Group Whitening: Balancing Learning Efficiency and Representational Capacity

Lei Huang[1,3] Yi Zhou[2] Li Liu[3] Fan Zhu[3] Ling Shao[3]
[1]SKLSDE, Institute of Artificial Intelligence, Beihang University, Beijing, China
[2]MOE Key Laboratory of Computer Network and Information Integration, Southeast University, China
[3]Inception Institute of Artificial Intelligence (IIAI), Abu Dhabi, UAE

Abstract

Batch normalization (BN) is an important technique commonly incorporated into deep learning models to perform standardization within mini-batches. The merits of BN in improving a model's learning efficiency can be further amplified by applying whitening, while its drawbacks in estimating population statistics for inference can be avoided through group normalization (GN). This paper proposes group whitening (GW), which exploits the advantages of the whitening operation and avoids the disadvantages of normalization within mini-batches. In addition, we analyze the constraints imposed on features by normalization, and show how the batch size (group number) affects the performance of batch (group) normalized networks, from the perspective of model's representational capacity. This analysis provides theoretical guidance for applying GW in practice. Finally, we apply the proposed GW to ResNet and ResNeXt architectures and conduct experiments on the ImageNet and COCO benchmarks. Results show that GW consistently improves the performance of different architectures, with absolute gains of $1.02\% \sim 1.49\%$ *in top-1 accuracy on ImageNet and* $1.82\% \sim 3.21\%$ *in bounding box AP on COCO.*

1. Introduction

Batch normalization (BN) [24] represents a milestone technique in deep learning [15, 52, 59], and has been extensively used in various network architectures [15, 52, 66, 51, 18]. BN standardizes the activations within a mini-batch of data, which improves the conditioning of optimization and accelerates training [24, 4, 46]. Further, the stochasticity of normalization introduced along the batch dimension is believed to benefit generalization [59, 48, 22]. However, this stochasticity also results in differences between the training distribution (using mini-batch statistics) and the test distribution (using estimated population statistics) [23], which is believed to be the main cause of BN's small-batch-size problem — BN's error increases rapidly as the batch size becomes smaller [59]. To address this issue, a number of approaches have been proposed [59, 43, 37, 23, 56, 50, 7]. One representative method is group normalization (GN), which divides the neurons into groups and then applies the standardization operation over the neurons of each group, for each sample, independently. GN provides a flexible solution to avoid normalization along the batch dimension, and benefits visual tasks limited to small-batch-size training [59, 29].

As a widely used operation in data pre-processing, whitening not only standardizes but also decorrelates the data [31], which further improves the conditioning of the optimization problem [31, 58, 12, 20]. A whitened input has also been shown to make the gradient descent updates similar to the Newton updates for linear models [31, 58, 20]. Motivated by this, Huang *et al.* [20] proposed batch whitening (BW) for deep models, which performs whitening on the activations of each layer within a mini-batch. BW has been shown to achieve better optimization efficiency and generalization than BN [20, 22, 41]. However, BW further amplifies the disadvantage of BN in estimating the population statistics, where the number of parameters to be estimated with BW is quadratic to the number of neurons/channels. Thus, BW requires a sufficiently large batch size to work well.

To exploit whitening's advantage in optimization, while avoiding its disadvantage in normalization along the batch dimension, this paper proposes group whitening (GW). GW divides the neurons of a sample into groups for standardization over the neurons in each group, and then decorrelates the groups. Unlike BW, GW has stable performance for a wide range of batch sizes, like GN, and thus can be applied to a variety of tasks. GW further improves the conditioning of optimization of GN with its whitening operation.

One important hyperparameter of GW is the group number. We observe that GW/GN has a significantly degenerated training performance when the group number is large, which is similar to the small-batch-size problem of BW/BN. We attribute this to the constraints on the output imposed by the normalization operation, which affect the model's representational capacity. As such, this paper defines the **constraint number** of normalization (as will be discussed in Section 4) to quantitatively measure the magnitude of the constraints provided by normalization methods. With the support of the constraint number, we analyze how the

batch size (group number) affects the model's representational capacity for batch (group) normalized networks. Our analysis also presents a new viewpoint for understanding the small-batch-size problem of BN.

We apply the proposed GW to two representative deep network architectures (ResNet [15] and ResNeXt [60]) for ImageNet classification [45] and COCO object detection and instance segmentation [35]. GW consistently improves the performance for both architectures, with absolute gains of 1.02% ~1.49% in top-1 accuracy for ImageNet and 1.82% ~3.21% in bounding box AP for COCO.

2. Preliminaries

For simplicity, we first consider the d-dimensional input vector $\mathbf{x}$, which will be generalized to a convolutional input in the subsequent section. Let $\mathbf{X} \in \mathbb{R}^{d\times m}$ be a data matrix denoting the mini-batch input of size m in a given layer.

Standardization. During training, batch normalization (BN) [24] standardizes the layer input within a mini-batch, for each neuron, as[1]:

$$\widehat{\mathbf{X}} = \phi_{BN}(\mathbf{X}) = \Lambda_d^{-\frac{1}{2}}(\mathbf{X} - \mu_d \mathbf{1}^T). \tag{1}$$

Here, $\mu_d = \frac{1}{m}\mathbf{X}\mathbf{1}$ and $\Lambda_d = \text{diag}(\sigma_1^2, \ldots, \sigma_d^2) + \epsilon\mathbf{I}$, where σ_i^2 is the variance over mini-batches for the i-th neuron, $\mathbf{1}$ is a column vector of all ones, and $\epsilon > 0$ is a small number to prevent numerical instability. During inference, the population statistics $\{\widehat{\Lambda}_d^{-\frac{1}{2}}, \hat{\mu}_d\}$ are required for deterministic inference, and they are usually calculated by running average over the training iterations, as follows:

$$\begin{cases} \hat{\mu}_d = (1-\lambda)\hat{\mu}_d + \lambda\mu_d, \\ \widehat{\Lambda}_d^{-\frac{1}{2}} = (1-\lambda)\widehat{\Lambda}_d^{-\frac{1}{2}} + \lambda\Lambda_d^{-\frac{1}{2}}. \end{cases} \tag{2}$$

Such an estimation process can limit the usage of BN in recurrent neural networks [30, 10], or harm the performance for small-batch-size training [23, 59].

To avoid the estimation of population statistics shown in Eqn. 2, Ba *et al.* proposed layer normalization (LN) [4] to standardize the layer input within the neurons for each training sample, as:

$$\widehat{\mathbf{X}} = \phi_{LN}(\mathbf{X}) = (\mathbf{X} - \mathbf{1}\mu_m^T)\Lambda_m^{-\frac{1}{2}}. \tag{3}$$

Here, $\mu_m = \frac{1}{d}\mathbf{X}^T\mathbf{1}$ and $\Lambda_m = \text{diag}(\sigma_1^2, \ldots, \sigma_m^2) + \epsilon\mathbf{I}$, where σ_i^2 is the variance over the neurons for the i-th sample. LN has the same formulation during training and inference, and is extensively used in natural language processing tasks [55, 65, 62].

Group normalization (GN) [59] further generalizes LN, dividing the neurons into groups and performing the standardization within the neurons of each group independently, for each sample. Specifically, defining the group division operation as $\Pi : \mathbb{R}^{d\times m} \mapsto \mathbb{R}^{c\times gm}$, where g is the group number and $d = gc$, GN can be represented as follows:

$$\widehat{\mathbf{X}} = \phi_{GN}(\mathbf{X}; g) = \Pi^{-1}(\phi_{LN}(\Pi(\mathbf{X}))), \tag{4}$$

where $\Pi^{-1} : \mathbb{R}^{c\times gm} \mapsto \mathbb{R}^{d\times m}$ is the inverse operation of Π. It is clear from Eqn. 4 that LN is a special case of GN with $g = 1$. By changing the group number g, GN is more flexible than LN, enabling it to achieve good performance on visual tasks limited to small-batch-size training (*e.g.*, object detection and segmentation [59]).

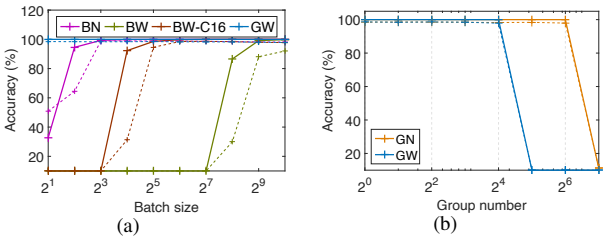


Figure 1. Effects of batch size (group number) for batch (group) normalized networks. We train a four-layer multilayer perceptron (MLP) with 256 neurons in each layer, for MNIST classification. We evaluate the training (thick 'plus' with solid line) and validation (thin 'plus' with dashed line) accuracies at the end of 50 training epochs. Note that 'BW-C16' indicates group-based BW with 16 neurons in each group. We vary the batch size and group number in (a) and (b), respectively. These results are obtained using a learning rate of 0.1, but we also obtain similar observations for other learning rates. Please see the *supplementary materials (SM)* for details.

Whitening. To exploit the advantage of whitening over standardization in improving the conditioning of optimization, Huang *et al.* proposed decorrelated BN [20], which performs zero-phase component analysis (ZCA) whitening to normalize the layer input within a mini-batch, as:

$$\phi_{ZCA}^W(\mathbf{X}) = \Sigma_d^{-\frac{1}{2}}(\mathbf{X} - \mu_d\mathbf{1}^T) = \mathbf{D}\Lambda^{-\frac{1}{2}}\mathbf{D}^T(\mathbf{X} - \mu_d\mathbf{1}^T), \tag{5}$$

where $\Lambda = \text{diag}(\tilde{\sigma}_1, \ldots, \tilde{\sigma}_d)$ and $\mathbf{D} = [\mathbf{d}_1, ..., \mathbf{d}_d]$ are the eigenvalues and associated eigenvectors of Σ, *i.e.* $\Sigma = \mathbf{D}\Lambda\mathbf{D}^T$, and $\Sigma = \frac{1}{m}(\mathbf{X} - \mu_d\mathbf{1}^T)(\mathbf{X} - \mu_d\mathbf{1}^T)^T + \epsilon\mathbf{I}$ is the covariance matrix of the centered input. One crucial problem in Eqn. 5 is the eigen-decomposition, which is computationally expensive on a GPU and numerically instable. To address this issue, iterative normalization ('ItN') [22] was proposed to approximate the ZCA whitening matrix $\Sigma_d^{-\frac{1}{2}}$ using Newton's iteration [5]:

$$\phi_{ItN}^W(\mathbf{X}) = \Sigma_d^{-\frac{1}{2}}(\mathbf{X} - \mu_d\mathbf{1}^T) = \frac{\mathbf{P}_T}{\sqrt{tr(\Sigma_d)}}(\mathbf{X} - \mu_d\mathbf{1}^T), \tag{6}$$

where $tr(\Sigma_d)$ indicates the trace of Σ_d and $\mathbf{P}_T$ is calculated iteratively as:

$$\begin{cases} \mathbf{P}_0 = \mathbf{I} \\ \mathbf{P}_k = \frac{1}{2}(3\mathbf{P}_{k-1} - \mathbf{P}_{k-1}^3\Sigma_d^N), \quad k = 1, 2, ..., T. \end{cases} \tag{7}$$

Here, $\Sigma_d^N = \frac{\Sigma_d}{tr(\Sigma_d)}$. Other BW methods also exist for calculating the whitening matrix [20, 49]; please refer to [27, 21] for more details.

[1]BN and other normalization methods discussed in this paper all use extra learnable scale and shift parameters [24]. We omit this for simplicity.

It is necessary for BW to estimate the population statistics of the whitening matrix $\widehat{\Sigma}_d^{-\frac{1}{2}}$ during inference, like BN. However, the number of independent parameters in $\widehat{\Sigma}_d^{-\frac{1}{2}}$ of BW is $\frac{d(d+1)}{2}$, while $\widehat{\Lambda}_d^{-\frac{1}{2}}$ of BN is d. This amplifies the difficulty in estimation and requires a sufficiently large batch size for BW to work well (Figure 1). Although group-based BW [20] — where neurons are divided into groups and BW is performed within each one — can relieve this issue, it is still sensitive to the batch size (Figure 1) due to its inherent drawback of normalizing along the batch dimension.

3. Group Whitening

We propose group whitening (GW). Given a sample $\mathbf{x} \in \mathbb{R}^d$, GW performs normalization as:

$$Group\ division: \mathbf{X}_G = \Pi(\mathbf{x}; g) \in \mathbb{R}^{g \times c}, \tag{8}$$

$$Whitening: \widehat{\mathbf{X}}_G = \Sigma_g^{-\frac{1}{2}}(\mathbf{X}_G - \mu_g \mathbf{1}^T), \tag{9}$$

$$Inverse\ group\ division: \hat{\mathbf{x}} = \Pi^{-1}(\widehat{\mathbf{X}}_G) \in \mathbb{R}^d, \tag{10}$$

where $\Pi : \mathbb{R}^d \mapsto \mathbb{R}^{g \times c}$ and its inverse transform $\Pi^{-1} : \mathbb{R}^{g \times c} \mapsto \mathbb{R}^d$. We can use different whitening operations [20, 49, 27] in Eqn. 9. Here, we use ZCA whitening (Eqn. 5) and its efficient approximation 'ItN' (Eqn 6), since they work well on discriminative tasks [20, 22, 41, 47, 64]. We provide the full algorithms (forward and backward passes) and PyTorch [42] implementations in the *SM*.

GW avoids normalization along the batch dimension, and it works stably across a wide range of batch sizes (Figure 1). GW also ensures that the normalized activation for each sample has the properties: $\widehat{\mathbf{X}}_G \mathbf{1} = \mathbf{0}$ and $\frac{1}{c}\widehat{\mathbf{X}}_G \widehat{\mathbf{X}}_G^T = \mathbf{I}$, which should improve the conditioning, like BW, and benefit training. We conduct several experiments to validate this, and the results in Figure 2 show that the group whitened output (by GW) has significantly better conditioning than the group standardized one (by GN), which is similar to normalization along the batch dimension [20]. Note that the condition number of BW is 1. We also find that GN/GW has better conditioning with increasing group number. Besides, we find that BN has better conditioning than GN/GW, which suggests that normalizing along the batch dimension is better for decorrelating the data than normalizing along the channel dimension.

Convolutional layer. For the convolutional input $\mathbf{X} \in \mathbb{R}^{d \times m \times H \times W}$, where H and W are the height and width of the feature maps, BN and BW consider each spatial position in a feature map as a sample [24] and normalize over the unrolled input $\mathbf{X} \in \mathbb{R}^{d \times mHW}$. In contrast, LN and GN view each spatial position in a feature map as a neuron [59] and normalize over the unrolled input $\mathbf{X} \in \mathbb{R}^{dHW \times m}$. Following GN, GW also views each spatial position as a neuron, *i.e.*, GW operations (Eqns. 8, 9 and 10) are performed for each sample with unrolled input $\mathbf{x} \in \mathbb{R}^{dHW}$.

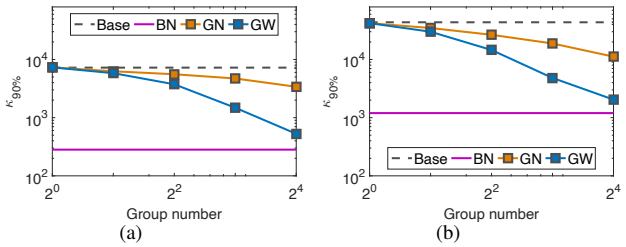


Figure 2. Conditioning analysis on the normalized output. We simulate the activations $\mathbf{X} = f(\mathbf{X}_0) \in \mathbb{R}^{256 \times 1024}$ using a network $f(\cdot)$, where $\mathbf{X}_0$ is sampled from a Gaussian distribution. We evaluate the more general condition number with respect to the percentage: $\kappa_p = \frac{\lambda_{max}}{\lambda_p}$, where λ_p is the pd-th eigenvalue (in descending order) and d is the total number of eigenvalues. We show the $\kappa_{90\%}$ of the covariance matrix of $\widehat{\mathbf{X}}$ normalized by GN/GW, while varying the group number. 'Base' and 'BN' indicate the condition number for $\mathbf{X}$ and the batch normalized output, respectively. We use a one-layer and two-layer MLP as $f(\cdot)$, in (a) and (b), respectively. Please refer to *SM* for more results.

Computational complexity. For a convolutional mini-batch input $\mathbf{X} \in \mathbb{R}^{d \times m \times H \times W}$, GW using ZCA whitening (Eqn. 5) costs $2mHWdg + mO(g^3)$. Using the more efficient 'ItN' operation (Eqn. 6), GW costs $2mHWdg + mTg^3$, where T is the iteration number. The 3×3 convolution with the same input and output feature maps costs $9mHWd^2$. The relative cost of GW for a 3×3 convolution is $\frac{2g}{9d} + \frac{Tg^3}{9HWd^2}$.

Difference from group-based BW. Our method is significantly different from the group-based BW [20], in which the whitening operation is also applied within mini-batch data. Specifically, group-based BW has difficulty in estimating the population statistics, as discussed in Section 2. Note that group-based BW is reduced to BN if the channel number in each group $c = 1$, while GW is reduced to GN if the group number $g = 1$.

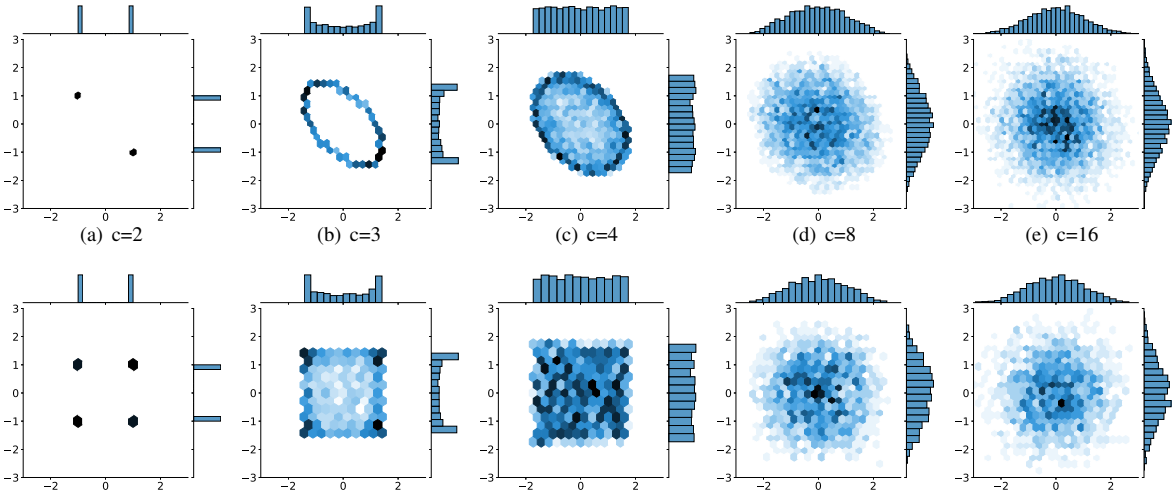
4. Revisiting the Constraint of Normalization

The normalization operation ensures that the normalized output $\widehat{\mathbf{X}} = \phi(\mathbf{X}) \in \mathbb{R}^{d \times m}$ has a stable distribution. This stability of distribution can be implicitly viewed as the constraints imposed on $\widehat{\mathbf{X}}$, which can be represented as a system of equations $\Upsilon_\phi(\widehat{\mathbf{X}})$. For example, BN provides the constraints $\Upsilon_{\phi_{BN}}(\widehat{\mathbf{X}})$ as:

$$\sum_{j=1}^{m} \widehat{\mathbf{X}}_{ij} = 0 \ and \ \sum_{j=1}^{m} \widehat{\mathbf{X}}_{ij}^2 - m = 0, \ for\ i = 1, ..., d. \tag{11}$$

Here, we define the **constraint number** of normalization to quantitatively measure the magnitude of the constraints provided by the normalization method.

Definition 1 *Given the input data $\mathbf{X} \in \mathbb{R}^{d \times m}$, the **constraint number** of a normalization operation $\phi(\cdot)$, referred to as $\zeta(\phi; \mathbf{X})$, is the number of independent equations in $\Upsilon_\phi(\widehat{\mathbf{X}})$.*



(a) c=2 (b) c=3 (c) c=4 (d) c=8 (e) c=16

(f) m=2 (g) m=3 (h) m=4 (i) m=8 (j) m=16

Figure 3. Illustration of the normalized output of GN/BN. We perform normalization over 1,680 examples sampled from a Gaussian distribution, varying the channel number for each group c of GN (the upper subfigures) and the batch size m of BN (the lower subfigures). We plot the bivariate histogram (using hexagonal bins) of the normalized output in the two-dimensional subspace, and marginal histograms (using rectangular bins) in the one-dimensional subspace.

	Normalization along a batch		Normalization along a group of neurons	
	BN	BW	GN	GW
$\zeta(\phi;\mathbf{X})$	$2d$	$\frac{d(d+3)}{2}$	$2gm$	$\frac{mg(g+3)}{2}$
$\zeta(\phi;\mathbb{D})$	$\frac{2Nd}{m}$	$\frac{Nd(d+3)}{2m}$	$2gN$	$\frac{Ng(g+3)}{2}$
Ranges of m/g	$m \geq 2$	$m \geq \frac{d+3}{2}$	$g \leq \frac{d}{2}$	$g \leq \frac{\sqrt{8d+9}-3}{2}$

Table 1. Summary of $\zeta(\phi;\mathbf{X})$, $\zeta(\phi;\mathbb{D})$ and ranges of m/g for normalization methods. The analysis can be naturally extended to CNN, following how BN (GN) extents from MLP to CNN shown in Section 3. For examples, the number of neurons to be normalized for GN/GW is $d = d'HW$, and the number of samples to be normalized for BN/BW is $m = m'HW$, given the input $\mathbf{X} \in \mathbb{R}^{d' \times m' \times H \times W}$ for CNN.

As an example, we have $\zeta(\phi_{BN};\mathbf{X}) = 2d$ from Eqn. 11. Furthermore, given training data $\mathbb{D}$ of size N, we consider the optimization algorithm with batch size m (we assume N is divisible by m). We calculate the constraint number of normalization over the entire training data $\zeta(\phi;\mathbb{D})$. Table 1 summarizes the constraint numbers of the main normalization methods discussed in this paper (please refer to the *SM* for derivation details). We can see that the whitening operation provides significantly stronger constraints than the standardization operation. Besides, the constrains get stronger for BN (GN), when reducing (increasing) the batch size (group number).

4.1. Constraint on Feature Representation

BN's benefits in accelerating the training of DNNs are mainly attributed to two reasons: 1) The distribution is more stable when fixing the first and second momentum of the activations, which reduces the internal covariant shifts [24]; 2) The landscape of the optimization objective is better conditioned [20, 46], by improving the conditioning of the activation matrix with normalization. Based on these arguments, GW/GN should have better performance when increasing the group number, due to the stronger constraints and better conditioning. However, we experimentally observe that GN/GW has significantly degenerated performance when the group number is too large (Figure 1 (b)), which is similar to the small-batch-size problem of BN/BW. We investigate the reason behind this phenomenon.

We first show that the batch size/group number has a value range, which can be mathematically derived. The normalization operation can be regarded as a way to find a solution $\widehat{\mathbf{X}}$ satisfying the constraints $\Upsilon_\phi(\widehat{\mathbf{X}})$. To ensure the solution is feasible, it must satisfy the following condition:

$$\zeta(\phi;\mathbf{X}) \leq \chi(\widehat{\mathbf{X}}), \tag{12}$$

where $\chi(\widehat{\mathbf{X}}) = md$ is the number of variables in $\widehat{\mathbf{X}}$. Based on Eqn. 12, we have $m >= 2$ for BN to ensure a feasible solution. We also provide the ranges of batch size/group number for other normalization methods in Table 1. Note that the batch size m should be larger than or equal to d to achieve a numerically stable solution for BW when using ZCA whitening in practice [20]. This also applies to GW, where g should be less than or equal to $\sqrt{d}$.

We then demonstrate that normalization eventually affects the feature representation in a certain layer. Figure 3 shows

the histogram of normalized output $\widehat{\mathbf{X}}$, by varying c of GN[2] and m of BN. We observe that: 1) the values of $\widehat{\mathbf{X}}$ are heavily constrained if c or m is too small, *e.g.*, the value of $\widehat{\mathbf{X}}$ is constrained to be $\{-1, +1\}$ if $c = 2$; 2) $\widehat{\mathbf{X}}$ is not Gaussian if c or m is too small, while BN/GN aims to produce a normalized output with a Gaussian distribution. We believe that the constrained feature representation caused by GN/GW with a large group number is the main factor leading to the degenerated performance of a network. Besides, we also observe that the normalized output of GN is more correlated than that of BN, which supports the claim that BN is more capable of improving the conditioning of activations than GN, as shown in Section 3.

We also seek to quantitatively measure the representation of a feature space. Given a set of features $\widetilde{\mathbb{D}} \in \mathbb{R}^{d \times N}$ extracted by a network, we assume the examples of $\widetilde{\mathbb{D}}$ belong in a d-dimensional hypercube $V = [-1, 1]^d$ (we can ensure that this assumption holds by dividing the maximum absolute value of each dimension). Intuitively, a powerful feature representation implies that the examples from $\widetilde{\mathbb{D}}$ spread over V with large diversity, while a weak representation indicates that they are limited to certain values without diversity. We thus define the diversity of $\widetilde{\mathbb{D}}$ based on the information entropy as follows, which can empirically indicate the representation ability of the feature space to some degree:

$$\Gamma_{d,T}(\widetilde{\mathbb{D}}) = \sum_{i=1}^{T^d} p_i \log p_i. \tag{13}$$

Here, V is evenly divided into T^d bins, and p_i denotes the probability of an example belonging to the i-th bin. We can thus calculate $\Gamma_{d,T}(\widetilde{\mathbb{D}})$ by sampling enough examples. However, calculating $\Gamma_{d,T}(\widetilde{\mathbb{D}})$ with reasonable accuracy requires $O(T^d)$ examples to be sampled from a d-dimensional space. We thus only calculate $\Gamma_{2,T}(\widetilde{\mathbb{D}})$ in practice by sampling two dimensions, and average the results. We show the diversity of group (batch) normalized features by varying the channels of each group (batch size) in Figure 4, from which we can obtain similar conclusions as in Figure 3.

In summary, our qualitative and quantitative analyses show that group/batch based normalizations have low diversity of feature representations when c/m is small. We believe these constrained feature representations affect the performance of a network, and can lead to significantly deteriorated results when the representation is over-constrained.

4.2. Effect on Representational Capacity of Model

The constraints introduced by normalization are believed to affect the representational capacity of neural networks [24], and thus the learnable scale and shift parameters are used to recover the representations [24, 4, 20, 59]. However, such an argument is seldom validated by either theoretical or empirical analysis. Theoretically analyzing the complexity measure (*e.g.*, VC dimensions [54] or the number of linear regions [40, 61]) of the representational capacity of neural networks with normalization is a challenging task, because normalized networks do not follow the assumptions for calculating linear regions or VC dimensions. Here, we conduct preliminary experiments, seeking to empirically show how normalization affects the representational capacity of a network, by varying the constraints imposed on the feature.

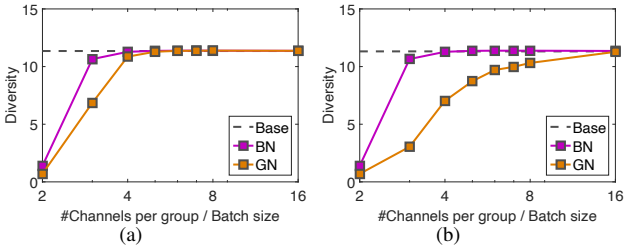
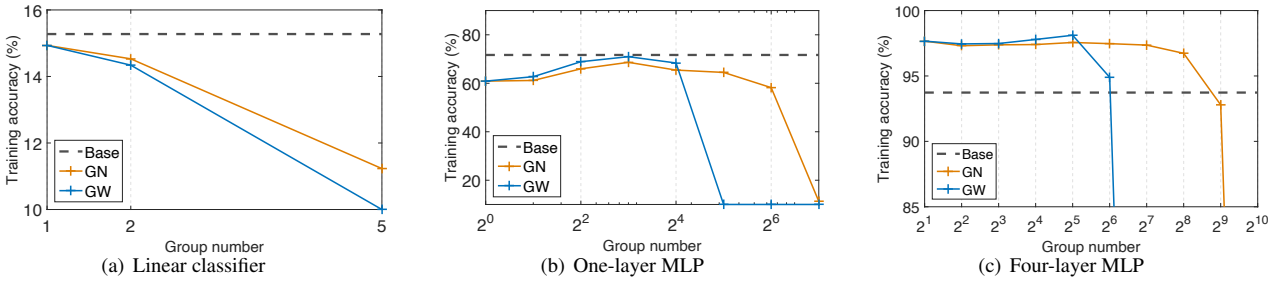


Figure 4. Diversity of group (batch) normalized features when varying the channels per group (batch size). We sample $N = 1,680,000$ examples and use $1,000^2$ bins. We use the sampled Gaussian data as features in (a) and the output of a one-layer MLP in (b). Here, 'Base' indicates the diversity of unnormalized features.

We follow the non-parametric randomization tests fitting random labels [67] to empirically compare the representational capacity of neural networks. To rule out the optimization benefits introduced by normalization, we first conduct experiments using a linear classifier, where normalization is also inserted after the linear module. We train over $1,000$ epochs using stochastic gradient descent (SGD) with a batch size of 16, and report the best training accuracy among the learning rates in $\{0.001, 0.005, 0.01, 0.05, 0.1\}$ in Figure 5 (a). We observe that GN and GW have lower training accuracy than when normalization is not used, which suggests that normalization does indeed reduce the model's representational capacity in this case. Besides, the accuracy of GN/GW decreases as the group number increases. This suggests that the model may have weaker representational ability when increasing the constraints on the feature. Note that we have the same observations regardless of whether or not the learnable scale and shift parameters of GN/GW are used.

To further consider the trade-off between the benefits of normalization on optimization and its constraints on representation, we conduct experiments on the one-layer and four-layer MLPs. The results are shown in Figure 5 (b) and (c), respectively. We observe that the model with GN/GW has significantly degenerated training accuracy when g is too large, which means that a large group number heavily limits the model's representational capacity by constraining the feature representation, as discussed in Section 4.2. We note that GW is more sensitive to the group number than GN. The main reason is that $\zeta(\phi_{GW}; \mathbb{D})$ is quadratic to g, while $\zeta(\phi_{GN}; \mathbb{D})$ is linear to it, from Table 1. Besides, we observe that GN and GW still have lower training accuracy than 'Base' on the one-layer MLP, but higher accuracy on

[2]Note that the channel number in each group $c = \frac{d}{g}$. We vary c, rather than g, for simplifying the discussion.



(a) Linear classifier (b) One-layer MLP (c) Four-layer MLP

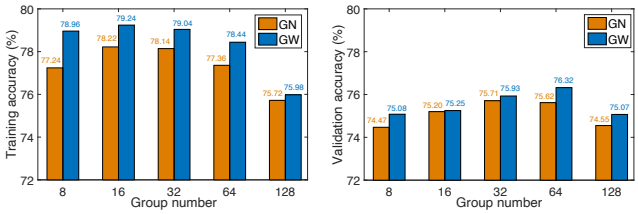
Figure 5. Comparison of model representational capacity when fitting random labels [67] on MNIST dataset using different architectures. We vary the group number of GN/GW and evaluate the training accuracy. 'Base' indicates the model without normalization. (a) Linear classifier; (b) One-layer MLP with 256 neurons in each layer; (c) Four-layer MLP with 1,280 neurons in each layer.

the four-layer MLP if the group number g is not too large. This suggests that the benefits of normalization on optimization dominate if the model's representation is not too limited. We also observe that the best training accuracy of GW is higher than that of GN. We attribute this to the fact that the whitening operation is better for improving the conditioning of optimization, compared to standardization. We also conduct similar experiments on convolutional neural networks (CNNs). Pleaser refer to the *SM* for details.

4.3. Discussion of Previous Work

Previous analyses on BN are mainly derived from the perspective of optimization [46, 33, 28, 8]. One argument is that BN can improve the conditioning of the optimization problem [46, 8, 13, 26, 11], either by avoiding the rank collapse of pre-activation matrices [11, 19] or alleviating the pathological sharpness of the landscape [46, 26, 19]. This argument has been further investigated by computing the spectrum of the Hessian for a large-scale dataset [13]. The improved conditioning enables large learning rates, thus improving the generalization [6, 38]. Another argument is that BN is scale invariant [24, 4], enabling it to adaptively adjust the learning rate [9, 17, 1, 8, 68, 32], which stabilizes and further accelerates training [24, 4]. Other analyses focus on investigating the signal and gradient propagation, either by exploiting mean-field theory [63, 57], or a neural tangent kernel (NTK) [25].

Different from these works, we are the first to investigate how BN/GN affects a model's representational capacity, by analyzing the constraint on the representation of internal features. This opens new doors in analyzing and understanding normalization methods. We also investigate how batch size affects the training performance of batch normalized networks (Figure 1 (a)), from the perspective of a model's representational capacity. Several works [48, 22, 21] have shown that batch size is related to the magnitude of stochasticity [2, 53] introduced by BN, which also affects the model's training performance. However, the stochasticity analysis [22] is specific to normalization along the batch dimension, and cannot explain why GN with a large group number has significantly worse performance (Figure 1 (b)). Our work provides a unified analysis for batch and group normalized networks.



(a) Training accuracy (b) Validation accuracy

Figure 6. Effects of group number of GW/GN on ResNet-50 for ImageNet classification. We evaluate the top-1 training and validation accuracies.

5. Large-Scale Visual Recognition Tasks

We investigate the effectiveness of our proposed GW on large-scale ImageNet classification [45], as well as COCO object detection and segmentation [35]. We use the more efficient and numerically stable 'ItN' (with $T = 5$) [22] to calculate the whitening matrix for both GW and BW, in all experiments. Our implementation is based on PyTorch [42].

5.1. ImageNet Classification

We experiment on the ImageNet dataset with 1,000 classes [45]. We use the official 1.28M training images as a training set, and evaluate the top-1 accuracy on a single-crop of 224x224 pixels in the validation set with 50k images. We investigate the ResNet [15] and ResNeXt [60] models.

5.1.1 Ablation Study on ResNet-50

We follow the same experimental setup as described in [15], except that we use two GPUs 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. Our baseline is the 50-layer ResNet (ResNet-50) trained with BN [24].

Effects of group number. We investigate the effects of group number for GW/GN, which we use to replace the BN of ResNet-50. We vary the group number g ranging in $\{8, 16, 32, 64, 128\}$ (we use the channel number if it is less than the group number in a given layer), and report the training and validation accuracies in Figure 6. We can see that

	S1	S1-B1	S1-B2	S1-B3	S1-B12
Baseline (BN)	76.23	76.23	76.23	76.23	76.23
BW [22]	76.58 (↑0.35)	76.68 (↑0.45)	76.86 (↑0.63)	76.53 (↑0.30)	76.60 (↑0.37)
BW_Σ [21]	76.63 (↑0.40)	76.80 (↑0.57)	76.76 (↑0.53)	76.52 (↑0.29)	76.71 (↑0.48)
GW	**76.76** (↑0.53)	**77.62** (↑1.39)	**77.72** (↑1.49)	**77.47** (↑1.24)	**77.45** (↑1.22)

Table 2. Effects of position when applying GW on ResNet-50 for ImageNet classification. We evaluate the top-1 validation accuracy on five architectures (**S1**, **S1-B1**, **S1-B2**, **S1-B3** and **S1-B12**).

Method	ResNet-50	ResNet-101	ResNeXt-50	ResNeXt-101
Baseline (BN) [24]	76.23	77.69	77.01	79.29
GN [59]	75.71 (↓0.52)	77.20 (↓0.49)	75.69 (↓1.32)	78.00 (↓1.29)
BW_Σ [21]	77.21 (↑0.98)	78.27 (↑0.58)	77.29 (↑0.28)	79.43 (↑0.14)
GW	**77.72** (↑1.49)	**78.71** (↑1.02)	**78.43** (↑1.42)	**80.43** (↑1.14)

Table 3. Comparison of validation accuracy on ResNets [15] and ResNeXts [60] for ImageNet. Note that we use an additional layer for BW_Σ to learn the decorrelated features, as recommended in [21].

GW has consistent improvement over GN in training accuracy, across all values of g, which indicates the advantage of the whitening operation over standardization in terms of optimization. Besides, GW also has better validation accuracy than GN. We believe this may be because the stronger constraints of GW contribute to generalization. We also observe that both GN and GW have significantly reduced training accuracy when the group number is too large (*e.g.*, g=128), which is consistent with the previous results in Figure 5.

Positions of GW. Although GW (g=64) provides slight improvement over the BN baseline (76.32% *vs.* 76.23%), it has a 90% additional time cost[3] on ResNet-50. Based on the analysis in Section 4, it is reasonable to only partially replace BN with GW in networks, because 1) normalization within a batch or a group of channels both have their advantages in improving the optimization and generalization; 2) whitening can achieve better optimization efficiency and generalization than standardization [20], but at a higher computational cost [20, 22, 49].

Here, we investigate the position at which to apply GW (g=64) in ResNet-50. ResNet and ResNeXt are both composed primarily of a stem layer and multiple bottleneck blocks [15]. We consider: 1) replacing the BN in the stem layer with GW (referred to as 'S1'); and 2) replacing the 1^{st}, 2^{nd}, 1^{st} & 2^{nd}, and 3^{rd} BNs in all the bottleneck blocks, which are referred to as 'B1', 'B2', 'B12' and 'B3', respectively. We investigate five architectures, **S1**, **S1-B1**, **S1-B2**, **S1-B3** and **S1-B12**, which have 1, 17, 17, 17 and 33 GW modules, respectively. We also perform experiments using BW [22] and BW_Σ [21] (employing a covariance matrix to estimate the population statistics of BW) for contrast.

We report the results in Table 2. BW/BW_Σ improve their BN counterparts on all architectures by a clear margin, which demonstrates the advantage of the whitening operation over standardization [22]. GW provides significant improvements over BW/BW_Σ on **S1-B1**, **S1-B2**, **S1-B3** and **S1-B12** (an absolute improvement of 0.9% on average). We attribute this to the advantage of GW in avoiding the estimation of population statistics. We also observe that GW has a slightly worse performance on **S1-B12** than on **S1-B1/S1-B2**. We believe there is a trade-off between GW and BN, in terms of affecting the model's representational capacity, optimization efficiency and generalization.

We also investigate the effect of inserting a GW/BW layer after the last average pooling (before the last linear layer) to learn the decorrelated feature representations, as proposed in [22]. This can slightly improve the performance (0.10% on average) when using GW, though the net gain is smaller than using BW (0.22%) or BW_Σ (0.43%). Please refer to the *SM* for details.

5.1.2 Validation on Larger Models

In this section, we further validate the effectiveness of GW on ResNet-101 [15], ResNeXt-50 and ResNeXt-101 [60]. We apply GW (g=64) in these models following the **S1-B2** architecture, which achieves the best performance (Table 2) without significantly increasing the computational cost (it is only increased by roughly 23%). For comparison, we also apply BW_Σ following the 'S1-B2' architecture, combining the learning of decorrelated features [21] (BW_Σ has a slightly improved performance compared to BW [21]). Our baselines are the original networks trained with BN, and we also provide the results trained with GN.

The results are shown in Table 3. We can see that 1) our method improves the baseline (BN) by a significant margin (between 1.02% and 1.49%); and 2) BW_Σ has consistently better performance than BN, but the net gain is reduced on wider networks (RexNeXt-50 and ResNeXt-101), which is probably caused by the difficulty in estimating the population statistics. We also conduct experiments using more advanced training strategies (*e.g.*, cosine learning rate decay [36], label smoothing [16] and mixup [69]) and GW again improves the baseline consistently. Please refer to the *SM* for details.

[3]Note that our implementations are based on the APIs provided by PyTorch and are not finely optimized. For more discussion on time costs, please refer to the *SM*.

	2fc head box			4conv1fc head box		
Method	AP^{bbox}	AP^{bbox}_{50}	AP^{bbox}_{75}	AP^{bbox}	AP^{bbox}_{50}	AP^{bbox}_{75}
$BN^\dagger$	36.31	58.39	38.83	36.39	57.22	39.56
GN	36.62$_{(\uparrow 0.31)}$	58.91$_{(\uparrow 0.52)}$	39.32$_{(\uparrow 0.49)}$	37.86$_{(\uparrow 1.47)}$	58.96$_{(\uparrow 1.74)}$	40.76$_{(\uparrow 1.20)}$
GW	**38.13**$_{(\uparrow 1.82)}$	**60.63**$_{(\uparrow 2.24)}$	**41.08**$_{(\uparrow 2.25)}$	**39.60**$_{(\uparrow 3.21)}$	**61.12**$_{(\uparrow 3.90)}$	**43.25**$_{(\uparrow 3.69)}$

Table 4. Detection results (%) on COCO using the Faster R-CNN framework implemented in [39]. We use ResNet-50 as the backbone, combined with FPN. All models are trained by 1x lr scheduling (90k iterations), with a batch size of 16 on eight GPUs.

Method	AP^{bbox}	AP^{bbox}_{50}	AP^{bbox}_{75}	AP^{mask}	AP^{mask}_{50}	AP^{mask}_{75}
$BN^\dagger$	42.24	63.00	46.19	37.53	59.82	39.96
GN	42.18$_{(\downarrow 0.06)}$	63.22$_{(\uparrow 0.22)}$	46.00$_{(\downarrow 0.19)}$	37.54$_{(\uparrow 0.01)}$	60.18$_{(\uparrow 0.36)}$	39.99$_{(\uparrow 0.03)}$
GW	**44.41**$_{(\uparrow 2.17)}$	**65.36**$_{(\uparrow 2.36)}$	**48.67**$_{(\uparrow 2.48)}$	**39.17**$_{(\uparrow 1.64)}$	**62.13**$_{(\uparrow 2.31)}$	**41.95**$_{(\uparrow 1.99)}$

Table 5. Detection and segmentation results (%) on COCO using the Mask R-CNN framework implemented in [39]. We use ResNeXt-101 as the backbone, combined with FPN. All models are trained by 1x lr scheduling (180k iterations), with a batch size of 8 on eight GPUs.

5.2. Object Detection and Segmentation on COCO

We fine-tune the models trained on ImageNet for object detection and segmentation on the COCO benchmark [35]. We experiment on the Faster R-CNN [44] and Mask R-CNN [14] frameworks using the publicly available codebase 'maskrcnn-benchmark' [39]. We train the models on the COCO $train2017$ set and evaluate on the COCO $val2017$ set. We report the standard COCO metrics of average precision (AP), AP_{50}, and AP_{75} for bounding box detection (AP^{bbox}) and instance segmentation (AP^m) [35]. For BN, we use its frozen version (indicated by $BN^\dagger$) when fine-tuning for object detection [59].

Results on Faster R-CNN. For the Faster R-CNN framework, we use the ResNet-50 models pre-trained on ImageNet (Table 3) as the backbones, combined with the feature pyramid network (FPN) [34]. We consider two setups: 1) we use the box head consisting of two fully connected layers ('2fc') without a normalization layer, as proposed in [34]; 2) following [59], we replace the '2fc' box head with '4conv1fc', which can better leverage GN, and apply GN/GW to the FPN and box head. We use the default hypeparameter configurations from the training scripts provided by the codebase [39] for Faster R-CNN. The results are reported in Table 4. The GW pre-trained model improves $BN^\dagger$ and GN by 1.82% and 1.51% AP, respectively. By adding GW/GN to the FPN and '4conv1fc' head box, GW improves $BN^\dagger$ and GN by 3.21% and 1.74% AP, respectively.

Results on Mask R-CNN. For the Mask R-CNN framework, we use the ResNeXt-101 [60] models pre-trained on ImageNet (Table 3) as the backbones, combined with FPN. We use the '4conv1fc' box head, and apply GN/GW to the FPN, box head and mask head. We again use the default hypeparameter configurations from the training scripts provided by the codebase for Mask R-CNN [39]. The results are shown in Table 5. GW achieves 44.41% in box AP and 39.17% in mask AP, an improvement over $BN^\dagger$ of 2.17% and 1.64%, respectively.

Small-batch-size training of BNs. Here, we further show that the network mixed with BNs and GWs can still work well under small-batch-size scenarios. As illustrated in [23], one main cause of BN's small-batch-size problem is the inaccurate estimation between training and inference distributions, which is amplified for a network with increased BN layers (these inaccuracies are compounded with depth). We believe inserting GW (which ensures the same distribution between training and inference) between consecutive BN layers will 'break' these compounding inaccuracies, thus relieving the small-batch-size problem of BNs in a network. We train Faster R-CNN from scratch and use normal BN that is not frozen. We follow the same setup as in the previous experiment (*e.g.*, two images/GPUs). We find that using all BNs only obtains 25.10% AP, while 28.37% AP is achieved using our mixture of BNs and GWs (the **S1-B2** architecture). Note that using all GNs (GWs) obtains 28.19% (28.79%) AP. This experiment further validates that our mixture of BNs and GWs may also help mitigate the small-batch-size problem of the BNs in a network.

6. Conclusion and Future Work

In this paper, we proposed group whitening (GW), which combines the advantages of normalization within a group of channels and the whitening operation. The effectiveness of GW was validated on large-scale visual recognition tasks. Furthermore, we also analyzed the feature constraints imposed by normalization methods, enabling further understanding of how the batch size (group number) affects the performance of batch (group) normalized networks from the perspective of representational capacity. This analysis can provide theoretical guidance for applying GW and other normalization methods in practice. It would be interesting to build a unified framework to further investigate the effects of normalization in representation, optimization and generalization, by combining the proposed constraint analysis with the previous conditioning analysis [31, 19] and stochasticity analysis [3, 21]. Our GW has also the potentialities to be used as a basic module in the switchable normalization methods [37, 41, 70] to improve their performance.

References

[1] Sanjeev Arora, Zhiyuan Li, and Kaifeng Lyu. Theoretical analysis of auto rate-tuning by batch normalization. In *ICLR*, 2019. 6

[2] Andrei Atanov, Arsenii Ashukha, Dmitry Molchanov, Kirill Neklyudov, and Dmitry Vetrov. Uncertainty estimation via stochastic batch normalization. In *ICLR Workshop*, 2018. 6

[3] Hossein Azizpour, Mattias Teye, and Kevin Smith. Bayesian uncertainty estimation for batch normalized deep networks. In *International Conference on Machine Learning (ICML)*, 2018. 8

[4] Lei Jimmy Ba, Ryan Kiros, and Geoffrey E. Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016. 1, 2, 5, 6

[5] Dario A. Bini, Nicholas J. Higham, and Beatrice Meini. Algorithms for the matrix pth root. *Numerical Algorithms*, 39(4):349–378, Aug 2005. 2

[6] Johan Bjorck, Carla Gomes, and Bart Selman. Understanding batch normalization. In *NeurIPS*, 2018. 6

[7] Tianle Cai, Shengjie Luo, Keyulu Xu, Di He, Tie-yan Liu, and Liwei Wang. Graphnorm: A principled approach to accelerating graph neural network training. *arXiv preprint arXiv:2009.03294*, 2020. 1

[8] Yongqiang Cai, Qianxiao Li, and Zuowei Shen. A quantitative analysis of the effect of batch normalization on gradient descent. In *ICML*, 2019. 6

[9] Minhyung Cho and Jaehyung Lee. Riemannian approach to batch normalization. In *NeurIPS*, 2017. 6

[10] Tim Cooijmans, Nicolas Ballas, César Laurent, and Aaron C. Courville. Recurrent batch normalization. In *ICLR*, 2017. 2

[11] Hadi Daneshmand, Jonas Kohler, Francis Bach, Thomas Hofmann, and Aurelien Lucchi. Theoretical understanding of batch-normalization: A markov chain perspective. *arXiv preprint arXiv:2003.01652*, 2020. 6

[12] Guillaume Desjardins, Karen Simonyan, Razvan Pascanu, and koray kavukcuoglu. Natural neural networks. In *NeurIPS*, 2015. 1

[13] Behrooz Ghorbani, Shankar Krishnan, and Ying Xiao. An investigation into neural net optimization via hessian eigenvalue density. In *ICML*, 2019. 6

[14] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross B. Girshick. Mask R-CNN. In *ICCV*, 2017. 8

[15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 1, 2, 6, 7

[16] Tong He, Zhi Zhang, Hang Zhang, Zhongyue Zhang, Junyuan Xie, and Mu Li. Bag of tricks for image classification with convolutional neural networks. In *CVPR*, 2019. 7

[17] Elad Hoffer, Ron Banner, Itay Golan, and Daniel Soudry. Norm matters: efficient and accurate normalization schemes in deep networks. In *NeurIPS*, 2018. 6

[18] Gao Huang, Zhuang Liu, and Kilian Q. Weinberger. Densely connected convolutional networks. In *CVPR*, 2017. 1

[19] Lei Huang, Jie Qin, Li Liu, Fan Zhu, and Ling Shao. Layer-wise conditioning analysis in exploring the learning dynamics of dnns. In *ECCV*, 2020. 6, 8

[20] Lei Huang, Dawei Yang, Bo Lang, and Jia Deng. Decorrelated batch normalization. In *CVPR*, 2018. 1, 2, 3, 4, 5, 7

[21] Lei Huang, Lei Zhao, Yi Zhou, Fan Zhu, Li Liu, and Ling Shao. An investigation into the stochasticity of batch whitening. In *CVPR*, 2020. 2, 6, 7, 8

[22] Lei Huang, Yi Zhou, Fan Zhu, Li Liu, and Ling Shao. Iterative normalization: Beyond standardization towards efficient whitening. In *CVPR*, 2019. 1, 2, 3, 6, 7

[23] Sergey Ioffe. Batch renormalization: Towards reducing minibatch dependence in batch-normalized models. In *NeurIPS*, 2017. 1, 2, 8

[24] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, 2015. 1, 2, 3, 4, 5, 6, 7

[25] Arthur Jacot, Franck Gabriel, and Clément Hongler. Freeze and chaos for dnns: an ntk view of batch normalization, checkerboard and boundary effects. *arXiv preprint arXiv:1907.05715*, 2019. 6

[26] Ryo Karakida, Shotaro Akaho, and Shun-ichi Amari. The normalization method for alleviating pathological sharpness in wide neural networks. In *NeurIPS*. 2019. 6

[27] Agnan Kessy, Alex Lewin, and Korbinian Strimmer. Optimal whitening and decorrelation. *The American Statistician*, 72(4):309–314, 2018. 2, 3

[28] Jonas Kohler, Hadi Daneshmand, Aurelien Lucchi, Thomas Hofmann, Ming Zhou, and Klaus Neymeyr. Exponential convergence rates for batch normalization: The power of length-direction decoupling in non-convex optimization. In *AISTATS*, 2019. 6

[29] Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Joan Puigcerver, Jessica Yung, Sylvain Gelly, and Neil Houlsby. Big transfer (bit): General visual representation learning. In *ECCV*, 2020. 1

[30] César Laurent, Gabriel Pereyra, Philemon Brakel, Ying Zhang, and Yoshua Bengio. Batch normalized recurrent neural networks. In *ICASSP*, 2016. 2

[31] Yann LeCun, Léon Bottou, Genevieve B. Orr, and Klaus-Robert Müller. Effiicient backprop. In *Neural Networks: Tricks of the Trade*, pages 9–50, 1998. 1, 8

[32] Zhiyuan Li and Sanjeev Arora. An exponential learning rate schedule for batch normalized networks. In *ICLR*, 2020. 6

[33] Xiangru Lian and Ji Liu. Revisit batch normalization: New understanding and refinement via composition optimization. In *AISTATS*, 2019. 6

[34] Tsung-Yi Lin, Piotr Dollár, Ross B. Girshick, Kaiming He, Bharath Hariharan, and Serge J. Belongie. Feature pyramid networks for object detection. In *CVPR*, 2017. 8

[35] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft coco: Common objects in context. In *ECCV*, 2014. 2, 6, 8

[36] Ilya Loshchilov and Frank Hutter. SGDR: stochastic gradient descent with restarts. In *ICLR*, 2017. 7

[37] Ping Luo, Jiamin Ren, and Zhanglin Peng. Differentiable learning-to-normalize via switchable normalization. *arXiv preprint arXiv:1806.10779*, 2018. 1, 8

[38] Ping Luo, Xinjiang Wang, Wenqi Shao, and Zhanglin Peng. Towards understanding regularization in batch normalization. In *ICLR*, 2019. 6

[39] Francisco Massa and Ross Girshick. maskrcnn-benchmark: Fast, modular reference implementation of Instance Segmentation and Object Detection algorithms in PyTorch. https://github.com/facebookresearch/maskrcnn-benchmark, 2018. Accessed: 09-26-2019. 8

[40] Guido F Montufar, Razvan Pascanu, Kyunghyun Cho, and Yoshua Bengio. On the number of linear regions of deep neural networks. In *NeurIPS*, 2014. 5

[41] Xingang Pan, Xiaohang Zhan, Jianping Shi, Xiaoou Tang, and Ping Luo. Switchable whitening for deep representation learning. In *ICCV*, 2019. 1, 3, 8

[42] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in PyTorch. In *NeurIPS Autodiff Workshop*, 2017. 3, 6

[43] Mengye Ren, Renjie Liao, Raquel Urtasun, Fabian H. Sinz, and Richard S. Zemel. Normalizing the normalizers: Comparing and extending network normalization schemes. In *ICLR*, 2017. 1

[44] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In *NeurIPS*, 2015. 8

[45] 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

[46] Shibani Santurkar, Dimitris Tsipras, Andrew Ilyas, and Aleksander Madry. How does batch normalization help optimization? In *NeurIPS*, 2018. 1, 4, 6

[47] Wenqi Shao, Shitao Tang, Xingang Pan, Ping Tan, Xiaogang Wang, and Ping Luo. Channel equilibrium networks for learning deep representation. In *ICML*, 2020. 3

[48] Alexander Shekhovtsov and Boris Flach. Stochastic normalizations as bayesian learning. In *ACCV*, 2018. 1, 6

[49] Aliaksandr Siarohin, Enver Sangineto, and Nicu Sebe. Whitening and coloring batch transform for gans. In *ICLR*, 2019. 2, 3, 7

[50] Saurabh Singh and Abhinav Shrivastava. Evalnorm: Estimating batch normalization statistics for evaluation. In *ICCV*, 2019. 1

[51] Christian Szegedy, Sergey Ioffe, and Vincent Vanhoucke. Inception-v4, inception-resnet and the impact of residual connections on learning. In *AAAI*, 2017. 1

[52] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *CVPR*, 2016. 1

[53] Mattias Teye, Hossein Azizpour, and Kevin Smith. Bayesian uncertainty estimation for batch normalized deep networks. In *ICML*, 2018. 6

[54] Vladimir N Vapnik. An overview of statistical learning theory. *IEEE transactions on neural networks*, 10(5):988–999, 1999. 5

[55] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, 2017. 2

[56] Guangrun Wang, Jiefeng Peng, Ping Luo, Xinjiang Wang, and Liang Lin. Kalman normalization: Normalizing internal representations across network layers. In *NeurIPS*, 2018. 1

[57] Mingwei Wei, James Stokes, and David J. Schwab. Mean-field analysis of batch normalization. *arXiv preprint arXiv:1903.02606*, 2019. 6

[58] Simon Wiesler and Hermann Ney. A convergence analysis of log-linear training. In *NeurIPS*, 2011. 1

[59] Yuxin Wu and Kaiming He. Group normalization. In *ECCV*, 2018. 1, 2, 3, 5, 7, 8

[60] Saining Xie, Ross B. Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *CVPR*, 2017. 2, 6, 7, 8

[61] Huan Xiong, Lei Huang, Mengyang Yu, Li Liu, Fan Zhu, and Ling Shao. On the number of linear regions of convolutional neural networks. In *ICML*, 2020. 5

[62] Jingjing Xu, Xu Sun, Zhiyuan Zhang, Guangxiang Zhao, and Junyang Lin. Understanding and improving layer normalization. In *NeurIPS*, 2019. 2

[63] Greg Yang, Jeffrey Pennington, Vinay Rao, Jascha Sohl-Dickstein, and Samuel S. Schoenholz. A mean field theory of batch normalization. In *ICLR*, 2019. 6

[64] Chengxi Ye, Matthew Evanusa, Hua He, Anton Mitrokhin, Tom Goldstein, James A. Yorke, Cornelia Fermuller, and Yiannis Aloimonos. Network deconvolution. In *ICLR*, 2020. 3

[65] Adams Wei Yu, David Dohan, Minh-Thang Luong, Rui Zhao, Kai Chen, Mohammad Norouzi, and Quoc V Le. Qanet: Combining local convolution with global self-attention for reading comprehension. In *ICLR*, 2018. 2

[66] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. In *BMVC*, 2016. 1

[67] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. In *ICLR*, 2017. 5, 6

[68] Guodong Zhang, Chaoqi Wang, Bowen Xu, and Roger B. Grosse. Three mechanisms of weight decay regularization. In *ICLR*, 2019. 6

[69] Hongyi Zhang, Moustapha Cisse, Yann N. Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. In *ICLR*, 2018. 7

[70] Ruimao Zhang, Zhanglin Peng, Lingyun Wu, Zhen Li, and Ping Luo. Exemplar normalization for learning deep representation. In *CVPR*, 2020. 8