Improving Generalization of Batch Whitening by Convolutional Unit Optimization

Yooshin Cho Hanbyel Cho Youngsoo Kim Junmo Kim
School of Electrical Engineering, KAIST, South Korea
{choys95, tlrl4658, ysoo.kim, junmo.kim}@kaist.ac.kr

Abstract

Batch Whitening is a technique that accelerates and stabilizes training by transforming input features to have a zero mean (Centering) and a unit variance (Scaling), and by removing linear correlation between channels (Decorrelation). In commonly used structures, which are empirically optimized with Batch Normalization, the normalization layer appears between convolution and activation function. Following Batch Whitening studies have employed the same structure without further analysis; even Batch Whitening was analyzed on the premise that the input of a linear layer is whitened. To bridge the gap, we propose a new Convolutional Unit that in line with the theory, and our method generally improves the performance of Batch Whitening. Moreover, we show the inefficacy of the original Convolutional Unit by investigating rank and correlation of features. As our method is employable off-the-shelf whitening modules, we use Iterative Normalization (IterNorm), the state-of-the-art whitening module, and obtain significantly improved performance on five image classification datasets: CIFAR-10, CIFAR-100, CUB-200-2011, Stanford Dogs, and ImageNet. Notably, we verify that our method improves stability and performance of whitening when using large learning rate, group size, and iteration number. Code is available at https://github.com/YooshinCho/pytorch_ConvUnitOptimization.

1. Introduction

Batch Normalization (BN) [11] is considered as a key component of deep neural networks. It significantly stabilizes and accelerates training by normalizing input features to have a zero mean (Centering) and a unit variance (Scaling), which is followed by a linear transform. Numerous follow-up studies have been proposed following the success of BN, and Batch Whitening [8, 20, 10, 9] studies were proposed that not only centering and scaling, but also removing linear correlation between input channels (Decorrelation) to improve BN. Although it is well-known that decorrelation increases network capacity, and stabilizes and accelerates training, it is not used in practice due to its ambiguity and complexity. Unlike centering and scaling, decorrelation is not an one-to-one mapping transform, and obtaining the inverse square root of the covariance matrix is computationally complex.

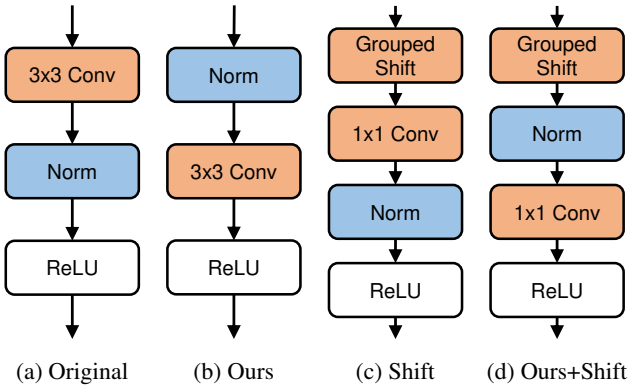


Figure 1: Illustration of the Convolutional Units. (a) is the Original Convolutional Unit. (b) is our modified Convolutional Unit that whitens the input of convolution. (c) is the Original Convolutional Unit that employs Grouped Shift. (d) is our modified Convolutional Unit that employs shift to directly whiten the input of point-wise convolution

Naturally, previous whitening studies have focused on introducing computationally efficient whitening methods, and investigating the reasons for the varying performance of each whitening method. Specifically, whitening methods [8, 20, 10] based on ZCA [2, 13], Cholesky Decomposition [5], and Newton's iterations [3] were proposed, and Iterative Normalization (IterNorm) [10], based on Newton's iterations, archeived the state-of-the-art performance owing to its small stochasticity. The superiority of IterNorm was investigated in [9] by comparing it with other whitening modules, but the performance gain is yet to be fully explored. To investigate the efficacy of IterNorm, we train ResNet [7] and Wide-Residual Network (WRN) [26]

on CIFAR-10, CIFAR-100 [15], and ImageNet [18]. We apply IterNorm by replacing all BN of ResNet and WRN with IterNorm. From the results shown in Table 1, we can observe that the performance of IterNorm is not satisfactory on CIFAR-100, despite the correlation of features being successfully removed.

To identify the reasons for the poor results, we revisit the theory of Batch Whitening [16]. We identify the gap between the theory and practice in terms of block design, and assume that the inefficacy of IterNorm can be attributed to the way in which whitening modules are used. Mechanisms of Batch Whitening were analyzed on the premise that the input of the linear layer is whitened; however, in commonly used block design, normalization layer is followed by a linear transform and the activation function before convolution as illustrated in Figure 1a. In this paper, we call these specific order of the layers as "Convolutional Unit". This Convolutional Unit was empirically optimized with BN without any analysis, but following Batch Whitening studies [8, 20, 10] have employed the same Convolutional Unit. Thus, irrespective of the efficacy of the whitening modules, the input of convolution is not centered, scaled, and decorrelated. Moreover, there are differences between spatial convolution and the linear layer considered in theory. Spatial convolution can be divided into shift operation [24, 12, 4] and point-wise convolution; thus, there is spatial misalignment between the input of spatial convolution and point-wise convolution. We call the gap as *input misalignment* in this paper. It means the whitening process is affected by the spatial shift operation, even if we directly perform whitening at the input of spatial convolution.

In this paper, we highlight three structural problems that contributes to the gap between the theory and practice: linear transform, position of whitening module, and *input misalignment*. To bridge the gap one step at a time, we modify the Convolutional Unit as illustrated in Figure 1b. Then, we empirically analyze the original and our Convolutional Unit. We employ IterNorm to empirically confirm the efficacy of whitening module is increased when used with our Convolutional Unit. Series of ablation studies show that the original Convolutional Unit is well-optimized with BN, but not optimized with whitening modules. As we expected, IterNorm outperforms when using with our Convolutional Unit, and linear transform makes training unstable and overfit. To support superiority of our method, we also investigate the rank and correlation of features. We empirically confirm that correlation is increased by about five times due to linear transform and activation function in practice. Also, we verify that input feature of normalization layer is not full rank when using the original Convolutional Unit. It makes decorrelated output extremely unstable due to noisy channels, and performance degenerate. By contrast, we observe that the input feature of normalization layer is full rank when using our Convolution Unit.

For further improvements, we close the gap, *input misalignment*, which is caused by spatial shift of convolution. To directly perform whitening at the input of point-wise convolution, we divide spatial convolution into Grouped Shift [24] and point-wise convolution, and place whitening modules between them as illustrated at Figure 1d. With modifications, we get much better performance than BN and IterNorm with original Convolutional Unit on CIFAR-10, CIFAR-100, CUB-200-2011 [23], Stanford Dogs [14]. To the best of our knowledge, this is the first paper that shows applicability of whitening modules in transfer learning, which we demonstrate on CUB-200-2011 and Stanford Dogs. Furthermore, we empirically confirm that our method enhances training stability of whitening modules. Our method shows significantly improved results at larger learning rates when the performance of BN, IterNorm using the original Convolution Unit decreases. Also, we compare stability of IterNorm with the original Convolutional Unit and our Convolutional Unit as increasing the iteration number. IterNorm using our Convolutional Unit shows much more stable behavior and significantly better performance with a iteration number larger than 7. Finally, we additionally adopt DBN [8], and demonstrate that our Convolutional Unit generally improves efficacy of whitening modules.

2. Related Works

2.1. Batch Standardization

Batch Standardization is a technique that only transforms features to have a zero mean (Centering) and a unit variance (Scaling) for computational efficiency. Since the success of Batch Normalization (BN) [11], a number of studies [22, 1, 25] have been proposed to improve the speed of learning by standardizing features. These studies focused on improving performance of micro-batch training and fixing a discrepancy between training and inference. They differed only in the target of normalization (e.g. batch, layer, instance, and group of channels), and they all performed the basic procedures of normalization: centering, scaling, and applying linear transformation. Therefore, they can be generally expressed using the following formula:

$$f(x_i) = \gamma_i \frac{x_i - \mu_i}{\sqrt{\sigma_i^2 + \epsilon}} + \beta_i, \tag{1}$$

where μ and σ are mean and standard deviation, respectively. The linear transformation is composed by scaling γ, and shifting β as described at Eq 1. It was intuitively, empirically introduced to prevent lose the original representation that can be lost during normalization. Also, the position of normalization was decided to be right after convolution based on the intuition that the output of convolution has "More-Gaussian" distribution without further analysis [11].

Naturally, the true reasons why BN helps optimization are still an active area of research, and Santurkar *et al*. [19] argues that BN helps training by smoothening loss landscape, not by the internal covariance shift.

2.2. Batch Whitening

It is generally known that performing centering, scaling, and removing the linear correlations between channel features on the batch helps efficient gradient descent. It conditions the Hessian of the weight and makes first-order gradient descent closer to the second-order gradient descent [16]. However, decorrelation is not adopted by BN due to its ambiguity and complexity. Thus, many studies [8, 20, 10] have proposed computationally efficient whitening methods and investigated the reason why whitening modules show different performances [9]. Specifically, Batch Whitening can be generally expressed as follows,

$$\hat{\boldsymbol{X}} = \boldsymbol{\Sigma}^{-\frac{1}{2}} \cdot (\boldsymbol{X} - \mu \cdot \boldsymbol{1}), \tag{2}$$

where $\boldsymbol{X}$ is the input, and $\boldsymbol{\Sigma}$ and μ are covariance matrix and mean of the input batch, respectively. Decorrelated Batch Normalization (DBN) [8] proposed ZCA whitening to get $\Sigma^{-\frac{1}{2}}$ and addressed the *stochastic axis swapping* issues. *Stochastic axis swapping* is caused by the ambiguity of rotation matrix of PCA whitening [6], but ZCA whitening fixes it by minimizing distortion caused by whitening. Siarohin *et al*. [20] adopted whitening based on Cholesky Decomposition [5] and introduced conditional Coloring transform to improve performance of GAN networks. Iterative Normalization (IterNorm) [10] proposed whitening based on Newton's iterations [3] and shows the state-of-the-art performance. Unlike previous methods, IterNorm approximates $\Sigma^{-\frac{1}{2}}$, and shows the smallest stochasticity.

While there have been many advances on Batch Whitening, there has been no analysis from the structural point of view. Thus, Batch Whitening studies followed the same Convolutional Unit that empirically optimized with BN. In this paper, we analyze the effects of block design, and maximize the efficacy of Batch Whitening by optimizing the Convolutional Unit.

3. Preliminary

In this section, we briefly describe Iterative Normalization and Shift Operation.

3.1. Iterative Normalization

Iterative Normalization (IterNorm) [10] is a state-of-the-art Batch Whitening module that employs Netwon's iterations [3] to obtain the inverse square root of covariance with controlling stochasticity. From the eq 2, we can say $\hat{\boldsymbol{X}}$ is a random variable which shows stochasticity caused by the batch sampling [21], and Huang *et al*. [9] suggested that the performance of the whitening module is related to its inherent stochasticity. Unlike previous whitening methods, the stochasticity of IterNorm can be controlled by two factors, iteration number and group size. IterNorm approximates the inverse square root of covariance by using an iterative process, and progressively stretches or squeezes the data along the axis to make eigenvalues 1. Also, IterNorm performs group-wise whitening to reduce stochasticity. Therefore, IterNorm performs whitening worse when iteration number and group size are small, but stochasticity is also decreased by ignoring the data along the axis with relatively small eigenvalues. It makes training stable and performance improve, but optimizing the trade-off between stochasticity and degree of whitening is difficult. In this study, we show that stability of IterNorm can be enhanced by Convolutional Unit optimization without loss of capability of whitening.

3.2. Shift Operation

The shift operation was originally introduced to reduce the number of parameters and FLOPs by replacing spatial convolution [24, 12, 4]. It was inspired by the fact that spatial convolution can be divided into shift operation and point-wise convolution. Basic spatial convolution can be expressed in the following formula:

$$\begin{aligned} \boldsymbol{Y} = \tilde{\boldsymbol{W}} \times \tilde{\boldsymbol{X}} &= \sum_k \boldsymbol{W}_{:,:,k} \cdot \boldsymbol{X}^k_{:,:} \\ &= \sum_k \boldsymbol{W}_{:,:,k} \cdot S_k(\boldsymbol{X}), \end{aligned} \tag{3}$$

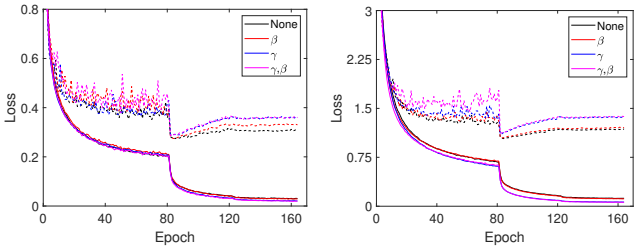
where $\tilde{\boldsymbol{W}} \in \mathbb{R}^{C_{out} \times kC_{in}}$ and $\tilde{\boldsymbol{X}} \in \mathbb{R}^{kC_{in} \times BHW}$ are weight matrix and concatenation of spatially shifted input matrix, respectively. k is kernel index and B is batch size. X^k is spatially shifted input corresponding to $W_{:,:,k}$, weight of specific kernel index k. $S_k(\cdot)$ is a shift operation of displacement of kernel index k. To directly perform whitening to the input of point-wise convolution, we employ Grouped Shift [24]. Grouped Shift is the operation that spatially shifts the features with fixed displacements. For consistency, we employ ShiftResNet and ShiftNet-A that proposed in [24] as baselines in Section 4.2. To effectively utilize Grouped Shift operation, channel sizes of intermediate features should be large. Similar to the bottleneck block used in ResNet, ShiftResNet and ShiftNet-A use the "Expansion Rate" ε to control the channel sizes. We conduct experiments with an expansion rate of 6 in the following sections.

4. Convolutional Unit Optimization

In this section, we discuss the way in which the Convolutional Unit can be optimized to match theory and to

Dataset / Arch.	BN	IterNorm
CIFAR-10 / ResNet20	**8.18±0.15**	**8.17±0.19**
CIFAR-10 / WRN-28-10	3.76±0.13	**3.68±0.16**
CIFAR-100 / ResNet56	**27.06±0.39**	27.53±0.35
CIFAR-100 / WRN-28-10	**18.71±0.13**	19.01±0.20
ImageNet / ResNet18	29.33	29.48 **28.86**

Table 1: Comparison of top-1 test errors (%) on ResNet and Wide-Residual Network (WRN) on CIFAR-10, CIFAR-100, and ImageNet. Except ImageNet, results are shown in the format of "mean±std". For ImageNet, the second row of IterNorm uses the "Full+DF" [10]. "Full+DF" means additonal IterNorm is plugged in after the last average pooling.



(a) CIFAR-10, ResNet20 (b) CIFAR-100, ResNet56

Figure 2: Illustration of train (solid lines), test (dashed lines) loss with respect to epochs. We train ResNet20 and ResNet56 using our Convolutional Unit with IterNorm on CIFAR-10 / 100. It shows the results of ablation studies by varying linear transform. Both (a) and (b) show that γ makes learning unstable and over-fit.

show generally improved performance in practice. We analyze the inefficacy of the linear transform and the position of whitening modules in Section 4.1, and address *input misalignment* and shift operation in Section 4.2. To empirically verify efficacy of our method, we employ IterNorm [10], the state-of-the-art whitening module. IterNorm is applied by replacing all BN [11] with IterNorm with a iteration number of 5 and full group size as suggested in [10]. For our modified Convolutional Unit, we do not use the linear transform in the experiments, unless otherwise stated.

4.1. Linear Transform and Position of Whitening

As shown in Table 1, we empirically verify the inefficacy of IterNorm, and assume that the sub-optimality of the Convolutional Unit degenerates the performance of the whitening modules. The linear transform following normalization was intuitively introduced to prevent possible loss of representation capability, and the position of normalization layer was decided based on the intuition that the output of convolution is more likely to have a symmetric, non-sparse distribution than the output of activation function without any analysis [11]. Based on the premise of theory that the input of the linear layer is whitened, we assume that whitening modules should be placed right before convolution without linear transform. Therefore, we modify the Convolutional Unit as illustrated in Figure 1b and apply to ResNet. We apply our Convolutional Unit by arranging the position of all normalization layers to be right before convolution.

To validate our assumption, we execute ablation studies by varying the Convolutional Units and the linear transform. We denote scaling and shifting operation of the linear transform as γ and β, respectively. We conduct experiments with ResNet [7] on two benchmark image classification datasets, CIFAR-10/100 [15]. Table 2 shows the results of the ablation studies. From the results, we validate that the original Convolutional Unit with linear transform is well-optimized with BN, but not optimized with IterNorm. Performance of IterNorm is generally increased as the Convolutional Unit is modified. Especially, IterNorm without any linear transform shows the largest performance gain when the Convolutional Unit is changed. Although, IterNorm with our Convolutional Unit still shows worse results than BN with original Convolutional Unit, it can be attributed to *input misalignment* which will be addressed in Section 4.2. Also, in Table 2, we can see that γ degrades the performance of IterNorm when using our Convolutional Unit. It has shown that removing correlation of input leads to better conditioning of the Hessian in updating the weights, and makes training closer to Newton's method [16]. Therefore, scaling factor γ is unnecessary when properly utilizing whitening modules. As shown in Figure 2, we can confirm γ makes learning unstable, and less compatible with the goal of conditioning Hessian via Batch Whitening.

To demonstrate the benefits of our Convolutional Unit, we investigate correlation of input feature of convolution and rank of input feature of normalization layer. We calculate the mean of the correlation ρ, and the mean of the rank divided by channel-size r, as follows:

$$\rho = \frac{1}{L-1}\sum_{l=1}^{L}\frac{2}{C^l(C^l-1)}\sum_{i=0}^{C^l}\sum_{j=i+1}^{C^l}(\tilde{\boldsymbol{X}}^l\tilde{\boldsymbol{X}}^{l\top})_{i,j}, \quad (4)$$

$$r = \frac{1}{L-1}\sum_{l=1}^{L}(\frac{rank(\boldsymbol{X}^l)}{C^l}), \quad (5)$$

where $\tilde{\boldsymbol{X}}^{\boldsymbol{l}}, \boldsymbol{X}^{\boldsymbol{l}} \in \mathbb{R}^{C^l \times BH^lW^l}$ are input matrix of lth convolution layer that normalized by l_2 norm of each channel and input matrix of lth normalization layer, respectively. L and C^l are number of convolution layers and channel size of $\boldsymbol{X}^{\boldsymbol{l}}$, respectively. We empirically show the severe effects of linear transform and activation function on decorrelation in Figure 3a. Correlation increases by almost five times due to

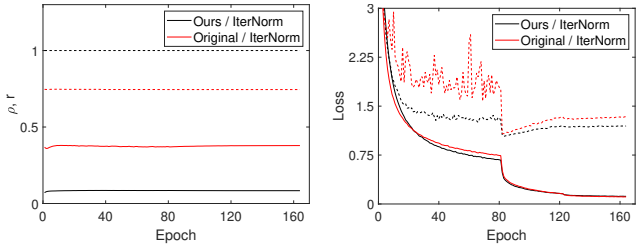
Methods	γ, β	γ	β	None
BN / Orignial	**8.18±0.15**	8.41± 0.22	8.40±0.21	8.88±0.20
BN / Ours	**8.43±0.19** (+0.25)	8.55±0.19 (+0.14)	8.68±0.14 (+0.28)	8.71±0.29 **(-0.17)**
IterNorm / Orignial	**8.17±0.19**	8.64±0.22	8.26±0.19	9.01±0.19
IterNorm / Ours	**8.17±0.17** (-0.0)	8.38±0.20 (-0.26)	**8.16±0.17** (-0.1)	8.29±0.20 **(-0.72)**

(a) CIFAR-10, ResNet20

Methods	γ, β	γ	β	None
BN / Orignial	**27.06±0.40**	27.49±0.39	27.48±0.33	28.31±0.24
BN / Ours	27.82±0.30 (+0.76)	27.82±0.25 (+0.33)	27.85±0.31 (+0.27)	**27.48±0.23 (-0.83)**
IterNorm / Orignial	**27.53±0.35**	28.33±0.24	**27.48±0.32**	28.35±0.37
IterNorm / Ours	**27.1±0.30** (-0.43)	27.64±0.30 (-0.69)	**27.12±0.25** (-0.36)	27.3±0.34 **(-1.05)**

(b) CIFAR-100, ResNet56

Table 2: Comparison of test errors (%) on ResNet20 and ResNet56 with CIFAR-10/100. All results are computed over 10 random seeds, and shown in the format of "mean±std". The values in parentheses indicate the test error difference between original and proposed unit.



(a) Correlation, Rank (b) Train, Test Loss

Figure 3: We train ResNet56 on CIFAR-100. The solid line in (a) is the average correlation of convolution input features, and the dashed line in (a) illustrates the average of the rank divided by channel-size of IterNorm input features. (b) is an illustration of train (solid lines) and test (dashed lines) losses of two Convolution Units with respect to epochs.

the linear transform and activation function. It indicates efficacy of Batch Whitening is severely affected by the linear transform and the activation function.

Rank of feature is related to the stochasticity of the whitening modules, and stochasticity of the whitening modules is considered as the key property of generalization capability of whitening modules [9]. If the input feature matrix of the whitening module is not full rank, the output will contain noisy channels caused by stretching the data along the axis with eigenvalues of 0. With the original Convolutional Unit, the input of whitening module, $\boldsymbol{X}^{\boldsymbol{l}}$ is not full rank when channel-size is increased by point-wise convolution; because, rank of feature is not increased by point-wise convolution, and output of convolution layer is directly connected to whitening module as illustrated in Figure 1a. Moreover, point-wise convolution is commonly used to increase channel-size in practice (e.g. bottleneck block of ResNet [7]). By contrast, in our Convolutional Unit, the output of convolution passes the activation function before the whitening module as illustrated in Figure 1b, and obtains an opportunity to increase its rank. We empirically confirm that the input of the whitening module is full rank in our Convolutional Unit and it is not in the original Convolutional Unit as illustrated in Figure 3a. Our method further enhances stability without controlling iteration number or group size as shown in Figure 3b. It indicates that our Convolutional Unit improves stability of IterNorm without loss of capability of whitening.

4.2. Shift Operation and Input Misalignment

Although Convolutional Unit modification generally improves the performance of IterNorm, BN with original Convolutional Unit still shows better or similar performance in the experiments, and β improves performance, despite affecting decorrelation and centering. We assume that the results are caused by the *input misalignment*. Spatial convolution can be generally expressed by eq 3. Spatial convolution spatially shifts input feature before passing to point-wise convolution, and it causes the gap between premise

Dataset	BN	IterNorm	γ, β	γ	β	None
CIFAR-10	6.96±0.12	7.03±0.15	6.89±0.26	7.01±0.17	6.88±0.06	**6.66±0.28**
CIFAR-100	28.37±0.38	28.38±0.32	27.75±0.12	27.83±0.33	27.33±0.17	**27.20±0.32**

Table 3: Comparisons of test errors (%) on ShiftResNet56 with CIFAR-10/100. All results are computed over 5 random seeds, and shown in the format of "mean±std". For simplicity, we denote IterNorm using our Convolutional Block by the sort of linear transform and omitting "Ours / IterNorm". We denote BN and IterNorm using the original Convolutional Unit with linear transform by "BN" and "IterNorm" and omitting "Original / γ, β", respectively.

and practice of whitening. We can express *input misalignment* by the following formula:

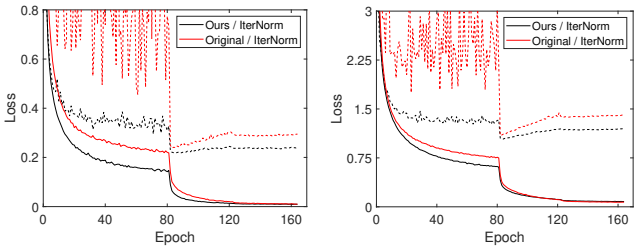
$$(\boldsymbol{X} \cdot \boldsymbol{X}^\top = \mathbf{I}) \nLeftrightarrow (\boldsymbol{S}(\boldsymbol{X}) \cdot \boldsymbol{S}(\boldsymbol{X})^\top = \mathbf{I}), \qquad (6)$$

where $\boldsymbol{S}()$ is channel-wise shift operation, and $\boldsymbol{X}$ is the input of spatial convolution. $(\boldsymbol{X} \cdot \boldsymbol{X}^\top = \mathbf{I})$ is what whitening modules in Figure 1b does, and $(\boldsymbol{S}(\boldsymbol{X}) \cdot \boldsymbol{S}(\boldsymbol{X})^\top = \mathbf{I})$ is what whitening modules in Figure 1d does. Whitening the input of spatial convolution does not imply whitening the input of point-wise convolution. In order to directly perform whitening at the input of point-wise convolution without modifying the whitening modules, we separate spatial convolution into Grouped Shift [24] and point-wise convolution, and place IterNorm between them. Subsequently, we propose the modified Convolutional Unit that employs Grouped Shift as illustrated in Figure 1d. For consistency, we employ ShiftResNet as the baseline, which was introduced in [24], instead of simply replacing the spatial convolution of ResNet with the shift operation and point-wise convolution. We conduct experiments with ShiftResNet as varying the Convolutional Unit and normalization modules on CIFAR-10 and CIFAR-100.

From the results shown in Table 3, we verify that the inefficacy of linear transform and original Convolutional Unit, and get general performance improvement as we expected. For simplicity, we do not compare performance by varying linear transform of BN and IterNorm with the original Convolution Unit, because we empirically verified that the original Convolutional Block is well-optimized with linear transform in Section 4.1. As we assumed, both γ and β degenerates performance, and IterNorm using our Convolutional Unit shows significantly better performance than both BN and IterNorm using the original Convolutional Unit. By comparing the performance improvement in Table 2, we can observe that performance of whitening module is severely affected by *input misalignment* that has not been considered before. From the loss graph in Figure 4, we demonstrate that our Convolutional Unit further improves stability and performance of training.

5. Experiment Results

In this section, we describe details of experiments. We additionally adopt DBN [8] to demonstrate applicability of our Convolutional Unit. DBN is applied with a group size of 64 for our Convolutional Unit, and 16 for original Convolutional Unit, because DBN is highly unstable with a group size larger than 16 with original Convolutional Unit. For simplicity, we call BN, DBN, and IterNorm using the original Convolutional Unit as BN, DBN, and IterNorm, respectively. In the following experiments, we do not use the linear transform for our Convolutional Unit, and we use the linear transform for original Convolutional Unit, unless otherwise stated.



(a) CIFAR-10 (b) CIFAR-100

Figure 4: Illustration of train (solid lines), test (dashed lines) loss with respect to epochs. We train ShiftResNet56 by varying the Convolutional Unit on CIFAR-10/100. Both (a) and (b) show that our Convolutional Unit without linear transform improves stability of training.

5.1. Image Classification

To investigate the effectiveness, we conduct experiments by varying the Convolutional Unit and normalization modules on CIFAR-10, CIFAR-100, CUB-200-2011, Stanford Dogs, and ImageNet [15, 23, 14, 18]. We demonstrate that our method also improve performance at large-scale dataset and transfer learning.

CIFAR-10/100. For CIFAR datasets, we train the network with 50k training images, and evaluate top-1 errors on 10k test images. Random horizontal flipping and translation by 4 pixels are adopted in our experiments. We use SGD with a batch size of 128 and apply momentum of 0.9 and weight decay of 0.0001. We set the initial learning rate to 0.1, then divide it by 10 at 81 and 122 epochs, and finish the training at 164 epochs.

Dataset / Methods	ShiftResNet 20			ShiftResNet 56		
	BN	DBN	IterNorm	BN	DBN	IterNorm
CIFAR-10 / Original	8.48±0.27	8.95±0.16	8.62±0.10	6.96±0.12	7.42±0.12	7.03±0.15
CIFAR-10 / Ours	-	**8.43±0.27**	8.45±0.17	-	6.84±0.20	**6.66±0.28**
CIFAR-10 / Original (lr: 1.0)	8.58±0.37	9.40±0.17	9.93±0.20	7.81±0.09	8.17±0.25	8.28±0.07
CIFAR-10 / Ours (lr: 1.0)	-	7.89±0.23	**7.47±0.33**	-	6.33±0.17	**6.04±0.07**
CIFAR-100 / Original	31.07±0.40	32.52±0.15	31.51±0.38	28.37±0.38	29.87±0.30	28.38±0.32
CIFAR-100 / Ours	-	30.42±0.22	**30.27±0.36**	-	27.85±0.12	**27.20±0.32**
CIFAR-100 / Original (lr: 1.0)	31.99±0.49	34.12±0.42	34.40±0.28	29.34±0.34	30.85±0.32	30.63±0.23
CIFAR-100 / Ours (lr: 1.0)	-	29.19±0.24	**29.09±0.28**	-	25.71±0.20	**25.51±0.18**

Table 4: Comparisons of test errors (%) on ShiftResNet 20/56 with CIFAR-10/100. To demonstrate applicability of our method, we additionally employ DBN and get improved performance. Also, to demonstrate enhanced stability of our Convolutional Unit, we train networks with initial learning rate of 1.0 and scheduling by cosine annealing. Best in bold, second-best underlined. All results are computed over 5 random seeds, and shown in the format of "mean±std".

From the results shown in Table 4, performance of DBN and IterNorm with our Convolutional Unit is better than BN, DBN, and IterNorm with original Convolutional Unit. We can observe that performance improvement by our Convolutional Unit increases as depth of network increases. IterNorm using our Convolutional Unit shows 0.37% and 1.18% performance improvement comparing with IterNorm on ShiftResNet56 with CIFAR-10/100, respectively. For DBN, we verify that performance improvement is even larger than IterNorm. As mentioned in [9, 8], DBN suffered from its inherent stochasticity, and our method effectively stabilizes whitening modules by increasing rank of the input matrix.

Methods	ShiftNet-A	ShiftNet-A-1.5
BN / Original	28.81 (9.73)	23.77 (7.12)
IterNorm / Original	28.25 (9.50)	23.87 (7.00)
IterNorm / Ours	**27.37 (8.97)**	**23.15 (6.86)**

Table 5: Comparisons of test errors (%) on ShiftNet-A with ImageNet. To shows performance at deeper and wider networks, we train ImageNet on ×1.5 deeper and wider ShiftNet-A. All results are shown in the format of "top-1 error(top-5 error)".

ImageNet. We train the network with 1.28M training images and evaluate top-1 and top-5 errors on a validation set with 50k images. We used standard augmentation with 224 pixels cropping. We use SGD with a batch size of 256 and apply a momentum of 0.9 and weight decay of 0.0001. We set the initial learning rate to 0.1, then divide it by 10 at every 30 epochs, and finish the training at 100 epochs.

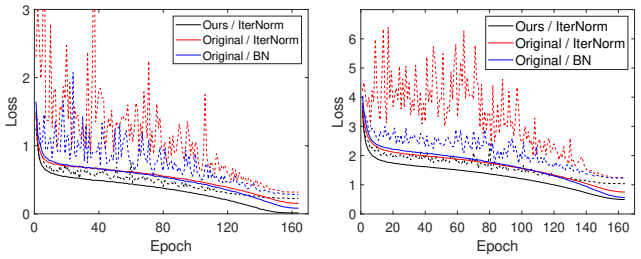
For consistency, we apply our method on ShiftNet-A that proposed to train ImageNet efficiently in [24]. To verify the effectiveness of our method on larger networks, we compare the performance on 1.5 times wider and deeper ShiftNet-A, which we denote as "ShiftNet-A-1.5". For fair comparison, we apply IterNorm using "Full+DF" that proposed in [10] to get the best performance. "Full+DF" mean additional IterNorm is applied after last global average pooling. Similar to results on CIFAR, we get the best performance among others. As shown in Table 5, our method obtain 1.44% and 0.62% performance improvement by comparing with BN on ShiftNet-A and ShiftNet-A-1.5, respectively.

Dataset	BN	IterNorm	Ours
CUB	17.64	16.53	**14.10**
Dogs	17.75	18.64	**16.95**

Table 6: Comparisons of top-1 test errors (%) on ShiftNet-A (pretrained with ImageNet) with CUB-200-2011 and Stanford Dogs. IterNorm with our block shows the best performance on both datasets.

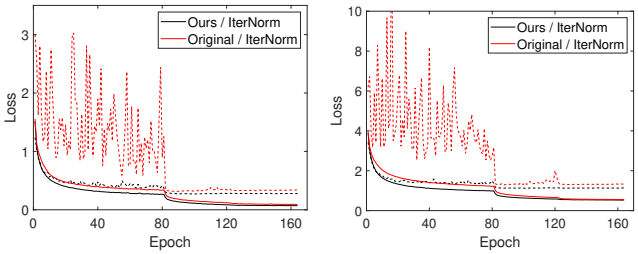
Transfer Learning. For CUB-200-2011 and Stanford Dogs, we train with the officially given 5,994 / 12,000 training images, and evaluate top-1 error on 5,794 / 8580 test images, respectively. We use random horizontal flipping with 448 pixels cropping. We use SGD with a batch size of 64 and apply momentum of 0.9 and weight decay of 0.0001. We set the initial learning rate to 0.01, then divide it by 10 at every 30 epochs (15 epochs for Stanford Dogs), and finish the training at 100 epochs (50 epochs for Stanford Dogs). We optimized the configuration by training ShiftNet with BN with a different learning rate in {0.1, 0.01, 0.001}. We employ ShiftNet-A pretrained with ImageNet.

As shown in Table 6, a significant performance improvement is achieved by our method with computation reduc-



(a) CIFAR-10, lr: 1.0 (b) CIFAR-100, lr: 1.0

Figure 5: Illustration of train (solid lines) and test (dashed lines) losses with respect to epochs. We train ShiftResNet20 on CIFAR-10/100 with the initial learning rate of 1.0. We can confirm that IterNorm using the original Convolutional Unit shows extremely unstable test loss. By contrast, IterNorm with our Convolutional Unit shows the best performance and stability among others.



(a) CIFAR-10, T: 8 (b) CIFAR-100, T: 8

Figure 6: Illustration of train (solid lines), test (dashed lines) loss with respect to epochs. We train ShiftResNet20 on CIFAR-10/100 with IterNorm with an iteration number of 8. We can observe that IterNorm using the our Convolutional Unit significantly improves stability when the iteration number is large.

tion by removing linear transform. We get 3.54% and 0.8% accuracy improvement comparing with BN on CUB-200-2011 and Stanford Dogs, respectively. To best of our knowledge, it is the first paper that shows applicability of whitening modules in transfer learning. Our results can potentially lead to the future whitening works in transfer learning.

5.2. Stability

Next, we domonstrate the superiority of our methods by showing enhanced stability. We train networks with 10 times larger learning rate as varying the Convolutional Unit and normalization layer. We apply cosine annealing [17] for a fair comparison without concerning scheduling tuning. As we reported in Table 4, IterNorm and DBN with our Convolutional Unit shows the best and second best performance among others. Notably, regardless of which whitening module is used, we observe that the performance of whitening module with our Convolutional Unit increases at a larger learning rate unlike with original Convolutional Unit. Comparing with the performance on the basic configuration, IterNorm with our method achieve 0.98% and 0.62% accuracy improvement with CIFAR-10 on ShiftResNet20 and ShiftResNet56, respectively. On CIFAR-100, IterNorm with our method achieve 1.18% and 1.69% accuracy improvement on ShiftResNet20 and ShiftResNet56, respectively.

We also show that our method enhances the stability of whitening modules with large group size and iteration number. Small group size or iteration number reduces stochasticity, but it also reduces whitening capability. IterNorm shows generally good performance at any group size; however, if iteration number is too large, learning become very unstable and shows poor results. This is caused by significant stochasticity due to noisy channels induced by small eigenvalues, and low rank of input feature is investigated in Section 4.1. Unlike previous studies, our Convolutional Unit fundamentally stabilizes IterNorm by making input features full rank as we confirmed in the Figure 3a. We verify this by comparing the loss graphs of IterNorm with the iteration number of 8. As we have shown in Figure 6, IterNorm using the original Convolutional Unit shows the extremely unstable test loss. By contrast, IterNorm using our Convolutional Unit shows stable test loss, and performance is also similar to that obtained by using the standard iteration number of 5. Also, as mentioned in [8], DBN with original Convolutional Unit with a group size larger than 16 shows extremely unstable behavior, but DBN with our Convolutional Unit is learnable with full group size.

6. Conclusion

In this paper, we investigate the way in which whitening modules, especially IterNorm, can be used. We optimize the efficacy of whitening by bridging the gap between practice and theory of Batch Whitening in terms of block design. The inefficacy of the original Convolutional Unit is empirically investigated, and results are in lined with the theory. We demonstrate the improved performance, stability, and transferability of our modified Convolutional Unit, and investigate the correlation and the rank of features to support our results. Our Convolutional Unit significantly stabilizes whitening modules by increasing the rank of features, and improves efficacy by properly choosing the target of whitening and removing the linear transform. Notably, we identify and solve the issue that we denote as *input misalignment*. Without modifying whitening module, we avoid the issue by employing Grouped Shift, and get a significant performance improvement on CIFAR-10/100, CUB-200-2011, Stanford Dogs, and ImageNet. Also, we demonstrate the significantly enhanced stability of our Convolutional Unit at large learning rate, iteration number, and group size.

References

[1] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization, 2016. 2

[2] Anthony J Bell and Terrence J Sejnowski. The "independent components" of natural scenes are edge filters. *Vision research*, 37(23):3327–3338, 1997. 1

[3] Dario A Bini, Nicholas J Higham, and Beatrice Meini. Algorithms for the matrix pth root. *Numerical Algorithms*, 39(4):349–378, 2005. 1, 3

[4] Weijie Chen, Di Xie, Yuan Zhang, and Shiliang Pu. All you need is a few shifts: Designing efficient convolutional neural networks for image classification. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7241–7250, 2019. 2, 3

[5] Dariusz Dereniowski and Marek Kubale. Cholesky factorization of matrices in parallel and ranking of graphs. In *International Conference on Parallel Processing and Applied Mathematics*, pages 985–992. Springer, 2003. 1, 3

[6] Guillaume Desjardins, Karen Simonyan, Razvan Pascanu, and Koray Kavukcuoglu. Natural neural networks. *arXiv preprint arXiv:1507.00210*, 2015. 3

[7] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 1, 4, 5

[8] Lei Huang, Dawei Yang, Bo Lang, and Jia Deng. Decorrelated batch normalization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 791–800, 2018. 1, 2, 3, 6, 7, 8

[9] Lei Huang, Lei Zhao, Yi Zhou, Fan Zhu, Li Liu, and Ling Shao. An investigation into the stochasticity of batch whitening. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6439–6448, 2020. 1, 3, 5, 7

[10] Lei Huang, Yi Zhou, Fan Zhu, Li Liu, and Ling Shao. Iterative normalization: Beyond standardization towards efficient whitening. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4874–4883, 2019. 1, 2, 3, 4, 7

[11] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. PMLR, 2015. 1, 2, 4

[12] Yunho Jeon and Junmo Kim. Constructing fast network through deconstruction of convolution. *arXiv preprint arXiv:1806.07370*, 2018. 2, 3

[13] Agnan Kessy, Alex Lewin, and Korbinian Strimmer. Optimal whitening and decorrelation. *The American Statistician*, 72(4):309–314, 2018. 1

[14] Aditya Khosla, Nityananda Jayadevaprakash, Bangpeng Yao, and Li Fei-Fei. Novel dataset for fine-grained image categorization. In *First Workshop on Fine-Grained Visual Categorization, IEEE Conference on Computer Vision and Pattern Recognition*, Colorado Springs, CO, June 2011. 2, 6

[15] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009. 2, 4, 6

[16] Yann A LeCun, Léon Bottou, Genevieve B Orr, and Klaus-Robert Müller. Efficient backprop. In *Neural networks: Tricks of the trade*, pages 9–48. Springer, 2012. 2, 3, 4

[17] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016. 8

[18] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015. 2, 6

[19] Shibani Santurkar, Dimitris Tsipras, Andrew Ilyas, and Aleksander Madry. How does batch normalization help optimization? *arXiv preprint arXiv:1805.11604*, 2018. 3

[20] Aliaksandr Siarohin, Enver Sangineto, and Nicu Sebe. Whitening and coloring batch transform for gans. *arXiv preprint arXiv:1806.00420*, 2018. 1, 2, 3

[21] Mattias Teye, Hossein Azizpour, and Kevin Smith. Bayesian uncertainty estimation for batch normalized deep networks. In *International Conference on Machine Learning*, pages 4907–4916. PMLR, 2018. 3

[22] Dmitry Ulyanov, Andrea Vedaldi, and Victor S. Lempitsky. Instance normalization: The missing ingredient for fast stylization. *CoRR*, abs/1607.08022, 2016. 2

[23] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie. The Caltech-UCSD Birds-200-2011 Dataset. Technical Report CNS-TR-2011-001, California Institute of Technology, 2011. 2, 6

[24] Bichen Wu, Alvin Wan, Xiangyu Yue, Peter Jin, Sicheng Zhao, Noah Golmant, Amir Gholaminejad, Joseph Gonzalez, and Kurt Keutzer. Shift: A zero flop, zero parameter alternative to spatial convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9127–9135, 2018. 2, 3, 6, 7

[25] Yuxin Wu and Kaiming He. Group normalization. In *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018. 2

[26] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016. 1