An Investigation into the Stochasticity of Batch Whitening

Lei Huang† Lei Zhao Yi Zhou† Fan Zhu† Li Liu† Ling Shao†
†Inception Institute of Artificial Intelligence (IIAI), Abu Dhabi, UAE
{lei.huang, yi.zhou, fan.zhu, li.liu, ling.shao}@inceptioniai.org bhneo@126.com

Abstract

Batch Normalization (BN) is extensively employed in various network architectures by performing standardization within mini-batches. A full understanding of the process has been a central target in the deep learning communities. Unlike existing works, which usually only analyze the standardization operation, this paper investigates the more general Batch Whitening (BW). Our work originates from the observation that while various whitening transformations equivalently improve the conditioning, they show significantly different behaviors in discriminative scenarios and training Generative Adversarial Networks (GANs). We attribute this phenomenon to the stochasticity that BW introduces. We quantitatively investigate the stochasticity of different whitening transformations and show that it correlates well with the optimization behaviors during training. We also investigate how stochasticity relates to the estimation of population statistics during inference. Based on our analysis, we provide a framework for designing and comparing BW algorithms in different scenarios. Our proposed BW algorithm improves the residual networks by a significant margin on ImageNet classification. Besides, we show that the stochasticity of BW can improve the GAN’s performance with, however, the sacrifice of the training stability.

1. Introduction

Normalization techniques have been extensively used for learning algorithms during data preprocessing [25, 23, 12]. It has been shown that centering, scaling and decorrelating the inputs speeds up the training [25]. Furthermore, whitening the input that combines all above operations, improves the conditioning of the Hessian, making the gradient descent updates similar to the Newton updates [25, 46, 16].

Batch Normalization (BN) [19] extends the idea of normalizing the input into the activations of intermediate layers of Deep Neural Networks (DNNs), and represents a milestone technique in the deep learning community [12, 43, 47]. BN standardizes the activations by executing centering and scaling within a mini-batch of data, facilitating the optimization [19, 22, 36] and generalization [4, 5].

(BW) further extends the scope of BN by removing the correlation of standardized activations [16]. It has been shown to improve the performance in discriminative scenarios [16] and Generative Adversarial Networks (GAN) [39].

Despite BW’s theoretical support in improving conditioning, there remains some intriguing observations relating to BW that are not well explored. Firstly, while various whitening transformations can equivalently improve the conditioning [20], they show significant differences in performance: 1) Principal Component Analysis (PCA) whitening hardly converges while Zero-phase Component Analysis (ZCA) whitening works well in discriminative scenarios [16]; 2) Cholesky Decomposition (CD) whitening achieves significantly better performance than ZCA in GAN training, while it has slightly worse performance in discriminative cases [39]. Secondly, while group based whitening—where features are divided into groups and whitening is performed within each one—is essential in discriminative scenarios...
full feature whitening has been shown to achieve better performance in training GANs [39].

This paper focuses on explaining the above observations of BW. We find that the stochasticity introduced by normalization over batch data (Figure 1) can be key to uncovering the intriguing phenomena of BW. We quantitatively investigate the magnitude of stochasticity in different whitening transformations, using the evaluation method called Stochastic Normalization Disturbance (SND) [17]. By doing so, we demonstrate that PCA whitening has significantly larger stochasticity, which is difficult to be controlled by either increasing batch size or using group based whitening. On the other side, ZCA whitening has the smallest stochasticity, while CD whitening has a moderate value, and more importantly, their stochasticity can be well controlled. This suggests ZCA whitening should have better optimization behaviors in training, while PCA whitening has problems in converging, due to the increased stochasticity which slows down the progress of the optimization [42, 17].

We also investigate the stochasticity during inference, which is caused by the estimation of population statistics averaged over the mini-batch statistics during training. We show that in terms of estimating the population statistics of the whitening matrix, it is more stable to use the mini-batch covariance matrix indirectly (We calculate the whitening matrix after training) than the mini-batch whitening matrix directly. We further provide an empirical investigation to understand the reasons behind this observation, and find that the stochastic sequences of the mini-batch whitening matrix have a large diversity than the covariance matrix.

Based on the above analyses, we provide a general framework for designing and comparing BW algorithms in different scenarios. We design new BW algorithm and apply them to the residual networks [12] for ImageNet dataset [8], significantly improving the performance over the original one. We further conduct thorough experiments on training GANs. We show that full feature whitening, when combined with coloring, can improve the final score of evaluation.

Another argument is that BN can adaptively adjust the learning rate [7, 15, 2] due to its scale invariance [19, 4]. This effect has been further discussed in combination with weight decay [49]. Other works have included an investigation into the signal propagation and gradient back-propagation [48]. Different from these approaches, our work focuses on analyzing the stochasticity of whitening over batch data.

The stochasticity introduced by normalization over batch data was first mentioned in the BN paper [19], and further explored in [3, 44] from the perspective of Bayesian optimization. This stochasticity results in differences between the training distribution (using mini-batch statistics) and the test distribution (using estimated population statistics) [18], which is believed to be the main cause of the small-batch-problem of BN [47]. To address this issue, a number of approaches have been proposed [47, 34, 29, 18, 45, 41]. Furthermore, it has been observed that BN also encounters difficulties in optimization during training [37, 17]. This phenomenon is explored by the stochastic analysis shown in [17]. Different from the above research which focuses on standardization, we analyze, for the first time, the stochasticity on batch whitening. We propose that analyzing whitening rather than standardization, has several advantages in understanding the behaviors of normalization over batch data: 1) There are an infinite number of whitening transformations and the main ones show significant differences as discussed in Section 1; 2) The extent of the whitening (stochasticity) can be well controlled by the batch and group size, which provides more information in designing experiments.

Our work is related to the previously proposed whitening methods regarding the activation of DNNs. One approach is to consider the whitening matrix as model parameters to be estimated using full data [9, 28]. This kind of whitening has also been exploited in image style transformation tasks [26, 38]. Another line of research is batch whitening, which is what this paper discusses. This approach treats the normalization as a function over a mini-batch input. The main works include PCA whitening, ZCA whitening [16] and its approximation iN [17], and CD whitening [39]. Pan et al. [31] propose switchable whitening to learn different batch/instance whitening/standardization operations in DNNs. However, they used only the ZCA whitening transformation. Our work aims to understand different whitening transformation behaviors based on stochastic analysis.

3. Stochasticity Analysis of Batch Whitening

Let \( X \in \mathbb{R}^{d \times m} \) be a data matrix denoting the mini-batch input of size \( m \) in a certain layer. For simplifying the discussion, we assume that the data is centered, by performing
where $\mu$ is the mean of $\mathbf{X}$, and $\mathbf{1}$ is a column vector of all ones. Whitening performs normalization over the mini-batch input as follows:

$$
\hat{\mathbf{X}} = \mathbf{G} \mathbf{X}.
$$

where $\mathbf{G}$ is the mini-batch whitening matrix that is derived from the corresponding covariance matrix $\Sigma = \frac{1}{m} \mathbf{X} \mathbf{X}^T$. The population statistics of the whitening matrix $\mathbf{G}$ used for inference, is usually calculated by running average over the mini-batches as follows:

$$
\hat{\mathbf{G}} = (1 - \lambda) \hat{\mathbf{G}} + \lambda \mathbf{G}.
$$

It is clear that both the whitened output $\hat{\mathbf{X}}$ (Eqn. 1) and the population statistics $\hat{\mathbf{G}}$ (Eqn. 2) can be viewed as stochastic variables, because they depend on the mini-batch inputs which are sampled over datasets. For illustration, we defer the analysis of the stochasticity to Sections 3.2 and 3.3, and first provide a review of the whitening transformations.

### 3.1. Whitening Transformations

There are an infinite number of possible whitening matrices, as shown in [20, 16], since any whitened data with a rotation is still whitened. This paper focuses on the whitening transformations based on PCA, ZCA and CD, since these three transformations have shown significant differences in performance when used in training DNNs [16, 39]. Note that BN [19] can be viewed as a special case of batch whitening, since it performs standardization without removing correlations, where $\hat{\mathbf{G}}_{BN} = (\text{diag}(\hat{\Sigma}))^{-1/2}$ with $\text{diag}(\hat{\Sigma})$ setting the off-diagonal elements of a matrix to zeros. To simplify the description, this paper regards BN as a (reduced) whitening transformation, unless otherwise stated.

**PCA Whitening** uses $\mathbf{G}_{PCA} = \Lambda^{-\frac{1}{2}} \mathbf{D}^T$, where $\Lambda = \text{diag}(\sigma_1, \ldots, \sigma_d)$ and $\mathbf{D} = [\mathbf{d}_1, \ldots, \mathbf{d}_d]$ are the eigenvalues and associated eigenvectors of $\hat{\Sigma}$, i.e. $\hat{\Sigma} = \mathbf{D} \Lambda \mathbf{D}^T$. Under this transformation, the variables are first rotated by the eigen-matrix (D) of the covariance, then scaled by the square root inverse of the eigenvalues ($\Lambda^{-\frac{1}{2}}$). PCA whitening over batch data suffers significant instability in training DNNs, and hardly converges, due to the so called Stochastic Axis Swapping (SAS), as explained in [16].

**ZCA Whitening** uses $\mathbf{G}_{ZCA} = \mathbf{D} \Lambda^{-\frac{1}{2}} \mathbf{D}^T$, where the PCA whitened input is rotated back by the corresponding rotation matrix $\mathbf{D}$. ZCA whitening works by stretching/squeezing the dimensions along the eigenvectors. It has been shown that ZCA whitening avoids the SAS and achieves better performance over standardization (used in BN) on discriminative classification tasks [16].

**CD Whitening** uses $\mathbf{G}_{CD} = L^{-1}$ where $\mathbf{L}$ is a lower triangular matrix from the Cholesky decomposition, with $\mathbf{L} \mathbf{L}^T = \hat{\Sigma}$. This kind of whitening works by recursively decorrelating the current dimension over the previous decorrelated ones, resulting in a triangular form of its whitening matrix. CD whitening has been shown to achieve the state-of-the-art performance in training GANs, while ZCA whitening has degenerated performance.

Figure 2. SND comparison of different batch whitening methods. We sample 60,000 examples from a Gaussian distribution as the training set. To calculate SND, we use $s = 200$ and $N = 20$. We show (a) the SND with respect to the dimensions ranging from $2^1$ to $2^9$, under a batch size of 1024; (b) the SND with respect to the batch size ranging from $2^7$ to $2^{12}$, under a dimension of 128.

The primary motivation of this paper is to investigate the following problem: while all whitening methods equivalently improve the conditioning of the layer input, why do they show significantly different behaviors in training DNNs? In the following sections, we provide a unified analysis and demonstrate that the key is the stochasticity introduced by normalization over batch data.

### 3.2. Stochasticity During Training

Given a sample $\mathbf{x} \in \mathbb{R}^d$ from a distribution $P_\mathbf{x}$, we take a sample set $\mathbf{X}^B = \{\mathbf{x}_1, \ldots, \mathbf{x}_B, \mathbf{x}_i \sim P_\mathbf{x}\}$ with a size of $B$. The whitened output $\hat{\mathbf{x}}$ can be formulated as $\hat{\mathbf{x}} = \mathbf{G}(\mathbf{X}^B; \mathbf{x})$. For a certain $\mathbf{x}$, $\mathbf{X}^B$ can be viewed as a random variable [3, 44, 17]. $\hat{\mathbf{x}}$ is thus another random variable showing stochasticity. Here, we investigate the stochasticity effects of different whitening transformations.

To provide a more intuitive illustration, we conduct a toy experiment to show how the normalized output of one sample changes when combined with a different sample set $\mathbf{X}^B$, by performing different whitening methods. Figure 1 (a), (b), (c) and (d) show the results when performing BN, PCA, ZCA and CD whitening, respectively. It is clear that the distribution of the PCA whitened output of one sample is very sparse, which means that $\hat{\mathbf{x}}$ has significant diversity. This suggests that PCA whitening provides large stochasticity. On the other side, the BN standardized output shows a tight Gaussian-style distribution, which suggests that BN has smaller stochasticity. Note that BN can not guarantee that the normalized output has an identity covariance matrix, while other whitening methods can. Similarly, we also observe that ZCA whitening provides reduced stochasticity, compared to CD. In fact, ZCA whitening has been shown to minimize the total squared distance between the original and whitened variables [20, 16]. We conjecture that such a property results in ZCA whitening having smaller stochasticity than CD.

#### 3.2.1 Quantitative Analysis

To provide a qualitative comparison, we exploit the evaluation called *Stochastic Normalization Disturbance* (SND), introduced in [17]. The empirical estimation of SND for
We notice that all whitening methods have an increased SND (Stochastic Axis Swapping) results in a large change of representation. Besides, ZCA has a smaller SND than CD, over all the dimensions, which is consistent with the observation shown in Figure 1. An interesting observation is that, in this case, BN works better than ZCA whitening. This is surprising since ZCA has improved conditioning over BN by removing the correlation [16], and it should theoretically have a better optimization behavior. However, the amplified stochasticity of ZCA whitening mitigates this advantage in optimization, thus resulting in a degenerated performance. Therefore, from an optimization perspective, we should control the extent of the stochasticity.

3.2.2 Controlling the Stochasticity by Groups

Huang et al. [16] proposed to use groups to control the extent of whitening. They argue that this method reduces the inaccuracy in estimating the full covariance matrix when the batch size is not sufficiently large. Here, we empirically show how group based whitening affects the SND, providing a good trade-off between introduced stochasticity and improved conditioning. This is essential for achieving a better optimization behavior.

We evaluate the SND of different whitening transformations by varying the group size ranging from 2 to 512, as shown in Figure 4 (a). We also display the spectrum of the covariance matrix of the whitened output (based on groups) in Figure 4 (b). We find that the group size effectively controls the SND of the ZCA/CD whitening. With decreasing group size, ZCA and CD show reduced stochasticity (Figure 4 (a)), while having also degenerated conditioning (Figure 4 (b)), since the output is only partially whitened. Besides, we observe that PCA whitening still has a large SND over all group sizes, and with no significant differences. This observation further corroborates the explanation of SAS given in [16], i.e., that the PCA whitening is extremely unstable.

We also show the SND of the approximate ZCA whitening method (called ItN [17]) in Figure 4 (a), which uses Newton’s iteration to approximately calculate the whitening matrix. We denote ‘ItNS’ as the ItN method with an iteration number of 5. An interesting observation is that ItN has smaller SND than BN, when using a large group size (e.g., 256) with a smaller iteration (e.g., T=5). This suggests that we can further combine group size and iteration number to control the stochasticity for ItN, providing an efficient and stable solution to approximate ZCA whitening [17].
The above observations furnish substantial insights into the application of whitening in neural networks (especially in scenarios that require decorating the activation in a certain layer), and we will further elaborate on this in Section 4.

We also use group-based ZCA/CD whitening methods on the four-layer MLP experiments. The results are shown in Figure 3 (b). We observe that ZCA and CD whitening, with a group size of 16 to control the stochasticity, achieve better training behaviors than BN.

### 3.3. Stochasticity During Inference

In the previous section we have shown that performing different whitening transformations, by introducing different magnitudes of stochasticity, results in significantly different training behaviors from an optimization perspective. It’s clear that such a stochasticity will also affect the final test performance during inference, because the population statistics $\hat{G}$ is estimated by running average (Eqn. 2) over the stochastic sequences $\{G_t\}_{t=1}^T$. This reveals why using $\Sigma$ is more stable for estimating $\hat{G}$ than using $\hat{G}$.

We further analyze how the diversity of stochastic sequences $\{M_t\}_{t=1}^T$ affect the estimation of $\hat{G}$, where $M \in \{\hat{G}, \Sigma\}$. Intuitively, if the stochastic sequence has high diversity, the effects of estimation will be worse. We view each element of $\hat{G}$ as an independent stochastic variable and calculate the standard deviation of each element during training, as follows:

$$
\delta(\hat{G}^{ij}_{M}) = \sqrt{\frac{1}{T} \sum_{t=1}^T (M_t^{ij} - \frac{1}{T} \sum_{t'=1}^T (M_t^{ij}))^2},
$$

where $\hat{G}^{ij}$ ($M_t^{ij}$) indicates the (i,j)-th element of $\hat{G}$ ($M_t$), respectively.

We further analyze how the diversity of stochastic sequences $\{M_t\}_{t=1}^T$ affect the estimation of $\hat{G}$, where $M \in \{\hat{G}, \Sigma\}$. Intuitively, if the stochastic sequence has high diversity, the effects of estimation will be worse. We view each element of $\hat{G}$ as an independent stochastic variable and calculate the standard deviation of each element during training, as follows:

$$
\delta(\hat{G}^{ij}_{M}) = \sqrt{\frac{1}{T} \sum_{t=1}^T (M_t^{ij} - \frac{1}{T} \sum_{t'=1}^T (M_t^{ij}))^2},
$$

where $\hat{G}^{ij}$ ($M_t^{ij}$) indicates the (i,j)-th element of $\hat{G}$ ($M_t$), respectively.

Furthermore, we calculate the normalized standard deviation of each element $\delta(\hat{G}^{ij}_{M})$, which, as defined in Eqn. 4, is calculated over $\hat{M}^{ij}_t = M_t^{ij} / \sqrt{\sum_{t'=1}^T (M_t^{ij})^2}$. Figure 6 (a) and (b) show the histogram of $\delta(\hat{G}^{ij}_{M})$ and $\delta(\hat{G}^{ij}_{M})$, respectively, when using ZCA whitening. We clearly find that $\hat{G}$ has a larger standard deviation on average, thus a larger diversity in general, compared to $\hat{G}$. This reveals why using $\Sigma$ is more stable for estimating $\hat{G}$ than using $G$.

### 4. Evaluation on Vision Tasks

Based on the previous analysis, we can design new BW algorithms, and construct more effective DNNs by using BW. We investigate this in classification and training GANs. The code to reproduce the experiments is available at https://github.com/huangleiBuaa/StochasticityBW.

#### 4.1. The Landscape of Batch Whitening Algorithms

We provide a general view of batch whitening algorithms in Algorithm 1 for vectorial inputs $X \in \mathbb{R}^{d \times m}$. Back-propagation is necessary to pass through the whitening transformation, and we provide the details in Appendix A for completeness. For feature map inputs $X_F \in \mathbb{R}^{h \times w \times d \times m}$, where $h$ and $w$ indicate the height and width, the whitening transformation is performed over the unrolled $X \in \mathbb{R}^{d \times (mhw)}$, since each spatial position of the feature map can be viewed as a sample [19].

Note that an extra step to recover the representation capacity of normalization is given in Line 11 of Algorithm 1, which is shown to be empirically effective in [19, 4, 16, 47].
We use the VGG networks tailored for CIFAR-10 datasets, from the perspective of optimization. We then show the performance on large-scale ImageNet classification [8], by applying the designed BW algorithm on residual networks [12].

4.2.2 Residual Network on ImageNet

We investigate the effectiveness of all kinds of whitening algorithms on residual networks for ImageNet classification with 1000 classes [8]. We use the given official 1.28M training images as a training set, and evaluate the top-1 accuracy on the validation set with 50k images.

Ablation Study on ResNet-18 We first execute an ablation study on the 18-layer residual network (ResNet-18) to explore multiple positions for replacing BN with BW. We consider three architectures: 1) ARC\textsubscript{A}: where we only replace the first BN module of ResNet-18 proposed in [16]; 2) ARC\textsubscript{B}: where we further plug-in the BW layer after the last average pooling (before the last linear layer) to learn the decorrelated feature representations based on ARC\textsubscript{A}, as proposed in [17]; 3) ARC\textsubscript{C}: where we also replace the \{2n, n = 1, 2, \ldots\}th BN modules (the \{3n\}th for ResNet-50) based on ARC\textsubscript{B}. We compare all the whitening transformations and estimation objects in Table 1. ‘ZCA’ and ‘ZCAST\textsubscript{2}’ denote the ZCA whitening that estimate the population statistics using $G$ and $\Sigma$, respectively. For ‘PCA’, ‘ZCA’ and ‘CD’, we use a group size in $[16, 64]$ and report the best performance from these two configurations.

We follow the same experimental setup as described in [12], except that we use one GPU and train over 100 epochs. We apply SGD with a mini-batch size of 256, momentum of 0.9 and weight decay of 0.0001. The initial learning rate is set to 0.1 and divided by 10 at 30, 60 and 90 epochs.

The results are shown in Table 2. For ARC\textsubscript{A}, we find...
Table 2. Comparison of validation accuracy (\%, single model and single-crop) on an 18-layer residual network for ImageNet.

| Method             | \(\text{ARC}_A\) | \(\text{ARC}_B\) | \(\text{ARC}_C\) |
|--------------------|-------------------|-------------------|-------------------|
| Baseline \[12\]    | 70.31             | –                 | –                 |
| PCA \[16\]        | 70.01 (\(\pm 0.30\)) | –                 | –                 |
| ZCA \[16\]        | 70.58 (\(\pm 0.27\)) | –                 | –                 |
| ZCA\(\Sigma\)     | 70.62 (\(\pm 0.31\)) | –                 | –                 |
| CD                 | 70.46 (\(\pm 0.15\)) | 70.80 (\(\pm 0.49\)) | 68.15 (\(\pm 2.16\)) |
| CD\(\Sigma\) \[39\] | 70.55 (\(\pm 0.24\)) | 70.89 (\(\pm 0.58\)) | 68.56 (\(\pm 1.75\)) |
| ItN \[17\]        | 70.62 (\(\pm 0.31\)) | 71.14 (\(\pm 0.83\)) | 71.26 (\(\pm 0.95\)) |
| ItN\(\Sigma\)     | 70.63 (\(\pm 0.32\)) | 71.33 (\(\pm 1.02\)) | 71.62 (\(\pm 1.31\)) |

Table 3. Results using ResNet-50 for ImageNet. We evaluate the top-1 validation accuracy (\%, single model and single-crop).

| Method         | Step decay | Cosine decay |
|----------------|------------|--------------|
| Baseline       | 76.20      | 76.62        |
| ItN\(\Sigma\)-ARC\(B\) | 77.18 (\(\pm 0.98\)) | 77.68 (\(\pm 1.06\)) |
| ItN\(\Sigma\)-ARC\(C\) | 77.28 (\(\pm 1.08\)) | 77.92 (\(\pm 1.30\)) |

Figure 8. Stability experiments on GAN with hinge loss \[30, 6\] for unconditional image generation. The box shows the quartiles while the whiskers extend to show the rest of the distribution (we limit the FID ranging in (20,100) for better representation, and we show the full range of FID in the Appendix C.2.1). Note that all methods use covariance matrix to estimate population statistics.
We report the best FID and the corresponding IS from all the configurations for different whitening methods.

![Table 4. Results on stability experiments shown in Section 4.3.1.](image)

We observe that ‘ZCA-64’ achieves better performance than ‘CD’. Nevertheless, it also makes the training unstable and sensitive to the hyperparameter. Besides, we observe that ‘ZCA-64’ achieves better performance than ‘CD’. This suggests that the advantage of ZCA whitening over CD in discriminative scenarios still exists when training GANS. We argue that controlling the magnitude of whitening (stochasticity) is also still important in training GANs. For example, for ItN, the approximation of ZCA whitening has nearly the same performance as CD (Table 4), and shows better stability (Figure 8) due to its effective control of the stochasticity, as explained in [17].

We also have obtained similar results when using the non-saturated loss [11], as shown in the Appendix C.2.1.

**Comparison of Estimation Object** Considering that the CD whitening in [39] uses the covariance matrix to indirectly estimate the whitening matrix when training a GAN, we also compare the whitening methods with different estimation objects. The results are shown in Figure 9. We observe that the methods using the covariance matrix as the estimation object, achieve consistently better FID scores and are generally more stable than using the whitening matrix.

**4.3.2 Validation on Larger Architectures**

Based on our previous analysis, we observe that ItN and ZCA whitening (with an appropriate group size) have similar performance to CD in training GANs, when using the covariance matrix as the estimation object. We further apply the ZCA whitening and ItN to the DCGAN and Resnet models, where CD whitening achieves nearly the state-of-the-art performance on CIFAR-10. We use the code provided in [39] and use the same setup as in [39]. We replace all the CD whitening in the models with ItN/ZCA whitening. The results are shown in Table 5. We observe that the CD whitening has no significant advantages over ItN/ZCA-G64. Ultimately, ItN achieves the best performances in terms of the FID evaluation, while CD whitening is the best in terms of IS. This suggests that there is still room to improve the performance of batch whitening in training GANs by further finely controlling the stochasticity.

**5. Conclusions**

In this paper, we provided a stochasticity analysis of batch whitening that thoroughly explains why different whitening transformations show significant differences in performance when training DNNs, despite their equivalent improvement of the conditioning. Our analysis provides insights for designing new normalization algorithms and constructing new network architectures. We believe that our analysis will open new avenues of research in better understanding normalization over batch data.

**Acknowledgement** We thank Anna Hennig and Ying Hu for their help with proofreading.
[39] Aliaksandr Siarohin, Enver Sangineto, and Nicu Sebe. Whitening and coloring batch transform for gans. In ICLR, 2019. 1, 2, 3, 5, 6, 7, 8, 11, 12, 13, 14, 15
[40] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In ICLR, 2015. 6
[41] Saurabh Singh and Abhinav Shrivastava. Evalnorm: Estimating batch normalization statistics for evaluation. In ICCV, 2019. 2
[42] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. J. Mach. Learn. Res., 15(1):1929–1958, Jan. 2014. 2
[43] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In CVPR, 2016. 1
[44] Mattias Teye, Hossein Azizpour, and Kevin Smith. Bayesian uncertainty estimation for batch normalized deep networks. In ICML, 2018. 2, 3
[45] Guangrun Wang, Jiefeng Peng, Ping Luo, Xinjiang Wang, and Liang Lin. Kalman normalization: Normalizing internal representations across network layers. In NIPS, 2018. 2
[46] Simon Wiesler and Hermann Ney. A convergence analysis of log-linear training. In NIPS, pages 657–665, 2011. 1
[47] Yuxin Wu and Kaiming He. Group normalization. In ECCV, 2018. 1, 2, 5
[48] Greg Yang, Jeffrey Pennington, Vinay Rao, Jascha Sohl-Dickstein, and Samuel S. Schoenholz. A mean field theory of batch normalization. In ICLR, 2019. 2
[49] Guodong Zhang, Chaoqi Wang, Bowen Xu, and Roger B. Grosse. Three mechanisms of weight decay regularization. In ICLR, 2019. 2
Algorithm I Back-propagation of batch whitening algorithms.

1: \textbf{Input:} mini-batch gradients \( \frac{\partial \mathcal{L}}{\partial \mathbf{X}} \in \mathbb{R}^{d \times m} \).
2: \textbf{Output:} \( \frac{\partial \mathcal{L}}{\partial \mathbf{X}} \in \mathbb{R}^{d \times m} \).
3: Calculate: \( \frac{\partial \mathcal{L}}{\partial \mathbf{X}} = \frac{\partial \mathcal{L}}{\partial \hat{\mathbf{Y}}} \frac{\partial \hat{\mathbf{Y}}}{\partial \mathbf{X}} \).
4: Calculate: \( \frac{\partial \mathcal{L}}{\partial \mathbf{Y}} = \frac{\partial \mathcal{L}}{\partial \mathbf{X}} \mathbf{X}^T \).
5: Calculate: \( \frac{\partial \mathcal{L}}{\partial \mathbf{A}} = \frac{\partial \mathcal{L}}{\partial \mathbf{Y}} \mathbf{D} \mathbf{D}^T \).
6: Calculate: \( \frac{\partial \mathcal{L}}{\partial \mathbf{D}} = \mathbf{K}^T \circ (\mathbf{D}^T \frac{\partial \mathcal{L}}{\partial \mathbf{D}}) + (\frac{\partial \mathcal{L}}{\partial \mathbf{A}})_{\text{diag}} \mathbf{D}^T \).

A. Back-propagation

Given the Batch Whitening (BW) algorithms described in Algorithm I of Section 4.1, we provide the corresponding back-propagation during training in Algorithm I. Note that it is easy to calculate \( \frac{\partial \mathcal{L}}{\partial \mathbf{X}} \) in Line 3 of Algorithm I, given the specific recovery operation shown in the paper. Here, we mainly elaborate on the nontrivial solution of back-propagation through the whitening transformation, i.e., how to calculate \( \frac{\partial \mathcal{L}}{\partial \mathbf{A}} \) given \( \frac{\partial \mathcal{L}}{\partial \mathbf{D}} \), shown in Line 5 of Algorithm I.

We consider the following whitening algorithms discussed in the paper: Principal Component Analysis (PCA) whitening [16], Zero-phase Component Analysis (ZCA) whitening [16], Cholesky Decomposition (CD) whitening [39] and ItN [17] that approximates ZCA whitening. The back-propagations of all whitening transformations are derived in their original papers [16, 39, 17]. Here, we provide the results for completeness.

A.1. PCA Whitening

PCA Whitening uses \( \mathbf{G}_{PCA} = \Lambda^{-\frac{1}{2}} \mathbf{D}^T \), where \( \Lambda = \text{diag}(\sigma_1, \ldots, \sigma_d) \) and \( \mathbf{D} = [\mathbf{d}_1, \ldots, \mathbf{d}_d] \) are the eigenvalues and associated eigenvectors of \( \Sigma \), i.e., \( \Sigma = \mathbf{D} \Lambda \mathbf{D}^T \).

The backward pass of PCA whitening is:

\[
\frac{\partial \mathcal{L}}{\partial \Lambda} = \left( \frac{\partial \mathcal{L}}{\partial \mathbf{G}_{PCA}} \right) \mathbf{D} \left( -\frac{1}{2} \Lambda^{-3/2} \right) \\
\frac{\partial \mathcal{L}}{\partial \mathbf{D}} = \left( \frac{\partial \mathcal{L}}{\partial \mathbf{G}_{PCA}} \right)^T \Lambda^{-1/2} \\
\frac{\partial \mathcal{L}}{\partial \Sigma} = \mathbf{D} \left( \mathbf{K}^T \circ (\mathbf{D}^T \frac{\partial \mathcal{L}}{\partial \mathbf{D}}) + (\frac{\partial \mathcal{L}}{\partial \Lambda})_{\text{diag}} \right) \mathbf{D}^T,
\]

where \( \mathbf{K} \in \mathbb{R}^{d \times d} \) is 0-diagonal with \( K_{ij} = \frac{1}{\sigma_i \sigma_j} \) for \( i \neq j \), the \( \circ \) operator is an element-wise matrix multiplication, and \( (\frac{\partial \mathcal{L}}{\partial \Lambda})_{\text{diag}} \) sets the off-diagonal elements of \( \frac{\partial \mathcal{L}}{\partial \Lambda} \) to zero.

A.2. ZCA Whitening

ZCA whitening uses \( \mathbf{G}_{ZCA} = \mathbf{D} \Lambda^{-\frac{1}{2}} \mathbf{D}^T \), where the PCA whitened input is rotated back by the corresponding rotation matrix \( \mathbf{D} \).

A.3. CD Whitening

CD whitening uses \( \mathbf{G}_{CD} = \mathbf{L} \), where \( \mathbf{L} \) is a lower triangular matrix from the Cholesky decomposition, with \( \mathbf{LL}^T = \Sigma \). The backward pass of CD whitening is:

\[
\frac{\partial \mathcal{L}}{\partial \mathbf{L}} = -\mathbf{G}_{CD}^T \frac{\partial \mathcal{L}}{\partial \mathbf{G}_{CD}} \mathbf{G}_{CD}^T \\
\frac{\partial \mathcal{L}}{\partial \Sigma} = \frac{1}{2} \mathbf{L}^{-T} (\mathbf{P} \circ \mathbf{L})^T \frac{\partial \mathcal{L}}{\partial \mathbf{L}} + (\mathbf{P} \circ \mathbf{L}^T \frac{\partial \mathcal{L}}{\partial \mathbf{L}})^T \mathbf{L}^{-1}.
\]

A.4. ItN Whitening

ItN, which approximates the ZCA whitening using Newton’s iteration, is shown in Algorithm II, where \( T \) is the iteration number and \( \text{tr}((\cdot)) \) denotes the trace of a matrix.
### Table A1. Results using ResNet-50 for ImageNet.

| Method            | Step decay | Cosine decay |
|-------------------|------------|--------------|
| Baseline          | 76.20      | 76.62        |
| ItN-ARC$_B$ [17]  | 77.07 (0.87) | 77.47 (0.85) |
| ItN-ARC$_C$ [17]  | 70.97 (0.77) | 77.43 (0.81) |
| ItN$_C$-ARC$_B$   | 77.18 (0.98) | 77.68 (1.06) |
| ItN$_C$-ARC$_C$   | **77.28 (1.08)** | **77.92 (1.30)** |

In Section 3.3 of the paper, we mentioned that the ItN$_C$ consistently provides a slight improvement over the original ItN proposed in [17] (using $G$ as an estimation object). Here, we provide the results, including the original ItN, in Table A1.

Given $\frac{\partial L}{\partial G_{tN}}$, the back-propagation is:

$$\frac{\partial L}{\partial P_T} = \frac{1}{\sqrt{\text{tr}(\Sigma)}} \frac{\partial L}{\partial G_{tN}}$$

$$\frac{\partial L}{\partial \Sigma_N} = -\frac{1}{2} \sum_{k=1}^{T} (P_{k-1}^2)^T \frac{\partial L}{\partial P_k}$$

$$\frac{\partial L}{\partial \Sigma} = \frac{1}{\text{tr}(\Sigma)} \frac{\partial L}{\partial \Sigma_N} - \frac{1}{\text{tr}(\Sigma)^2} \text{tr}(\frac{\partial L}{\partial \Sigma_N} \Sigma) I$$

$$-\frac{1}{2 \text{tr}(\Sigma)^3/2} \text{tr}((\frac{\partial L}{\partial G_{tN}})^T \Sigma) I. \quad (13)$$

Here, $\frac{\partial L}{\partial P_T}$ can be calculated by the following iterations:

$$\frac{\partial L}{\partial P_{k-1}} = \frac{3}{2} \frac{\partial L}{\partial P_k} - \frac{1}{2} \frac{\partial L}{\partial P_k} (P_{k-1}^2 (\sum_N)^T) - \frac{1}{2} \frac{\partial L}{\partial P_k} (P_{k-1}^2)^T \sum_N$$

$$-\frac{1}{2} (P_{k-1})^T \frac{\partial L}{\partial P_k} (P_{k-1}^2 (\sum_N)^T), \; k = T, \ldots, 1. \quad (14)$$

### B. Stochasticity During Inference

In Section 3.3 of the paper, we mentioned that we try other learning rates in the experiments to compare the estimation methods during inference. Here, we provide the results under a learning rate of 0.5, shown in Figure A1.

### C. More Details for Experiments

#### C.1. Classification on ImageNet

In Section 4.2.2 of the paper, we mentioned that ItN$_C$ consistently provides a slight improvement over the original ItN proposed in [17] (using $G$ as an estimation object). Here, we provide the results, including the original ItN, in Table A1. We use an iteration number of 5 for all ItN-related methods. For ItN, we use a group size of 64, as recommended in [17]. For ItN$_C$, we use a larger group size of 256, since itN$_C$ can still provide a good estimation of population statistics in high dimensions during inference, as shown in the paper.

#### C.2. Training GANs

##### C.2.1 Stability Experiments

**Experimental Setup** The DCGAN architecture in these experiments is shown in Table A7. Note that spectral normalization [30] is applied on the discriminator, following the setup shown in [39, 30]. We evaluate the FID between the 10k test data and 10k randomly generated examples. We use the Adam optimizer [21] and train for 100 epochs. We consider 15 hyper-parameter settings (Table A2) by varying the learning rate $\alpha$, first and second momentum ($\beta_1$, $\beta_2$) of Adam, and number of discriminator updates per generator update $n_{\text{dis}}$. The learning rate is linearly decreased until it reaches 0.

**Results** In Figures 8 and 9 of the paper, we only show the results with FID ranging in (20, 100) for better representation. Here, we provide the corresponding FIDs with full range in Figure A2 and A3.

We perform experiments on DCGAN with the non-saturating loss [11]. Figure A4 shows the results. Here, we obtain similar observations as for the DCGAN with the hinge loss, shown in Section 4.3.1 of the paper.
Figure A2. Stability experiments on GAN with hinge loss [30, 6] for unconditional image generation. We show the FIDs with full range.

Figure A3. Comparison of BW methods when using the Covariance Matrix (CM) and Whitening Matrix (WM) as the estimation object. We show the FIDs with full range.

We note that the computational cost of the Singular Value Decomposition (SVD) used in ZCA/PCA whitening and the Cholesky Decomposition highly depends on the specific deep learning platform (e.g., PyTorch [32] or Tensorflow [1]). We thus conduct experiments both on PyTorch (version number: 1.0.1) and Tensorflow (version number: 1.13), with CUDA (version number: 10.0.130). All implementations of BW are based on the API provided by PyTorch and Tensorflow. We hope that our experiments can provide good guidelines for applying BW in practice.

### C.3.1 Classification Models Using PyTorch

Table A4 shows the time costs of the VGG models for CIFAR-10 classification, described in Section 4.2.1 of the paper. Figure A5 and Table A5 show the time costs of the ResNet-18 and ResNet-50 described in Section 4.2.2 of the paper, respectively. Note that we run ResNet-18 on one GPU, while ResNet-50 on two GPUs.

Our main observations include: (1) ItN is more computationally efficient. (2) CD whitening is slower than ZCA whitening, which is contrary to the observation in [17] where CD whitening was significantly more efficient than ZCA. The main reason for this is that CD whitening requires the Cholesky decomposition and matrix inverse, which are slower than SVD under the PyTorch platform. We note that under Tensorflow, which was used to implement CD whitening in [39], CD whitening is more efficient than ZCA, as shown in Section C.3.2. (3) Group based whitening with small groups requires more training time than larger groups. The main reason is our unoptimized implementation where we whiten each group sequentially instead of in parallel, because the corresponding platform does not provide an easy way to use linear algebra library of CUDA in parallel, which is also mentioned in [16]. We believe BW would be more widely used if the group based methods could be easily implemented in parallel.

| Method       | Time (Group Size) |
|--------------|-------------------|
|              | 16    | 32    | 64    | 128   | 256   | 512   |
| BN           | 0.049 | 0.049 | 0.049 | 0.049 | 0.049 | 0.049 |
| ItN          | 0.624 | 0.338 | 0.192 | 0.13  | 0.11  | 0.106 |
| ZCA          | 2.245 | 1.536 | 0.906 | 0.498 | 0.505 | 0.512 |
| CD           | 9.8   | 4.818 | 2.391 | 1.29  | 0.907 | 0.793 |

Table A4. Time cost (s/iteration) on VGG for CIFAR-10 classification described in Section 4.2.1 of the paper. We compare the BW methods with different group sizes, and the results are averaged over 100 iterations.

| Method       | Time |
|--------------|------|
| Baseline (BN)| 0.418|
| ItN-G256-ARC_{B} | 0.464|
| ItN-G256-ARC_{C} | 0.546|
| ItN-G64-ARC_{B} | 0.522|
| ItN-G64-ARC_{C} | 0.662|

Table A5. Time cost (s/iteration) on ResNet-50 for ImageNet classification described in Section 4.2.2 of the paper. The results are averaged over 100 iterations.
Figure A4. Stability experiments on GAN with non-saturating loss [11] for unconditional image generation.

Figure A5. Time cost (s/iteration) on ResNet-18 for ImageNet classification described in Section 4.2.2 of the paper. We compare the BW methods with different group sizes under (a) ARC_B and (b) ARC_C, and the results are averaged over 100 iterations.

Figure A6. Time cost (ms/iteration) on GANs for stability experiments described in Section 4.3.1 of the paper. We compare the BW methods with different group sizes running on (a) a GPU and (b) a CPU. We evaluate the time cost of the generator where BW is applied, and the results are averaged over 100 iterations.

C.3.2 GANs Using Tensorflow

Figure A6 shows the time costs of the GAN models described in Section 4.3.1 of the paper. We perform experiments with the BW operation running on both a GPU and a CPU, shown in Figure A6 (a) and A6 (b), respectively. We observe that ZCA whitening running on GPUs has a significantly more expensive computational cost than other methods, which is consistent with the observation in [39]. We find this can be alleviated when running ZCA whitening on CPUs.

Table A6. Time cost (s/iteration) on large GAN architecture described in Section 4.3.2 of the paper. We evaluate the time cost of the generator where BW is applied, and the results are averaged over 100 iterations.

| Method | DCGAN CPU | DCGAN GPU | Resnet CPU | Resnet GPU |
|--------|-----------|-----------|------------|------------|
| CD     | 0.023     | 0.019     | 0.036      | 0.038      |
| ZCA-64 | 0.018     | 0.544     | 0.036      | 0.894      |
| lRN    | 0.017     | 0.016     | 0.034      | 0.034      |

Table A6 shows the time cost of the GAN models described in Section 4.3.2 of the paper.
Table A7. DCGAN architectures in stability experiments.

(a) Generator

\[
\begin{align*}
& z \in \mathbb{R}^{128} \sim \mathcal{N}(0, I) \\
& \text{dense, } 4 \times 4 \times 256 \\
& 4 \times 4, \text{ stride=2 deconv. BN 256 ReLU} \\
& 4 \times 4, \text{ stride=2 deconv. BN 128 ReLU} \\
& 4 \times 4, \text{ stride=2 deconv. BN 64 ReLU} \\
& 3 \times 3, \text{ stride=1 conv. 3 Tanh}
\end{align*}
\]

(b) Discriminator

\[
\begin{align*}
& \text{RGB image } x \in \mathbb{R}^{32 \times 32 \times 3} \\
& 3 \times 3, \text{ stride=1 conv 32 lReLU} \\
& 3 \times 3, \text{ stride=2 conv 64 lReLU} \\
& 3 \times 3, \text{ stride=1 conv 64 lReLU} \\
& 3 \times 3, \text{ stride=2 conv 128 lReLU} \\
& 3 \times 3, \text{ stride=1 conv 128 lReLU} \\
& 3 \times 3, \text{ stride=2 conv 256 lReLU} \\
& 3 \times 3, \text{ stride=1 conv 256 lReLU} \\
& \text{dense } \rightarrow 1
\end{align*}
\]

Table A8. Large architecture of DCGAN.

(a) Generator

\[
\begin{align*}
& z \in \mathbb{R}^{128} \sim \mathcal{N}(0, I) \\
& \text{dense, } 4 \times 4 \times 512 \\
& 4 \times 4, \text{ stride=2 deconv. BN 512 ReLU} \\
& 4 \times 4, \text{ stride=2 deconv. BN 256 ReLU} \\
& 4 \times 4, \text{ stride=2 deconv. BN 128 ReLU} \\
& 3 \times 3, \text{ stride=1 conv. 3 Tanh}
\end{align*}
\]

(b) Discriminator

\[
\begin{align*}
& \text{RGB image } x \in \mathbb{R}^{32 \times 32 \times 3} \\
& 3 \times 3, \text{ stride=1 conv 64 lReLU} \\
& 4 \times 4, \text{ stride=2 conv 128 lReLU} \\
& 3 \times 3, \text{ stride=1 conv 128 lReLU} \\
& 4 \times 4, \text{ stride=2 conv 256 lReLU} \\
& 3 \times 3, \text{ stride=1 conv 256 lReLU} \\
& 4 \times 4, \text{ stride=2 conv 512 lReLU} \\
& 3 \times 3, \text{ stride=1 conv 512 lReLU} \\
& \text{dense } \rightarrow 1
\end{align*}
\]

Table A9. Large architecture of ResNet. The ResBlock follows [39].

(a) Generator

\[
\begin{align*}
& z \in \mathbb{R}^{128} \sim \mathcal{N}(0, I) \\
& \text{dense, } 4 \times 4 \times 256 \\
& \text{ResBlock up 256} \\
& \text{ResBlock up 256} \\
& \text{ResBlock up 256} \\
& \text{BN, ReLU, } 3 \times 3 \text{ conv. 3 Tanh}
\end{align*}
\]

(b) Discriminator

\[
\begin{align*}
& \text{RGB image } x \in \mathbb{R}^{32 \times 32 \times 3} \\
& \text{ResBlock down 128} \\
& \text{ResBlock down 128} \\
& \text{ResBlock 128} \\
& \text{ResBlock 128} \\
& \text{ReLU} \\
& \text{Global sum pooling} \\
& \text{dense } \rightarrow 1
\end{align*}
\]