Representative Batch Normalization with Feature Calibration

Shang-Hua Gao[1] Qi Han[1] Duo Li[2] Ming-Ming Cheng[1] * Pai Peng[3]
TKLNDST, CS, Nankai University[1] HKUST[2] Tencent[3]
http://mmcheng.net/rbn

Abstract

Batch Normalization (BatchNorm) has become the default component in modern neural networks to stabilize training. In BatchNorm, centering and scaling operations, along with mean and variance statistics, are utilized for feature standardization over the batch dimension. The batch dependency of BatchNorm enables stable training and better representation of the network, while inevitably ignores the representation differences among instances. We propose to add a simple yet effective feature calibration scheme into the centering and scaling operations of BatchNorm, enhancing the instance-specific representations with the negligible computational cost. The centering calibration strengthens informative features and reduces noisy features. The scaling calibration restricts the feature intensity to form a more stable feature distribution. Our proposed variant of BatchNorm, namely Representative BatchNorm, can be plugged into existing methods to boost the performance of various tasks such as classification, detection, and segmentation. The source code is available in http://mmcheng.net/rbn.

1. Introduction

Convolutional Neural Networks (CNNs) [19, 30, 52] have boosted the performance of various computer vision tasks [10, 18, 29, 48] with its powerful representation ability. While with the growth of structural complexity and model parameters, CNNs are facing more training difficulties. Batch normalization (BatchNorm) [24] eases the training difficulty by constraining intermediate features within the normalized distribution with mini-batch statistical information. In BatchNorm, the reliance on mini-batch information is built on the assumption that features generated from different instances fit into the same distribution within a channel [24, 65]. However, this assumption can not always hold on two cases [7, 32, 54, 68]: i) the possible inconsistency between the mini-batch statistics in training and the running statistics in testing; ii) the instances in the testing set may not always fall into the distribution of the training set. To avoid the side effects introduced by these two kinds of inconsistencies, some works [2, 59, 63] utilize instance-specific statistics instead of mini-batch statistics to normalize intermediate features. However, due to the lack of batch information, the training instability makes their performance inferior to BatchNorm in many cases [37, 55]. Other works utilize mini-batch and instance statistics by combining multiple normalization techniques [36, 37, 51] or introducing attention mechanisms [25, 31, 32, 39]. However, these methods usually introduce more overheads, making them not friendly in practical usage. A question has raised, can we maintain the mini-batch benefits of BatchNorm and enhance the instance-specific representations with some minor adjustments? To answer this question, we propose a simple yet effective feature calibration scheme to calibrate the feature standardization operation of BatchNorm with a negligible cost.

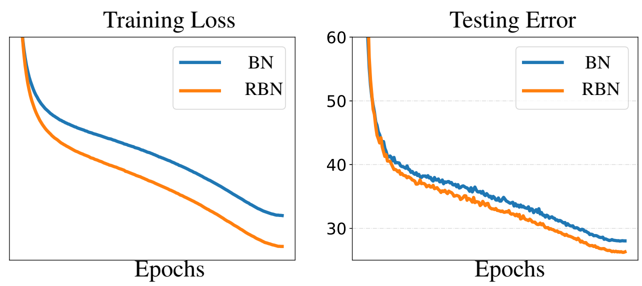

Figure 1. Image classification results on the ImageNet dataset. MobileNet v2 [50] equipped with Representative BatchNorm achieves smaller training loss and testing error than using BatchNorm. Representative BatchNorm enhances the instance-specific representations and maintains the benefit of the BatchNorm.

BatchNorm is composed of the feature standardization and affine transformation operation. In this paper, we focus on the standardization operation composed of feature centering and scaling operations. During training, based on mini-batch statistics, the centering operation ensures features to have the zero-mean property, and the scaling operation makes features to have unit-variance. The zero-mean

*M.M. Cheng (cmm@nankai.edu.cn) is the corresponding author.

and unit-variance property of features cannot always be maintained during testing due to the statistics inconsistency and the instance inconsistency. Centering with inappropriate running mean values makes centered features contain extra noises or lose some informative representations after activation. When the mean value of testing instance features is below the running mean value, as shown in Fig. 2(a), some representative features are mistakenly removed by the activation. In contrast, as shown in Fig. 2(b), noises with small value should be filtered out by the activation are kept when the mean value of features is greater than the running mean value. Also, inappropriate running variance causes the scaling operation to produce scaled features with too small or too large intensity, as shown in Fig. 2(c) and (d), resulting in unstable feature distribution among channels. For example, suppose a scaled feature from one channel is much larger than the other channels in the same layer during testing. The feature in this channel would dominate the features produced by the next convolutional layer.

To reduce the side effect introduced by some inappropriate running statistics while maintaining the benefits of BatchNorm, we utilize instance-specific statistics to calibrate the centering and scaling operations with a negligible cost. We propose the centering calibration to strengthen representative features and reduce noisy features by moving features with instance-specific statistics. The scaling calibration accordingly scales the intensity of features with statistics of instances to produce a more stable feature distribution. These two calibrations only introduce three weights in each channel and require a negligible computational cost.

We propose the Representative Batch Normalization (RBN) by adding the centering and scaling calibrations to the BatchNorm, to make the intermediate features normalized by BatchNorm to be more representative. Fig. 1 shows the model equipped with Representative BatchNorm achieves smaller training loss and testing error than using BatchNorm. We show that Representative BatchNorm can replace the BatchNorm in existing methods to boost the performance of various tasks such as classification, detection, and segmentation with the negligible cost and parameters.

2. Related Work

Various normalization (Norm) techniques [6,22,49] have been proposed to achieve more effective feature normalization [24, 59, 63] and task specified feature transformation [8, 12, 69].

Statistics for Normalization. Statistics calculated from different dimensions and regions are utilized for feature Normalization. BatchNorm [24] utilized mini-batch statistics to normalize intermediate features and stabilize training. A larger mini-batch across multiple GPUs is applied in Synchronized BN [44] to obtain more accurate batch statistics. In contrast, GhostNorm [11] acquired statistics on small virtual batch to reduce the generalization error. EvalNorm [54] re-estimated normalization statistics during evaluation. KalmanNorm [60] estimated the statistics of a layer with its preceding layers. Cross-iteration BatchNorm [67] obtained statistics from recent iterations. LayerNorm [2], InstanceNorm [59], and GroupNorm [63] normalized features with statistics from the channel, sample, and channel group dimensions, respectively. Instead of calculating statistics using all pixels within a dimension, local normalization techniques [41, 47] utilized statistics of neighboring regions. Normalizing with mini-batch independent statistics can improve the model stability when mini-batch statistics are particularly inaccurate, *i.e.,* training batch size is too small. However, due to the lack of batch information, the training instability makes their performance inferior to BatchNorm in many cases [37, 55].

Combinations of Multiple Dimensional Normalization. Some works take advantage of statistics from different dimensions by combining multiple normalizations. MixtureNorm [27] disentangled distribution into different modes via a Gaussian mixture model and independently normalized features within each mode. ModeNorm [9] extended normalization statistics to multiple modes to address the heterogeneous nature of complex datasets. SwitchNorm series [37, 38, 51] learned to switch among exiting normalization techniques for different dimensions according to the task [37, 51] or samples [38]. In comparison, our RBN tends to calibrate BN features, which can be regarded as a new module in the normalization set used by the SN. Batch-Channel Norm [46] combined the BatchNorm with channel-normalized techniques to eliminate singularities. The grouping mechanism in GroupNorm is expanded to both channel and batch dimensions by Batch group Norm [55]. Generalized BatchNorm [68] applied a variety of alternative statistics and deviation measures for standardization. Extended BatchNorm [35] computes the mean and variance along different dimensions to enhance training stability. However, these multi-normalization combining methods usually require the extra computational cost to normalize features among different dimensions.

Alternatives of Standardization. Instead of using the standardization composed of the centering and scaling, some works utilized standardization alternatives. L^1-Norm was utilized in [20, 62] to replace the commonly used L^2-Norm for scaling operation. Huang *et al*. [23] proposed to use the iteratively ZCA whitening to normalize features. Filter response Norm [53] only performed the instance specified scaling operation and abandoned the centering operation. Similarly, Yan *et al*. [65] performed scaling only in BatchNorm to handle the small batch size training. Liu *et al*. [34] searched to combine the normalization with activation. Still, our proposed calibration scheme can be applied to these alternatives of standardization.

Conditional Transformation. Some works modified the affine transformation conditioned on the task or instance specified properties [12, 31, 32]. Conditional transformations after the normalization have been widely used in generative networks [12, 28, 40, 58], image synthesis [3, 42, 56, 69], visual reasoning [45], meta-learning [4], language-vision task [8], style transfer [26, 42], and domain adaptation [5, 61]. Some works introduced the attention mechanisms to generate the weights for affine transformation [25, 31, 32, 39], forming a more generalized transformation. Our work focuses on the calibration of feature standardization, thus can cooperate with these conditional transformation methods.

3. Method

3.1. Revisiting Batch Normalization

We first revisit the formulation of BatchNorm. BatchNorm is composed of the feature centering, feature scaling, and affine transformation operation. Given the input feature $\mathbf{X} \in \mathbb{R}^{N \times C \times H \times W}$, where N, C, H, and W are batch size, the number of channels, height, width of the input feature, respectively, the centering, scaling, and affine transformation can be written as follows [24]:

$$\begin{aligned} &\text{Centering} : \mathbf{X}_m = X - \mathrm{E}(\mathbf{X}), \\ &\text{Scaling} : \mathbf{X}_s = \frac{\mathbf{X}_m}{\sqrt{\mathrm{Var}(\mathbf{X}) + \epsilon}}, \\ &\text{Affine} : \mathbf{Y} = \mathbf{X}_s \gamma + \beta. \end{aligned} \tag{1}$$

$\mathrm{E}(\mathbf{X})$ and $\mathrm{Var}(\mathbf{X})$ denote the mean and variance, and are used for centering and scaling. γ and β are learned scale and bias factors for affine transformation, and ϵ is used to avoid zero variance. During training, mean and variance values calculated within the mini-batch are written as follows:

$$\begin{aligned} \mu_B &= \frac{1}{NHW} \sum_{n=1}^{N} \sum_{h=1}^{H} \sum_{w=1}^{W} \mathbf{X}_{(n,c,h,w)}, \\ \sigma_B^2 &= \frac{1}{NHW} \sum_{n=1}^{N} \sum_{h=1}^{H} \sum_{w=1}^{W} \left(\mathbf{X}_{(n,c,h,w)} - \mu_B\right)^2 . \end{aligned} \tag{2}$$

The statistics of $\mathrm{E}(\mathbf{X})$ and $\mathrm{Var}(\mathbf{X})$ are accumulated over the dataset during training, while keeping fixed during testing. The running mean and variance are obtained by:

$$\begin{aligned} \mathrm{E}(\mathbf{X}) &\Leftarrow m\mathrm{E}(\mathbf{X}) + (1-m)\mu_B, \\ \mathrm{Var}(\mathbf{X}) &\Leftarrow m\,\mathrm{Var}(\mathbf{X}) + (1-m)\sigma_B^2, \end{aligned} \tag{3}$$

where m is the accumulation momentum.

The mini-batch statistics μ_B and σ_B^2 over the mini-batch are utilized in BatchNorm to stabilize training, and model parameters are trained to fit features normalized by batch statistics. However, the mini-batch and running statistics cannot be strictly aligned, and the testing instances may not always fit in the running distribution accumulated during training. Therefore, the inconsistency between the training and testing process weakens the role of BatchNorm [66]. However, simply abandon the mini-batch statistics to use instance statistics hurts the model performance in many cases [2, 59, 63], as BatchNorm stabilizes the training with distributions over many training instances. Therefore, we propose to calibrate the centering and scaling operations with instance statistics to enhance the instance-specific representations and maintain the mini-batch benefits of BatchNorm.

3.2. Representative Batch Normalization

We aim to enhance the instance-specific representations and maintain the benefits of BatchNorm. In this work, we focus on the feature standardization operation composed of feature centering and feature scaling. Our proposed Representative Batch Normalization, which is equipped with the simple yet effective feature calibration scheme, strengthens instance specified features and produces a more stable feature distribution.

3.2.1 Statistics for Calibration

The statistics of certain instance-specific features are needed for calibrating the running statistics in BatchNorm. This paper mainly studies statistics over channel dimensions as the BatchNorm is designed to count statistics over channels. The channel dimension statistics, the mean μ_c and variance σ_c^2 of feature channels are given as follows:

$$\begin{aligned} \mu_c &= \frac{1}{HW} \sum_{h=1}^{H} \sum_{w=1}^{W} \mathbf{X}_{(n,c,h,w)}, \\ \sigma_c^2 &= \frac{1}{HW} \sum_{h=1}^{H} \sum_{w=1}^{W} \left(\mathbf{X}_{(n,c,h,w)} - \mu_c\right)^2 . \end{aligned} \tag{4}$$

We also study the effect of statistics over the spatial dimension in Tab. 3. We will apply these statistics to calibrate the centering and scaling operations of the BatchNorm layer.

3.2.2 Centering Calibration

The running mean values count the mean statistics of channels over the training dataset. When ignoring the effect of the affine transformation, features with larger values than the running mean are kept after the following activation layer, and vice versa. However, the running mean value of channels may not be accurate when the features vary widely. As shown in Fig. 2, we draw a line on the image and sample

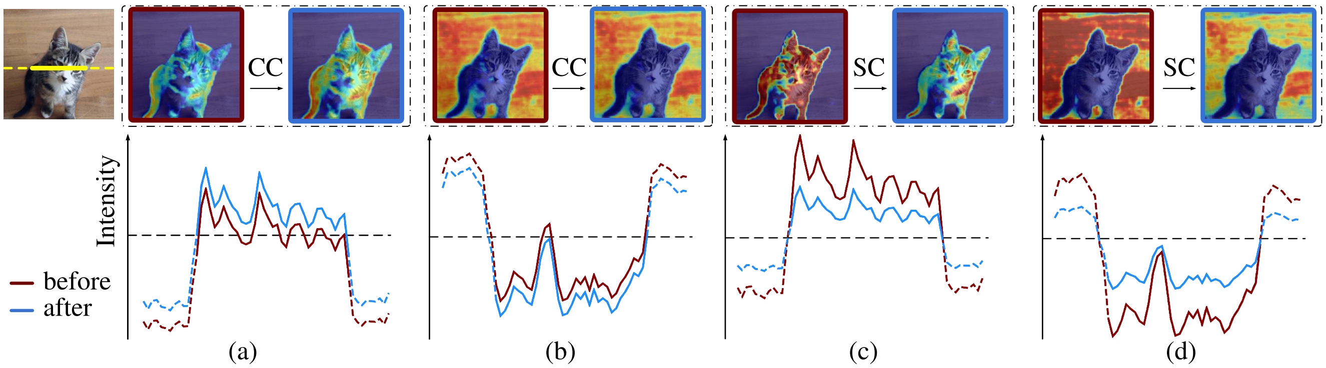

Figure 2. Representative BatchNorm is composed of centering calibration (CC) and scale calibration (SC). The feature intensity distributions are sampled from the yellow line of the image. The centering calibration (a) strengths representative features and (b) reduces the noisy features. The scale calibration in (c) and (d) restricts the feature intensity to form a more stable feature distribution.

feature intensity along this line. In that case, directly centering features with the running mean value may lose informative representations (Fig. 2(a)) or introduce extra noises (Fig. 2(b)). To make the centering operation less rely on the running mean value, we add a centering calibration scheme driven by instance statistics.

Formulation of Centering Calibration. The centering calibration is added before the centering operation of the original BatchNorm layer. Given input feature $\mathbf{X}$, the centering calibration of features is written as follows:

$$\mathbf{X}_{cm(n,c,h,w)} = \mathbf{X}_{(n,c,h,w)} + w_m \odot \mathbf{K}_m, \tag{5}$$

where $w_m \in \mathbb{R}^{1\times C\times 1\times 1}$ is the learnable weight vector and $\mathbf{K}_m$ is the statistics of feature $\mathbf{X}$ that can have multiple shapes, *i.e.,* $\mathbf{K}_m \in \mathbb{R}^{N\times C\times 1\times 1}$ or $\mathbf{K}_m \in \mathbb{R}^{N\times 1\times H\times W}$. We set $\mathbf{K}_m$ to $\mu_c \in \mathbb{R}^{N\times C\times 1\times 1}$ by default. $\odot$ is the dot product operator that broadcast two features to the same shape and then conduct dot product. For notation simplicity, we replace $\odot$ with $\cdot$ where there is no ambiguity.

Mechanism Proof. The instance-related term $w_m \cdot \mathbf{K}_m$ in Eqn. (5) introduces the instance specified information. The learnable weight w_m is proposed to calibrate the centering operation by balancing mini-batch and instance-specific statistics. We show the theoretical proof of the mechanism of centering calibration. Suppose the running mean of features before and after the centering calibration are $\mathrm{E}(\mathbf{X})$ and $\mathrm{E}(\mathbf{X}_{cm})$, respectively. When the $\mathbf{K}_m$ in Eqn. (5) is set to μ_c, the running mean of $\mathbf{K}_m$ is equal to $\mathrm{E}(\mathbf{X})$. Therefore, according to Eqn. (5), the relation between $\mathrm{E}(\mathbf{X})$ and $\mathrm{E}(\mathbf{X}_{cm})$ can be written as:

$$\mathrm{E}(\mathbf{X}_{cm}) = (1 + w_m) \cdot \mathrm{E}(\mathbf{X}). \tag{6}$$

According to the centering operation shown in Eqn. (1), the centered features with/without the centering calibration can be written as:

$$\begin{aligned} \mathbf{X}_{cal} &= \mathbf{X}_{cm} - \mathrm{E}(\mathbf{X}_{cm}), \\ \mathbf{X}_{no} &= \mathbf{X} - \mathrm{E}(\mathbf{X}). \end{aligned} \tag{7}$$

The difference between these two centralized features is written as follows:

$$\begin{aligned} &\mathbf{X}_{cal} - \mathbf{X}_{no} \\ &= (\mathbf{X}_{cm} - \mathrm{E}(\mathbf{X}_{cm})) - (\mathbf{X} - \mathrm{E}(\mathbf{X})) \\ &= \mathbf{X} + w_m \cdot \mathbf{K}_m - (1 + w_m) \cdot \mathrm{E}(\mathbf{X}) - (\mathbf{X} - \mathrm{E}(\mathbf{X})) \\ &= w_m \cdot (\mathbf{K}_m - \mathrm{E}(\mathbf{X})). \end{aligned} \tag{8}$$

When the absolute value of w_m is close to zero, the centering operation still relies on the running statistics. In contrast, the importance of instance-specific features grows when $|w_m|$ is larger. There are two cases where features are strengthened or weakened after the centering calibration considering the $w_m \cdot \mathbf{K}_m$. On the condition that $w_m > 0$, when $\mathbf{K}_m > \mathrm{E}(\mathbf{X})$, the representative features tend to be activated are strengthened, and vice versa. On the condition that $w_m < 0$, when $\mathbf{K}_m > \mathrm{E}(\mathbf{X})$, the noisy features tend to be activated are weakened, and vice versa. We also visualize in Fig. 2(a) when $w_m \cdot \mathbf{K}_m > 0$, the feature is strengthened to represent the whole part of the cat. While the background feature is weakened to reduce noises above the cat part when $w_m \cdot \mathbf{K}_m < 0$, as shown in Fig. 2(b). Also, we observe in Fig. 3 that w_m in some layers of trained models are close to zero, showing that our proposed centering calibration can take advantage of both batch and instance statistics through training.

3.2.3 Scaling Calibration

Unlike the centering operation that determines the features to be kept after the activation, the scaling operation only changes the feature intensity, when ignoring the effect of

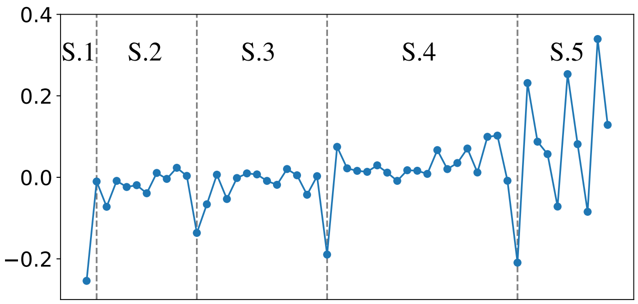

Figure 3. Averaged weight w_m in centering calibration among layers of ResNet-50 model.

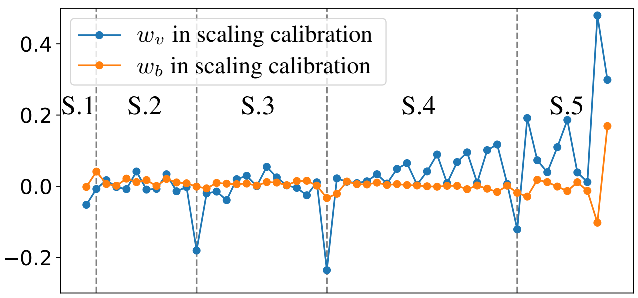

Figure 4. Averaged weights w_v, w_b in scaling calibration among layers of ResNet-50 model.

the affine transformation. The scaling operation scales features to have unit-variance with the running variance values. However, scaling features with the inaccurate running variance causes unstable feature intensity, *i.e.,* features from certain channels are much larger than those of other channels. We propose the scaling calibration to calibrate the feature intensity based on instance statistics.

Formulation of Scaling Calibration. We add the scaling calibration after the original scaling operations. Given the input feature $\mathbf{X}_s$, the calibrated feature is written as follows:

$$\mathbf{X}_{cs(n,c,h,w)} = \mathbf{X}_{s(n,c,h,w)} \cdot \mathbf{R}(w_v \odot \mathbf{K}_s + w_b), \tag{9}$$

where $w_v, w_b \in \mathbb{R}^{1\times C\times 1\times 1}$ are learnable weight vectors, and $\mathbf{R}()$ is the restricted function, which can be defined with multiple forms. In this work, we choose to use the Sigmoid function to suppress extreme values. Similar to $\mathbf{K}_m$, $\mathbf{K}_s$ is the statistics of the instance feature $\mathbf{X}_s$ that can be set to multiple values, as shown in Tab. 3.

Mechanism Proof. The restricted function $\mathbf{R}()$ along with the w_v and w_b in Eqn. (9) suppresses out-of-distribution features, making the feature distribution more stable. Since both the scaling operation and scaling calibration would not change the sign of features, and the negative features will be deactivated after the activation layer, we only consider the part where features are positive. According to Eqn. (9), the variance of features after the scaling calibration is written as:

$$\mathrm{Var}(\mathbf{X}_{cs}) = \mathrm{Var}\left(\mathbf{X}_s \cdot \mathbf{R}(w_v \cdot \mathbf{K}_s + w_b)\right). \tag{10}$$

Since the restricted function $0 < \mathbf{R}() < 1$, there must exist a τ whose value meets $\mathbf{R}() < \tau < 1$. Therefore, the $\mathrm{Var}(\mathbf{X}_{cs})$ can be relaxed to:

$$\mathrm{Var}(\mathbf{X}_{cs}) < \mathrm{Var}(\mathbf{X}_s\tau) = \tau^2\,\mathrm{Var}(\mathbf{X}_s). \tag{11}$$

After the scaling calibration, the feature variance is restricted to be smaller. Weights w_v and w_b control the strength and location of the restriction, respectively. And Fig. 5 shows that channels with a large variance of feature mean μ_c are restricted to a smaller variance by the scaling calibration according to w_v and w_b. The variance values of different channels in Fig. 5 are smaller, resulting in a more stable distribution among channels. $\mathrm{Var}(\mathbf{X}_{cs})$ is smaller when w_v becomes smaller, and w_b learns to adjust the position to be restricted. We observe from Fig. 4 that $w_v \le 1$ in trained models. $w_v \le 1$ tends to make features fall into the unsaturated region of $\mathbf{R}()$. We visualize in Fig. 2(c) and (d) that scaling calibration accordingly restrict features to avoid overlarge values.

3.2.4 Implementation of Representative BatchNorm

Given the input feature $\mathbf{X} \in \mathbb{R}^{N\times C\times H\times W}$, the formulation of the Representative BatchNorm (RBN) is written as:

$$\begin{aligned}
&\text{Centering Calibration}: \mathbf{X}_{cm} = \mathbf{X} + w_m \odot \mathbf{K}_m,\\
&\text{Centering}: \mathbf{X}_m = \mathbf{X}_{cm} - \mathrm{E}(\mathbf{X}_{cm}),\\
&\text{Scaling}: \mathbf{X}_s = \frac{\mathbf{X}_m}{\sqrt{\mathrm{Var}(\mathbf{X}_{cm}) + \epsilon}},\\
&\text{Scaling Calibration}: \mathbf{X}_{cs} = \mathbf{X}_s \cdot \mathbf{R}(w_v \odot \mathbf{K}_s + w_b),\\
&\text{Affine}: \mathbf{Y} = \mathbf{X}_{cs}\gamma + \beta.
\end{aligned} \tag{12}$$

To utilize the optimization of BatchNorm in existing deep learning frameworks, we add the centering and scaling calibrations at the beginning and ending of the original normalization layer of BatchNorm, respectively.

4. Experiments

4.1. Implementation Details

In this section, we report the implementation details of our experiments. We implement our method using the PyTorch [43], MindSpore [1], and Jittor [21] frameworks. On the ImageNet [10] dataset, we follow common settings [13, 19, 64] to randomly crop images to 224×224 pixels from a resized image and utilize the same basic data argumentation strategies are used in [13, 19, 64] for training. When training large models such as ResNet [19], ResNeXt [64], and Res2Net [13], we use the SGD optimizer to train the model for 100 epochs, and set weight decay to $1e^{-4}$, momentum to 0.9, and mini-batch to 256. The learning rate is initially set to 0.1, and divided by 10 every 30

ImageNet	Norm.	top-1 err.	top-5 err.
ResNet-50 [19]	IN [59]	28.40	-
	GN [63]	24.33	7.30
	BN [24]	23.85	7.13
	ILM [25]	23.57	
	SN [36]	23.10	-
	BCN [46]	23.09	6.55
	IEBN [32]	22.90	-
	FRN [53]	22.79	6.43
	RBN	22.64	6.54
	RBN*	**22.40**	**6.27**
ResNet-101 [19]	BN [24]	22.63	6.44
	RBN	**21.48**	**5.74**
ResNeXt-50 [64]	BN [24]	22.38	6.30
	RBN	**21.97**	**6.13**
Res2Net-50 [13]	BN [24]	22.01	6.15
	RBN	**21.16**	**5.88**
MobileNet v2 [50]	BN [24]	28.03	9.19
	RBN	**26.23**	**8.37**

Table 1. Classification performance of utilizing RBN on multiple network architectures on the ImageNet dataset. RBN* indicates set $\mathbf{K}_s = \sigma_c$ in Eqn. (9) instead of the faster version $\mathbf{K}_s = \mu_c$ used in RBN.

epochs. For the lightweight model MobileNet v2 [50], we train the model for 200 epochs using a Cosine learning rate scheduler with 0.05 initial learning rate, 256 mini-batch, $4e^{-5}$ weight decay, and 5 epochs warm-up.

To make sure that it is the role of calibration instead of initialization. We need to make the centering and scaling calibrations play no role at the beginning of the training. Therefore, the centering calibration weights w_m are all initialized with zero, and the scaling calibration weights w_v, w_b are initialized with zero and one, respectively. By default, μ_c of $\mathbf{X}$ and μ_c of $\mathbf{X}_s$ are utilized as $\mathbf{K}_m$ in Eqn. (5) and $\mathbf{K}_s$ in Eqn. (9), respectively, for the high computational efficiency.

4.2. Performance Evaluation and Ablation

In this section, we verify the effectiveness of our proposed RBN on various networks. We also conduct ablations to have a more comprehensive understanding of RBN.

Cooperating with Multiple Networks. As the variant of the original BatchNorm (BN), our proposed RBN can replace BN layers in networks to achieve better classification performance on the ImageNet dataset, as shown in Tab. 1. RBN based ResNet-50 surpasses the BN based ResNet-50 with 1.21% on top-1 err. On the deeper model ResNet-101, the RBN still outperforms BN by 1.15%, showing its robustness over the convergence difficulty of deeper networks. When cooperating with more advanced networks ResNeXt [64] and Res2Net [13], RBN based models surpass their baselines with 0.41% and 0.85% improvement, respectively. We also verify the effectiveness of RBN on the lightweight model. MobileNet v2 [50] equipped with RBN has an improvement of 1.8% over the baseline.

ImageNet	MobileNet v2	ResNet-50
BN	28.03	23.85
BN + Scaling cal.	26.96	23.06
BN + Centering cal.	26.55	22.85
RBN	**26.23**	**22.64**

Table 2. Effectiveness ablation (Top-1 err.) of the centering and scaling calibrations using multiple networks on the ImageNet dataset.

C cal.	-	μ_c	μ_c	μ_c	μ_c	μ_s	σ_c	σ_s
S cal.	-	μ_c	μ_s	σ_c	σ_s	μ_c	μ_c	μ_c
Err.	23.85	22.64	22.58	**22.40**	22.65	22.93	22.76	22.90

Table 3. Ablation (Top-1 err.) of using different statistics for calibrations in Eqn. (5) and Eqn. (9) using ResNet-50 on the ImageNet dataset. C cal. and S cal. represent centering calibration and scaling calibration, respectively.

Comparison with Normalization Methods. We also compare RBN with existing variants of BatchNorm, as shown in Tab. 1. GN [63] abandons the batch statistics, making it worse than BN in the commonly used training configuration. RBN maintains the benefits of batch dependency, and introduces the instance statistics to improve the representation ability of the model stably. Also, RBN performs better than other state-of-the-art normalization methods such as SN [36], BCN [46], ILM [25], IEBN [32], and FRN [53]. These normalization methods do not involve the centering and scaling calibrations, thus they can cooperate with RBN. We will conduct these experiments in our extended work.

Ablation on Centering and Scaling Calibrations. We verify the effectiveness of the centering and scaling calibrations in Tab. 2. We conduct ablations on the large-scale ImageNet dataset using the ResNet-50 and the lightweight model MobileNet v2. On MobileNet v2, the scaling and centering calibration improve the performance by 1.07% and 1.48% over the baseline, respectively. On ResNet-50, the performance gain brought by the scaling and centering calibration is 0.79% and 1.0% over the baseline, respectively. Combining the scaling and centering calibration, the performance is further improved. On the large-scale dataset, the centering calibration plays a slightly more important role than scaling operation.

Choice of Instance Statistics. Because the mean value of channels μ_c is computationally efficient and achieves decent performance, we use μ_c statistics in both Eqn. (5) and Eqn. (9) by default. We also show in Tab. 3 the effectiveness of other statistics such as the standard division of channels σ_c, the mean and standard division over spatial dimensions, denoted by μ_s and σ_s, respectively. Using μ_c and σ_c in centering and scaling calibrations achieves the best performance with more computational cost than only using μ_c. We observe that using μ_c in centering calibration is the best choice. Since scaling calibration only restricts the feature intensity while not changing the amount of information, scaling with both channel and spatial statistics results in a similar performance.

Observations. We first study the effect of RBN on different positions in the network. In Tab. 4, we show that adding the RBN to the early and last stages achieves better performance than the middle stages. We assume that the early stages are more dependent on the input instance, and the last stages are more related to the semantic meanings of the instance. We also visualize the averaged weight w_m in centering calibration, and the averaged weights w_v, w_b in scaling calibration.

As shown in Fig. 3, w_m in most layers are close to zero, indicating that the batch statistics still play an important role in most layers. The $|w_m|$ in the first layer of each stage is usually larger than other layers. We assume that the resolution changing of features makes the batch dependency unstable, requiring the feature calibration to strengthen features and reduce noises. Also, the absolute value w_m in the last stage is larger as this stage has more instance-specific features. We visualize features before and after the centering calibration in Fig. 6. Some features are strengthened or weakened after calibration, while features from some channels remain unchanged as the mini-batch statistics still dominate these channels.

As shown in Fig. 4, $|w_v|$ in scaling calibration becomes larger when the network goes deeper. We assume that instance-specific semantics in deeper layers may make the feature distribution unstable, thus requires more feature scaling calibration. We also visualize the standard deviation statistics of μ_c in the testing set of channels before and after the scaling calibration in Fig. 5. The scaling calibration learns to accordingly restrict feature variance of different channels to be closer, making the feature distribution more stable among channels. Features before and after the scaling calibration are visualized in Fig. 6. Features after scaling calibration are restricted to have smaller intensities.

4.3. Generalization to Tasks

Our proposed RBN can replace the BN to boost the representation ability of models in many tasks [14–17, 57]. This section verifies the effectiveness of our proposed RBN

CIFAR	No	S.1	S.2	S.3	S.4	All
ResNet-50-RBN	77.10	78.40	77.70	78.14	79.26	79.99
ResNeXt-29-RBN	79.69	81.21	80.22	80.64	-	82.12

Table 4. Ablation (Top-1 acc.) of applying RBN on different stages of the network.

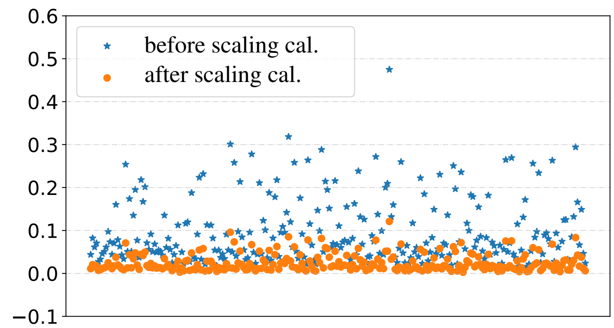

Figure 5. The standard deviation of μ_c in each channel before and after scaling calibration from layer 'layer3.2.bn2' in ResNet-50. Statistics are calculated in the testing set.

Object det.	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
ResNet-50-BN	37.8	58.0	41.3	21.8	41.0	49.3
ResNet-50-RBN	**39.3**	**59.7**	**42.8**	**22.7**	**43.4**	**51.8**
ResNet-101-BN	39.6	59.6	43.1	22.5	43.5	51.6
ResNet-101-RBN	**41.5**	**61.7**	**45.1**	**23.7**	**45.8**	**54.3**

Table 5. Performance of object detection on the COCO validation set using Faster-RCNN [48] with $\times 1$ lr schedule.

on down-stream tasks such as object detection, instance segmentation, and panoptic segmentation. By default, μ_c of $\mathbf{X}$ and μ_c of $\mathbf{X}_s$ are utilized as $\mathbf{K}_m$ in Eq.5 and $\mathbf{K}_s$ in Eq.9, respectively, for the high computational efficiency. All models utilizing RBN and original BN are trained with the same configuration.

Object Detection. For the object detection task, we verify the proposed method on the MS COCO [33] dataset using Faster-RCNN [48] as the baseline. We replace all BN layers in Faster-RCNN with our proposed RBN. As shown in Tab. 5, the proposed RBN cooperating with ResNet-50 outperforms its counterpart by 1.5% on average precision (AP) and 1.7% on AP@IoU=0.5. For ResNet-101, the RBN based model still outperforms the baseline by 1.9% on AP and 2.1% on AP@IoU=0.5. The RBN makes objectness features for object detection more representative, therefore improve the performance of the Faster-RCNN.

Instance Segmentation. Instance segmentation combines object detection and semantic segmentation. Correct objectness and accurate segmentation masks are both needed for this task. We validate instance segmentation on the MS COCO [33] dataset using Mask-RCNN [18] as

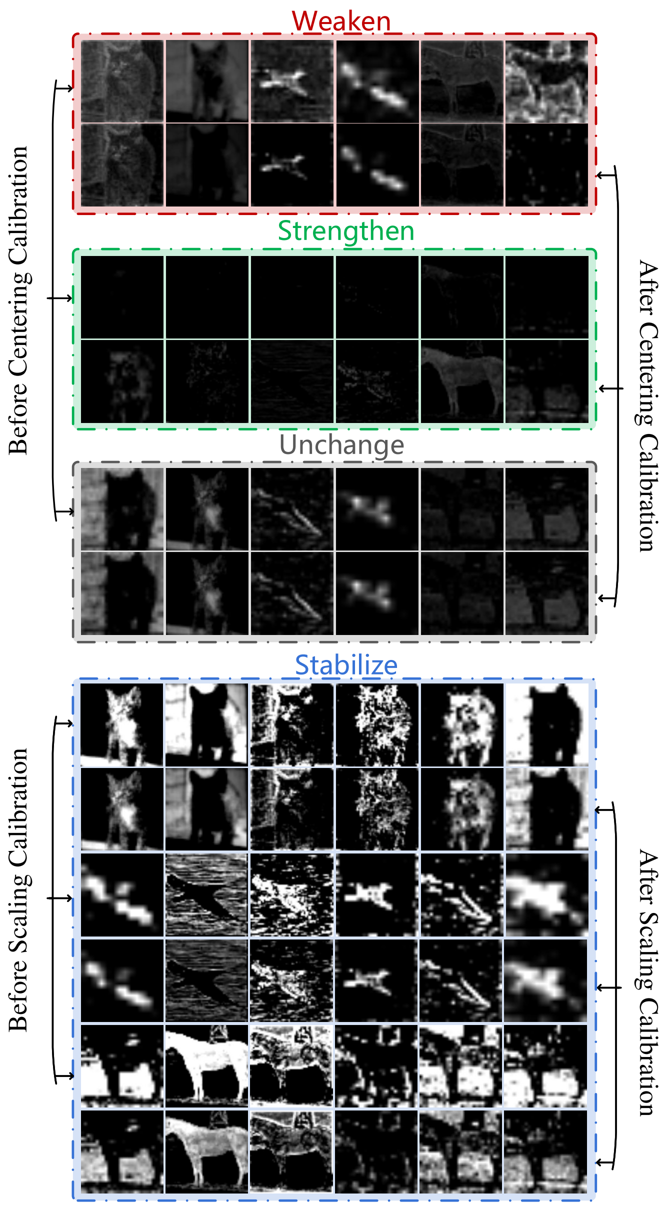

Figure 6. Visualization of features after calibration in RBN.

the baseline method. As shown in Tab. 6, on ResNet-50 based Mask-RCNN, replacing BN with our proposed RBN achieves 1.6% and 1.4% better performance on box AP and mask AP, respectively. Using ResNet-101, the performance gains brought by RBN are 1.2% on box AP and 1.2% on mask AP. The RBN consistently improves the representation ability of models for both box detection and mask segmentation.

Panoptic Segmentation. Panoptic segmentation is generalized from the instance segmentation and semantic segmentation. This task requires to segment all things at the instance level, and semantically segment all pixels of uncountable staff. We conduct panoptic segmentation on the

Box	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
ResNet-50-BN	38.7	58.6	42.4	22.4	42.0	50.2
ResNet-50-RBN	**40.3**	**60.4**	**43.9**	**23.2**	**44.3**	**52.7**
ResNet-101-BN	40.3	60.0	44.0	23.0	44.1	52.9
ResNet-101-RBN	**41.5**	**61.1**	**45.4**	**24.3**	**45.7**	**54.3**
Mask	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
ResNet-50-BN	34.7	55.4	37.1	18.5	37.8	46.7
ResNet-50-RBN	**36.1**	**57.5**	**38.3**	**19.4**	**39.7**	**48.9**
ResNet-101-BN	36.0	57.1	38.4	18.8	39.4	49.2
ResNet-101-RBN	**37.2**	**58.4**	**39.9**	**20.5**	**41.0**	**50.6**

Table 6. Performance of instance segmentation on the COCO validation set using Mask-RCNN [18] with $\times 1$ lr schedule.

	AP_{box}	AP_{mask}	SQ	RQ	PQ
ResNet-50-BN	33.3	31.0	75.6	44.8	36.3
ResNet-50-RBN	**35.0**	**32.3**	**76.0**	**45.9**	**37.2**
ResNet-101-BN	35.7	32.9	76.3	46.4	37.8
ResNet-101-RBN	**37.7**	**34.3**	**76.9**	**48.1**	**39.1**

Table 7. Performance of panoptic segmentation on the COCO validation set using Panoptic FPN [29] with $\times 1$ lr schedule.

MS COCO [33] dataset using Panoptic FPN [29]. As shown in Tab. 7, RBN cooperating with ResNet-50 architecture surpasses the baseline with 1.7% on AP_{box}, 1.3% on AP_{mask}, and 0.9% on panoptic quality (PQ). When cooperating with ResNet-101, replacing BN with our RBN achieves 2% on AP_{box}, 1.4% on AP_{mask}, and 1.3% on PQ, higher performance than the baseline. On the deeper network, RBN based model achieve more performance gain than on the shallow network, indicating that deeper panoptic segmentation models may benefit more from the stable and representative features introduced by RBN.

5. Conclusion

This paper proposes the Representative Batch Normalization (RBN) equipped with a simple yet effective feature calibration scheme to enhance the instance-specific representations and maintain the benefits of BatchNorm. The centering calibration strengthens informative features and weakens noisy features. The scaling calibration restricts the feature intensity to form a more stable feature distribution. RBN can be plugged into existing methods to boost the performance with negligible cost and parameters.

Acknowledgement This research was supported by NSFC (61922046, 61620106008) and S&T innovation project from Chinese Ministry of Education. We thank MindSpore [1] for the partial support of this work.

References

[1] Mindspore. http://www.mindspore.cn, 2020. 5, 8

[2] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016. 1, 2, 3

[3] Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale gan training for high fidelity natural image synthesis. In *Int. Conf. Learn. Represent.*, 2019. 3

[4] John Bronskill, Jonathan Gordon, James Requeima, Sebastian Nowozin, and Richard E Turner. Tasknorm: Rethinking batch normalization for meta-learning. In *Int. Conf. on Machine Learning Workshop*, 2020. 3

[5] Woong-Gi Chang, Tackgeun You, Seonguk Seo, Suha Kwak, and Bohyung Han. Domain-specific batch normalization for unsupervised domain adaptation. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 7354–7362, 2019. 3

[6] Vitaliy Chiley, Ilya Sharapov, Atli Kosson, Urs Koster, Ryan Reece, Sofia Samaniego de la Fuente, Vishal Subbiah, and Michael James. Online normalization for training neural networks. In *Adv. Neural Inform. Process. Syst.*, pages 8433–8443, 2019. 2

[7] Tim Cooijmans, Nicolas Ballas, César Laurent, Çağlar Gülçehre, and Aaron Courville. Recurrent batch normalization. In *Int. Conf. Learn. Represent.*, 2016. 1

[8] Harm De Vries, Florian Strub, Jérémie Mary, Hugo Larochelle, Olivier Pietquin, and Aaron C Courville. Modulating early visual processing by language. In *Adv. Neural Inform. Process. Syst.*, pages 6594–6604, 2017. 2, 3

[9] Lucas Deecke, Iain Murray, and Hakan Bilen. Mode normalization. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. 2

[10] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 248–255, 2009. 1, 5

[11] Neofytos Dimitriou and Ognjen Arandjelovic. A new look at ghost normalization. *arXiv preprint arXiv:2007.08554*, 2020. 2

[12] Vincent Dumoulin, Ishmael Belghazi, Ben Poole, Alex Lamb, Martín Arjovsky, Olivier Mastropietro, and Aaron C. Courville. Adversarially learned inference. In *Int. Conf. Learn. Represent.*, 2017. 2, 3

[13] Shang-Hua Gao, Ming-Ming Cheng, Kai Zhao, Xin-Yu Zhang, Ming-Hsuan Yang, and Philip Torr. Res2net: A new multi-scale backbone architecture. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1–1, 2021. 5, 6

[14] Shang-Hua Gao, Qi Han, Zhong-Yu Li, Pai Peng, Liang Wang, and Ming-Ming Cheng. Global2local: Efficient structure search for video action segmentation. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2021. 7

[15] Shang-Hua Gao, Yong-Qiang Tan, Ming-Ming Cheng, Chengze Lu, Yunpeng Chen, and Shuicheng Yan. Highly efficient salient object detection with 100k parameters. In *Eur. Conf. Comput. Vis.*, 2020. 7

[16] Yu-Chao Gu, Li-Juan Wang, Yun Liu, Yi Yang, Yu-Huan Wu, Shao-Ping Lu, and Ming-Ming Cheng. Dots: Decoupling operation and topology in differentiable architecture search. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2021. 7

[17] Qi Han, Kai Zhao, Jun Xu, and Ming-Ming Cheng. Deep hough transform for semantic line detection. In *Eur. Conf. Comput. Vis.*, 2020. 7

[18] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Int. Conf. Comput. Vis.*, pages 2961–2969, 2017. 1, 7, 8

[19] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 770–778, 2016. 1, 5, 6

[20] Elad Hoffer, Ron Banner, Itay Golan, and Daniel Soudry. Norm matters: efficient and accurate normalization schemes in deep networks. In *Adv. Neural Inform. Process. Syst.*, pages 2160–2170, 2018. 2

[21] Shi-Min Hu, Dun Liang, Guo-Ye Yang, Guo-Wei Yang, and Wen-Yang Zhou. Jittor: a novel deep learning framework with meta-operators and unified graph execution. *Science China Information Sciences*, 63(12):1–21, 2020. 5

[22] Lei Huang, Jie Qin, Yi Zhou, Fan Zhu, Li Liu, and Ling Shao. Normalization techniques in training dnns: Methodology, analysis and application. *arXiv preprint arXiv:2009.12836*, 2020. 2

[23] Lei Huang, Yi Zhou, Fan Zhu, Li Liu, and Ling Shao. Iterative normalization: Beyond standardization towards efficient whitening. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 4874–4883, 2019. 2

[24] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Int. Conf. on Machine Learning Workshop*, 2015. 1, 2, 3, 6

[25] Songhao Jia, Ding-Jie Chen, and Hwann-Tzong Chen. Instance-level meta normalization. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 4865–4873, 2019. 1, 3, 6

[26] Yongcheng Jing, Xiao Liu, Yukang Ding, Xinchao Wang, Errui Ding, Mingli Song, and Shilei Wen. Dynamic instance normalization for arbitrary style transfer. In *AAAI*, pages 4369–4376, 2020. 3

[27] Mahdi M Kalayeh and Mubarak Shah. Training faster by separating modes of variation in batch-normalized models. *IEEE Trans. Pattern Anal. Mach. Intell.*, 42(6):1483–1500, 2019. 2

[28] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 4401–4410, 2019. 3

[29] Alexander Kirillov, Ross Girshick, Kaiming He, and Piotr Dollár. Panoptic feature pyramid networks. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 6399–6408, 2019. 1, 8

[30] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *Adv. Neural Inform. Process. Syst.*, pages 1106–1114, 2012. 1

[31] Xilai Li, Wei Sun, and Tianfu Wu. Attentive normalization. In *Eur. Conf. Comput. Vis.*, 2020. 1, 3

[32] Senwei Liang, Zhongzhan Huang, Mingfu Liang, and Haizhao Yang. Instance enhancement batch normalization: An adaptive regulator of batch noise. In *AAAI*, pages 4819–4827, 2020. 1, 3, 6

[33] 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 *Eur. Conf. Comput. Vis.*, pages 740–755. Springer, 2014. 7, 8

[34] Hanxiao Liu, Andrew Brock, Karen Simonyan, and Quoc V Le. Evolving normalization-activation layers. In *Adv. Neural Inform. Process. Syst.*, 2020. 2

[35] Chunjie Luo, Jianfeng Zhan, Lei Wang, and Wanling Gao. Extended batch normalization. *arXiv preprint arXiv:2003.05569*, 2020. 2

[36] Ping Luo, Jiamin Ren, Zhanglin Peng, Ruimao Zhang, and Jingyu Li. Differentiable learning-to-normalize via switchable normalization. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. 1, 6

[37] Ping Luo, Ruimao Zhang, Jiamin Ren, Zhanglin Peng, and Jingyu Li. Switchable normalization for learning-to-normalize deep representation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 2019. 1, 2

[38] Ping Luo, Peng Zhanglin, Shao Wenqi, Zhang Ruimao, Ren Jiamin, and Wu Lingyun. Differentiable dynamic normalization for learning deep representation. In *Int. Conf. on Machine Learning Workshop*, pages 4203–4211, 2019. 2

[39] Vincent Michalski, Vikram Voleti, Samira Ebrahimi Kahou, Anthony Ortiz, Pascal Vincent, Chris Pal, and Doina Precup. An empirical study of batch normalization and group normalization in conditional computation. *arXiv preprint arXiv:1908.00061*, 2019. 1, 3

[40] Takeru Miyato and Masanori Koyama. cgans with projection discriminator. In *Int. Conf. Learn. Represent.*, 2018. 3

[41] Anthony Ortiz, Caleb Robinson, Dan Morris, Olac Fuentes, Christopher Kiekintveld, Md Mahmudulla Hassan, and Nebojsa Jojic. Local context normalization: Revisiting local normalization. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 11276–11285, 2020. 2

[42] Taesung Park, Ming-Yu Liu, Ting-Chun Wang, and Jun-Yan Zhu. Semantic image synthesis with spatially-adaptive normalization. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 2337–2346, 2019. 3

[43] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. In *Adv. Neural Inform. Process. Syst.*, pages 8026–8037, 2019. 5

[44] Chao Peng, Tete Xiao, Zeming Li, Yuning Jiang, Xiangyu Zhang, Kai Jia, Gang Yu, and Jian Sun. Megdet: A large mini-batch object detector. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 6181–6189, 2018. 2

[45] Ethan Perez, Harm De Vries, Florian Strub, Vincent Dumoulin, and Aaron Courville. Learning visual reasoning without strong priors. In *ICMLW*, 2017. 3

[46] Siyuan Qiao, Huiyu Wang, Chenxi Liu, Wei Shen, and Alan Yuille. Rethinking normalization and elimination singularity in neural networks. *arXiv preprint arXiv:1911.09738*, 2019. 2, 6

[47] Mengye Ren, Renjie Liao, Raquel Urtasun, Fabian H Sinz, and Richard S Zemel. Normalizing the normalizers: Comparing and extending network normalization schemes. In *Int. Conf. Learn. Represent.*, 2016. 2

[48] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Adv. Neural Inform. Process. Syst.*, pages 91–99, 2015. 1, 7

[49] Samuel Rota Bulò, Lorenzo Porzi, and Peter Kontschieder. In-place activated batchnorm for memory-optimized training of dnns. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 5639–5647, 2018. 2

[50] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 4510–4520, 2018. 1, 6

[51] Wenqi Shao, Tianjian Meng, Jingyu Li, Ruimao Zhang, Yudian Li, Xiaogang Wang, and Ping Luo. Ssn: Learning sparse switchable normalization via sparsestmax. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 443–451, 2019. 1, 2

[52] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *Int. Conf. Learn. Represent.*, 2015. 1

[53] Saurabh Singh and Shankar Krishnan. Filter response normalization layer: Eliminating batch dependence in the training of deep neural networks. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 11237–11246, 2020. 2, 6

[54] Saurabh Singh and Abhinav Shrivastava. Evalnorm: Estimating batch normalization statistics for evaluation. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 3633–3641, 2019. 1, 2

[55] Cecilia Summers and Michael J. Dinneen. Four things everyone should know to improve batch normalization. In *Int. Conf. Learn. Represent.*, 2020. 1, 2

[56] Wei Sun and Tianfu Wu. Image synthesis from reconfigurable layout and style. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 10531–10540, 2019. 3

[57] Yong-Qiang Tan, Shang-Hua Gao, Xuan-Yi Li, Ming-Ming Cheng, and Bo Ren. Vecroad: Point-based iterative graph exploration for road graphs extraction. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2020. 7

[58] Zhentao Tan, Dongdong Chen, Qi Chu, Menglei Chai, Jing Liao, Mingming He, Lu Yuan, and Nenghai Yu. Rethinking spatially-adaptive normalization. *arXiv preprint arXiv:2004.02867*, 2020. 3

[59] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Instance normalization: The missing ingredient for fast stylization. *arXiv preprint arXiv:1607.08022*, 2016. 1, 2, 3, 6

[60] Guangrun Wang, Ping Luo, Xinjiang Wang, Liang Lin, et al. Kalman normalization: Normalizing internal representations across network layers. In *Adv. Neural Inform. Process. Syst.*, pages 21–31, 2018. 2

[61] Ximei Wang, Ying Jin, Mingsheng Long, Jianmin Wang, and Michael I Jordan. Transferable normalization: Towards im-

proving transferability of deep neural networks. In *Adv. Neural Inform. Process. Syst.*, pages 1953–1963, 2019. 3

[62] Shuang Wu, Guoqi Li, Lei Deng, Liu Liu, Dong Wu, Yuan Xie, and Luping Shi. $l1$-norm batch normalization for efficient training of deep neural networks. *TNNLS*, 30(7):2043–2051, 2018. 2

[63] Yuxin Wu and Kaiming He. Group normalization. In *Eur. Conf. Comput. Vis.*, pages 3–19, 2018. 1, 2, 3, 6

[64] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 1492–1500, 2017. 5, 6

[65] Junjie Yan, Ruosi Wan, Xiangyu Zhang, Wei Zhang, Yichen Wei, and Jian Sun. Towards stabilizing batch statistics in backward propagation of batch normalization. In *Int. Conf. Learn. Represent.*, 2020. 1, 2

[66] Greg Yang, Jeffrey Pennington, Vinay Rao, Jascha Sohl-Dickstein, and Samuel S Schoenholz. A mean field theory of batch normalization. In *Int. Conf. Learn. Represent.*, 2019. 3

[67] Zhuliang Yao, Yue Cao, Shuxin Zheng, Gao Huang, and Stephen Lin. Cross-iteration batch normalization. *arXiv preprint arXiv:2002.05712*, 2020. 2

[68] Xiaoyong Yuan, Zheng Feng, Matthew Norton, and Xiaolin Li. Generalized batch normalization: Towards accelerating deep neural networks. In *AAAI*, volume 33, pages 1682–1689, 2019. 1, 2

[69] Peihao Zhu, Rameen Abdal, Yipeng Qin, and Peter Wonka. Sean: Image synthesis with semantic region-adaptive normalization. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 5104–5113, 2020. 2, 3