and $\gamma_2$. The baseline BLEU score was 34.789% when applying $L_2$ regularization to $W$. Here, when applying $L_2$ regularization to $W$ and $\gamma_1$, the BLEU score increased up to 34.906%. Similarly, when applying $L_2$ regularization to $W$ and $\gamma_2$, the BLEU score improved up to 35.139%. This result is consistent with Guideline 1T, which states that a small $\gamma$ is desirable for transformer blocks because they accumulate variance.

Our claim was also verified in the text classification task using BERT for the SST-2 task from GLUE [28]. The test accuracy was 91.881% when applying $L_2$ regularization to $W$. Following Guideline 1T, when applying $L_2$ regularization to $W$, $\gamma_1$, and $\gamma_2$, test accuracy improved to 92.087% as well.
5 DISCUSSION

5.1 Guidelines for Weights

In Section 2, we assumed that the mean and variance of the weight determined by He initialization were maintained during training. However, when the variance of the weights changes, it will affect the variance of the residual block. In this section, we analyze the $L_2$ regularization of weights.

**Variance perspective.** First, consider PreActResNet. Because the variance of the last weight of the residual branch affects the variance of the residual branch, it is desirable to decay the $W_{last}$. Since the downsampling block has a weight layer in the skip path, we should not decay $W_{down}$. As the weight layer in the early stage affects variance at the stage starting point, we should not decay $W_0$. Finally, $W_{others}$ should be decayed.

However, consider the original ResNet. In the residual branch, since the last BN normalizes the scale of weight, $W_{last}$ is not involved in determining the variance of the residual block. Similarly, in the downsampling block, the scale of $W_{down}$ is normalized. In the early stage, the scale of the weight layer is normalized as well. Thus, all weights in the original ResNet do not involve determining the variance of the residual block. For the original ResNet, the performance gain from $L_2$ regularization would be from the improvement of the effective learning rate.

**Initialization perspective.** We discussed the control of variance of residual blocks through $L_2$ regularization of $\gamma$ in training. In addition, the variance of the residual block can be controlled through $\gamma$ initialization. While $\gamma$ of BN is basically initialized to 1, Goyal et al. [9] and Zhang et al. [37] reported that initializing $\gamma_{last}$ of the residual branch to zero induces the residual block to be close to identity mapping and improves performance. This shares a similar context with our Guideline 1.

This practice is known as zero_init_last and is preferred in various libraries. In the original ResNet, $\gamma_{last}$ is initialized to zero. Further, in PreActResNet, zero_init_last is applied to the last weight layer of the residual branch to initialize $W_{last}$ to the zero matrix $O$. This implies that the variance of the residual branch can be controlled not only by $\gamma$ but also by weight.

5.2 On Other Architectures

Besides residual networks, CNNs such as VGG and Inception are often used. Our variance analysis considers a residual network with skip connections; thus, it cannot be applied to VGG or Inception. Here, we discuss $L_2$ regularization for $\gamma$ by considering VGG-style architecture with BN without skip connection.

First, in terms of variance, our variance comparison analysis cannot be applied to CNNs without a skip connection. One possible explanation is that if the operation between each feature map was close to identity, the feature maps would be similar. However, if $\gamma$ decay by applying $L_2$ regularization, the output of the [Weight–BN–ReLU] block approaches zero, and the operation of the block becomes different from the identity. Therefore, it is not desirable to apply $L_2$ regularization to $\gamma$ from the viewpoint of variance.

Second, however, effective learning rate analysis is applicable to VGG-style architecture with BN. Therefore, even in VGG-style architecture with BN without skip connection, since $L_2$ regularization for $\gamma$ improves the effective learning rate, it is desirable to apply $L_2$ regularization to $\gamma$.

As such, for VGG-style architecture with BN, $L_2$ regularization for $\gamma$ is (1) undesirable in terms of variance and (2) desirable in terms of effective learning rate. As mentioned in Section 4, performance improvement is not guaranteed if the two effects coexist. Therefore, we do not advocate $L_2$ regularization for $\gamma$ in VGG-style architecture with BN. This argument is also applicable to the
Inception architecture. Indeed, Summers and Dinneen [24] empirically observed that weight decay for $\gamma$ improved the performance in ResNet-50 but degraded it in Inception-v3.

6 CONCLUSION

In this article, we studied $L_2$ regularization for the $\gamma$ parameters of BN. We theoretically showed that this $\gamma$ should be controlled so that the residual blocks behave similarly to identity mapping. Specifically, we discussed the cases in which it is desirable to decay the $\gamma$ parameters of BN and the cases in which it is not, presenting four guidelines for their management. Experiments showed that applying $L_2$ regularization to applicable $\gamma$ increased 1% to 4% classification accuracy, whereas applying $L_2$ regularization to inapplicable $\gamma$ decreased 1% to 3% classification accuracy, which validates the proposed four guidelines. The proposed guidelines were further verified in experiments on several variants of residual networks and transformer models.
APPENDIX

A VARIANCE ANALYSIS IN TRANSFORMER

In this section, we provide variance analysis for the transformer. The transformer block is composed of self-attention blocks and MLP blocks. We first visit the MLP block (Figure 9(b)). For simplicity, we approximate GeLU to ReLU. Here, we denote the residual branch of the MLP block as $\varnothing_{l+1}(x_{l+1}) = W_{2,l+1}ReLU(W_{1,l+1}LN_{2,l+1}(x_{l+1}))$. Similar to the main text, we have that

$$Var[\varnothing_{l+1,i}(x_{l+1})] = 2 \cdot 2 \cdot Var[ReLU(LN_{2,l+1,i}(x_{l+1}))]$$

(25)

$$= 2\gamma_{2,l+1,i}^2.$$  

(26)

Thus, the MLP block accumulates the variance of the residual branch:

$$Var[x_{l+2,i}] = Var[x_{l+1,i}] + 2\gamma_{2,l+1,i}^2.$$  

(27)

Fig. 9. Transformer block: (a) self-attention block and (b) MLP block. Similar to residual networks, the two blocks have a skip connection.
We now visit the self-attention block (Figure 9(a)). Borrowing the notation from the vision transformer, we have that

\[
[q, k, v] = LN_{l,1}(x_l) W_{qkv, l},
\]

\[
A = \text{softmax}(qk^\top / \sqrt{D_h}),
\]

\[
f_l(x_l) = W_{\text{proj}, l} A v.
\]

For simplicity, we assume that the softmax function provides a one-hot encoded vector from the max-indicator function, which is independent of the input scale. Thus, the scale that the first LN controls does not affect \(A\) but affects \(v\). Then, the scale of \(f_l(x_l)\) is determined by \(W_{\text{proj}, l}, W_{qkv, l}\), and \(\gamma_{1,l}\):

\[
\text{Var}[f_l, i(x_l)] = 4\gamma_{1,l, i}^2.
\]

Thus,

\[
\text{Var}[x_{l+1, i}] = \text{Var}[x_{l, i}] + 4\gamma_{1,l, i}^2.
\]

Equations (27) and (32) indicate that both the MLP block and the self-attention block play a role in variance accumulation. Thus, we should apply \(L_2\) regularization to \(\gamma_1\) and \(\gamma_2\) in LNs.

REFERENCES

[1] Lei Jimmy Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. 2016. Layer normalization. CoRR abs/1607.06450 (2016).

[2] Lukas Bossard, Matthieu Guillaumin, and Luc Van Gool. 2014. Food-101 — Mining discriminative components with random forests. In ECCV (6), Vol. 8694. 446–461. https://data.vision.ee.ethz.ch/cvl/datasets_extra/food-101/

[3] Andrew Brock, Soham De, and Samuel L. Smith. 2021. Characterizing signal propagation to close the performance gap in unnormalized ResNets. In ICLR.

[4] Mauro Cettolo, Jan Niehues, Sebastian Stüker, Luisa Bentivogli, and Marcello Federico. 2014. Report on the 11th IWSLT evaluation campaign. In Proceedings of the 11th International Workshop on Spoken Language Translation: Evaluation Campaign. 2–17. https://workshop2014.iwslt.org/

[5] Soham De and Samuel L. Smith. 2020. Batch normalization biases residual blocks towards the identity function in deep networks. In NeurIPS.

[6] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. ImageNet: A large-scale hierarchical image database. In CVPR. 248–255. https://www.image-net.org/

[7] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In NAACL-HLT (1). 4171–4186.

[8] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. 2021. An image is worth 16x16 words: Transformers for image recognition at scale. In ICLR.

[9] Priya Goyal, Piotr Dollár, Ross B. Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. 2017. Accurate, large minibatch SGD: Training ImageNet in 1 hour. CoRR abs/1706.02677 (2017).

[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification. In ICCV. 1026–1034.

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In CVPR. 770–778.

[12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Identity mappings in deep residual networks. In ECCV (4). Vol. 9908. 630–645.

[13] Tong He, Zhi Zhang, Hang Zhang, Zhongyue Zhang, Junyuan Xie, and Mu Li. 2019. Bag of tricks for image classification with convolutional neural networks. In CVPR. 558–567.

[14] Elad Hoffer, Ron Banner, Itay Golan, and Daniel Soudry. 2018. Norm matters: Efficient and accurate normalization schemes in deep networks. In NeurIPS. 2164–2174.

[15] Grant Van Horn, Steve Branson, Ryan Farrell, Scott Haber, Jessie Barry, Panos Ipeirotis, Pietro Perona, and Serge J. Belongie. 2015. Building a bird recognition app and large scale dataset with citizen scientists: The fine print in fine-grained dataset collection. In CVPR. 595–604. https://dl.allaboutbirds.org/nabirds
[16] Sergey Ioffe and Christian Szegedy. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In ICML, Vol. 37. 448–456.

[17] Xianyan Jia, Shutao Song, Wei He, Yangzihao Wang, Haidong Rong, Feihu Zhou, Liqiang Xie, Zhenyu Guo, Yuanzhou Yang, Liwei Yu, Tiegang Chen, Guangxiao Hu, Shaohuai Shi, and Xiaowen Chu. 2018. Highly scalable deep learning training system with mixed-precision: Training ImageNet in four minutes. CoRR abs/1807.11205 (2018).

[18] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. 2012. ImageNet classification with deep convolutional neural networks. In NIPS. 1106–1114.

[19] Xingjian Li, Haoyi Xiong, Zeyu Chen, Jun Huan, Cheng-Zhong Xu, and Dejing Dou. 2021. “In-network ensemble”: Deep ensemble learning with diversified knowledge distillation. ACM Trans. Intell. Syst. Technol. 12, 5 (2021), 63:1–63:19.

[20] Dongyu Liu, Weiwei Cui, Kai Jin, Yuxiao Guo, and Huamin Qu. 2019. DeepTracker: Visualizing the training process of convolutional neural networks. ACM Trans. Intell. Syst. Technol. 10, 1 (2019), 6:1–6:25.

[21] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: A method for automatic evaluation of machine translation. In ACL. 311–318.

[22] Omkar M. Parkhi, Andrea Vedaldi, Andrew Zisserman, and C. V. Jawahar. 2012. Cats and dogs. In CVPR. 3498–3505. https://www.robots.ox.ac.uk/~vgg/data/pets/

[23] Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, and Martin A. Riedmiller. 2015. Striving for simplicity: The all convolutional net. In ICLR (Workshop).

[24] Cecilia Summers and Michael J. Dinneen. 2020. Four things everyone should know to improve batch normalization. In ICLR.

[25] Qing Tian, Shun Peng, and Tinghui Ma. 2023. Source-free unsupervised domain adaptation with trusted pseudo samples. ACM Trans. Intell. Syst. Technol. 14, 2 (2023), 30:1–30:17.

[26] Twan van Laarhoven. 2017. L2 regularization versus batch and weight normalization. CoRR abs/1706.05350 (2017).

[27] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In NIPS. 5998–6008.

[28] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2019. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In ICLR. https://gluebenchmark.com/

[29] Jindong Wang, Yiqiang Chen, Wenjie Feng, Han Yu, Meiyu Huang, and Qiang Yang. 2020. Transfer learning with dynamic distribution adaptation. ACM Trans. Intell. Syst. Technol. 11, 1 (2020), 6:1–6:25.

[30] Yuxin Wu and Kaiming He. 2020. Group normalization. Int. J. Comput. Vis. 128, 3 (2020), 742–755.

[31] Saining Xie, Ross B. Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. 2017. Aggregated residual transformations for deep neural networks. ACM Trans. Intell. Syst. Technol. 14, 2 (2023), 22:1–22:25.

[32] Junjie Yan, Ruosi Wan, Xiangyu Zhang, Wei Zhang, Yichen Wei, and Jian Sun. 2020. Towards stabilizing batch statistics in backward propagation of batch normalization. In ICLR.

[33] Sergey Zagoruyko and Nikos Komodakis. 2016. Wide residual networks. In BMVC.

[34] Matthew D. Zeiler and Rob Fergus. 2014. Visualizing and understanding convolutional networks. In ECCV (1), Vol. 8689. 818–833.

[35] Matthew D. Zeiler and Rob Fergus. 2014. Visualizing and understanding convolutional networks. In ECCV (1), Vol. 8689. 818–833.

[36] Guodong Zhang, Chaoqi Wang, Bowen Xu, and Roger B. Grosse. 2019. Three mechanisms of weight decay regularization. In ICLR.

[37] Hongyi Zhang, Yann N. Dauphin, and Tengyu Ma. 2019. Fixup initialization: Residual learning without normalization. In ICLR.

Received 19 July 2023; revised 25 January 2024; accepted 29 January 2024