Momentum Batch Normalization for Deep Learning with Small Batch Size

Hongwei Yong[1,2], Jianqiang Huang[2], Deyu Meng[3,4], Xiansheng Hua[2], and Lei Zhang[1,2]

[1] Department of Computing, The Hong Kong Polytechnic University
{cshyong,cslzhang}@comp.polyu.edu.hk
[2] DAMO Academy, Alibaba Group
{jianqiang.jqh,huaxiansheng}@gmail.com
[3] Macau University of Science and Technology
[4] School of Mathematics and Statistics, Xi'an Jiaotong University
dymeng@mail.xjtu.edu.cn

Abstract. Normalization layers play an important role in deep network training. As one of the most popular normalization techniques, batch normalization (BN) has shown its effectiveness in accelerating the model training speed and improving model generalization capability. The success of BN has been explained from different views, such as reducing internal covariate shift, allowing the use of large learning rate, smoothing optimization landscape, etc. To make a deeper understanding of BN, in this work we prove that BN actually introduces a certain level of noise into the sample mean and variance during the training process, while the noise level depends only on the batch size. Such a noise generation mechanism of BN regularizes the training process, and we present an explicit regularizer formulation of BN. Since the regularization strength of BN is determined by the batch size, a small batch size may cause the under-fitting problem, resulting in a less effective model. To reduce the dependency of BN on batch size, we propose a momentum BN (MBN) scheme by averaging the mean and variance of current mini-batch with the historical means and variances. With a dynamic momentum parameter, we can automatically control the noise level in the training process. As a result, MBN works very well even when the batch size is very small (e.g., 2), which is hard to achieve by traditional BN.

Keywords: Batch normalization, small batch size, noise, momentum

1 Introduction

During the past decade, deep neural networks(DNNs) have achieved remarkable success in a variety of applications, such as image classification [14], object detection [31, 13], speech recognition [1] , natural language processing [28] and computer games [29, 36], etc. The success of DNNs comes from the advances in higher computing power (e.g., GPUs), large scale datasets [10], and learning algorithms [20, 37, 11]. In particular, advanced network architecture [14, 15]

and optimization techniques [20, 22] have been developed, making the training of very deep networks from a large amount of training data possible.

One of the key issues in DNN training is how to normalize the training data and intermediate features. It is well-known that normalizing the input data makes training faster [22]. The widely used batch normalization (BN) technique [19] naturally extends this idea to the intermediate layers within a deep network by normalizing the samples in a mini-batch during the training process. It has been validated that BN can accelerate the training speed, enable a bigger learning rate, and improve the model generalization accuracy [14, 15]. BN has been adopted as a basic unit in most of the popular network architectures such as ResNet [14] and DenseNet [15]. Though BN has achieved a great success in DNN training, how BN works remains not very clear. Researchers have tried to explain the underlying working mechanism of BN from different perspectives. For example, it is argued in [19] that BN can reduce internal covariate shift (ICS). However, it is indicated in [33] that there is no clear link between the performance gain of BN and the reduction of ICS. Instead, it is found that BN makes the landscape of the corresponding optimization problem smoother so that it allows larger learning rates, while stochastic gradient decent (SGD) with a larger learning rate could yield faster convergence along the flat direction of the optimization landscape so that it is less likely to get stuck in sharp minima [5].

Apart from better convergence speed, another advantage of BN is its regularization capability. Because the sample mean and variance are updated on mini-batches during training, their values are not accurate. Consequently, BN will introduce a certain amount of noise, whose function is similar to dropout. It will, however, increase the generalization capability of the trained model. This phenomenon has been empirically observed from some experimental results in [43, 44]. Teye et al. [38, 27] tried to give a theoretical explanation of the generalization gain of BN from a Bayesian perspective; however, it needs additional assumptions and priors, and the explanation is rather complex to understand.

In this paper, we present a simple noise generation model to clearly explain the regularization nature of BN. Our explanation only assumes that the training samples are independent and identically distributed (i.i.d.), which holds well for the randomly sampled mini-batches in the DNN training process. We prove that BN actually introduces a certain level of noise into the sample mean and variance, and the noise level only depends on the batch size. When the training batch size is small, the noise level becomes high, increasing the training difficulty. We consequently propose a momentum BN (MBN) scheme, which can automatically control the noise level in the training process. MBN can work stably for different mini-batch sizes, as validated in our experiments on benchmark datasets.

2 Related Work

Batch Normalization: BN [19] was introduced to address the internal covariate shift (ICS) problem by performing normalization along the batch dimension. For a layer with d-dimensional input $\mathbf{x} = (x^{(1)}, x^{(2)}, ..., x^{(d)})$ in a mini-batch $\mathcal{X}_B$

with size m, BN normalizes each dimension of the input samples as:

$$\widehat{x}^{(k)} = \frac{x^{(k)} - \mu_{\mathcal{B}}^k}{\sqrt{{\sigma_{\mathcal{B}}^k}^2 + \epsilon}} \tag{1}$$

where $\mu_{\mathcal{B}}^k = \frac{1}{m}\sum_{i=1}^m x_i^{(k)}$, $\sigma_{\mathcal{B}}^k = \frac{1}{m}\sum_{i=1}^N (x_i^{(k)} - \mu_{\mathcal{B}}^k)^2$, and ϵ is a small positive constant. And for inference step, the mean and variance of mini-batch are replaced with that of population, often estimated by moving average.

BN has achieved remarkable performance in terms of improving training speed and model generalization ability for many applications [14, 45]. In the case of small batch size, unfortunately, the sample mean and variance can be very different from those of the population. Consequently, BN may not perform well with a small batch size. To address this problem, batch renormalization (BReN) [18] was proposed by constraining the range of estimated mean and variance of a batch. However, it is very hard to tune the hyper-parameters in BReN, which limits its application to different tasks.

Other Normalization Methods: Besides BN, other normalization methods [23, 32, 39, 43, 3, 30] have been proposed to normalize data along other dimensions. For example, layer normalization (LN) [23] normalizes all activations or feature maps along feature dimension, instance normalization (IN) [39] performs normalization for each feature map of each sample and group normalization (GN) [43] normalizes feature maps for each input sample in a divided group. Although these methods depend less on training batch size, BN still outperforms them in many visual recognition tasks. Switchable normalization [26, 34, 25] uses a weighted combination of BN, LN and IN, which introduces more parameters and costs more computation. There are also some variants of BN, such as Batch Kalman Norm (BKN) [41], L1 BN [42], Decorrelated BN (DBN) [16], Riemannian BN [8], Iterative BN [17], Memorized BN [12] etc. Instead of operating on features, Weight Normalization (WN) [32] normalizes the filter weights. WN can also accelerate training but cannot outperform BN.

3 The Regularization Nature of BN

3.1 Noise Generation of BN

Several previous works [43, 44] have indicated that the BN layer can enhance the generalization capability of DNNs experimentally; however, little work has been done on the theoretical analysis about why BN has this capability. The only work we can find is [38], where Teye et al. tried to give a theoretical illustration for the generalization gain of BN from a Bayesian perspective with some additional priors. In the work [27], Luo et al. presented a regularization term based on the result of [38]. Shekhovtsov et al. [35] gave an interpretation of BN from the perspective of noise generation. However, it is assumed that the input activations follows strictly i.i.d. Gaussian distribution and there is no further theoretical analysis on how the noise affects the training process. In

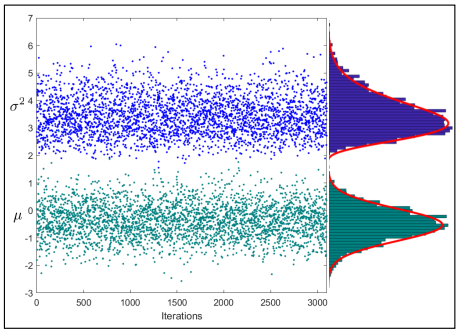


Fig. 1. The mean and variance of mini-batches vs. iterations. The mean (green points) and variance (blue points) are from one channel of the first BN layer of ResNet18 in the last epoch when training with batch size 16 on CIFAR100. The histograms of batch mean and variance are plotted on the right, which can be well fitted by Gaussian distribution and Chi-square distribution, respectively.

this section, we theoretically shows that BN can be modeled as process of noise generation.

Let's first assume that one activation input in a layer follows the Gaussian distribution $\mathcal{N}(x|\mu, \sigma^2)$, where μ and σ^2 can be simply estimated by population mean $\mu_{\mathcal{P}}$ and variance $\sigma_{\mathcal{P}}$ of training data. This assumption can be extended to more general cases other than Gaussian distribution, as we will explain later. In stochastic optimization [7, 6], randomly choosing a mini-batch of training samples can be considered as a sample drawing process, where all samples x_i in a mini-batch $\mathcal{X}_b = \{x_i\}_{i=1}^m$ are i.i.d., and follows $\mathcal{N}(x|\mu, \sigma^2)$. For the mini-batch $\mathcal{X}_b$ with mean $\mu_B = \frac{1}{m}\sum_{i=1}^m x_i$ and variance $\sigma_B^2 = \frac{1}{m}\sum_{i=1}^m (x_i - \mu_B)^2$, we can define two random variables ξ_μ and ξ_σ as follows [9]:

$$\xi_\mu = \frac{\mu - \mu_B}{\sigma} \sim \mathcal{N}(0, \frac{1}{m}), \quad \xi_\sigma = \frac{\sigma_B^2}{\sigma^2} \sim \frac{1}{m}\chi^2(m-1) \tag{2}$$

where χ^2 denotes the Chi-squared distribution and ξ_σ follows a Scaled-Chi-squared distribution with $E(\xi_\sigma) = \frac{m-1}{m}$ and $Var(\xi_\sigma) = \frac{2(m-1)}{m^2}$.

In Fig. 1 we plot the means and variances of mini-batches computed at the first BN layer of ResNet18 in the last training epoch when training with batch size 16 on CIFAR100 dataset. One can see that these means and variances are distributed like biased random noise. Specifically, the histogram of mean values can be well modeled as a Gaussian distribution, while the histogram of variances can be well modeled as a scaled Chi-Square distribution. By neglecting the small constant ϵ in Eq.(1), the BN in training process can be rewritten as

$$\widehat{x} = \frac{x - \mu_B}{\sigma_B} = \frac{x - \mu + (\mu - \mu_B)}{\sigma\frac{\sigma_B}{\sigma}} = \frac{\frac{x-\mu}{\sigma} + \xi_\mu}{\sqrt{\xi_\sigma}} = \frac{\widetilde{x} + \xi_\mu}{\sqrt{\xi_\sigma}} \tag{3}$$

where $\widetilde{x} = \frac{x-\mu}{\sigma}$ is the population normalized formula.

From Eq.(3), we can see that BN actually first adds Gaussian noise ξ_μ (**additive noise**) to the sample after population normalization, and then multiplies with a Scaled-Inverse-Chi noise $\frac{1}{\sqrt{\xi_\sigma}}$ (**multiplicative noise**). That is, training with BN is actually introducing a mixture of additive and multiplicative noise. With the introduced additive noise ξ_μ and multiplicative noise ξ_σ, the output variable $\widehat{x}$ follows $Nt(\widetilde{x}, m-1)$, which is a noncentral t-distribution [24], and its probability density function is very complex. Fortunately, we can still get the mean and variance of $\widehat{x}$ as follows:

$$E[\widehat{x}] = \widetilde{x}\sqrt{\frac{m-1}{2}}\frac{\Gamma((m-2)/2)}{\Gamma((m-1)/2)}, \quad Var[\widehat{x}] = \frac{1}{m}(\frac{m-1}{m-3}(1+\widetilde{x}^2) - E[\widehat{x}]^2) \quad (4)$$

When m is very large, $E[\widehat{x}] \approx \widetilde{x}$ and $Var[\widehat{x}] \approx 0$. However, when m is small, the noise generated by BN depends on not only the statistics of entire training data $\mathcal{X}$ (e.g., mean μ and variance σ^2) but also the batch size m.

With the above analyses, we can partition BN into three parts: a normalizer part (i.e., $\widetilde{x} = \frac{x-\mu}{\sigma}$); a noise generator part (i.e., $\widehat{x} = \frac{\widetilde{x}+\xi_\mu}{\sqrt{\xi_\sigma}}$) ; and an affine transformation part (i.e., $y = \gamma\widehat{x} + \beta$). In the training stage, only the noise generator part is related to batch size m. In the inference stage, the batch mean and variance are replaced with population mean and variance, and thus BN only has the normalizer part and the affine transformation part. It should be emphasized that μ and σ are unknown in training, and they also vary during the training process. At the end of training and when statistics for activations of all samples are stable, they can be viewed as fixed.

Now we have shown that the BN process actually introduces noises ξ_μ and ξ_σ into the BN layer in the training process. When the batch size is small, the variances of both additive noise ξ_μ and multiplicative noise ξ_σ become relatively large, making the training process less stable. In our above derivation, it is assumed that the activation input follows the Gaussian distribution. However, in practical applications the activations may not follow exactly the Gaussian distribution. Fortunately, we have the following theorem.

Theorem 1: *Suppose samples x_i for $i = 1, 2, ..., m$ are i.i.d. with $E[x] = \mu$ and $Var[x] = \sigma^2$, ξ_μ and ξ_σ are defined in Eq.(2), we have:*

$$\lim_{m\to\infty} p(\xi_\mu) \to \mathcal{N}(0, \frac{1}{m}), \quad \lim_{m\to\infty} p(\xi_\sigma) \to \frac{1}{m}\chi^2(m-1).$$

Theorem 1 can be easily proved by the central limit theorem. Please refer to the **Supplementary Materials** for the detailed proof. In particular, when m is larger than 5, ξ_μ and ξ_σ nearly meet the distribution assumptions. As for the i.i.d. assumption on the activations of samples in a mini-batch, it generally holds because the samples are randomly drawn from the pool in training.

3.2 Explicit regularization formulation

It has been verified in previous works [43, 44] that introducing a certain amount of noise into training data can increase the generalization capability of the neural

network. However, there lacks a solid theoretical analysis on how this noise injection operation works. In this section, we aim to give a clear formulation.

Additive Noise: We first take additive noise ξ_μ into consideration. Let $l(t, f(x))$ (abbreviate as $l(x)$ in the following development) denote the loss w.r.t. one activation input x, where t is the target, $f(\cdot)$ represents the network and $l(\cdot)$ is the loss function. When additive noise ξ_μ is added to the activation, the loss becomes $l(x + \xi_\mu)$. By Taylor expansion [4], we have

$$E_{\xi_\mu}[l(x+\xi_\mu)] = l(x) + R^{add}(x), \quad R^{add}(x) = \sum_{n=1}^{\infty} \frac{E[\xi_\mu^n]}{n!} \frac{d^n l(x)}{dx^n}. \tag{5}$$

where $E(\cdot)$ is the expectation and R^{add} is the additive noise residual term, which is related to the n-th order derivative of loss function w.r.t. activation input and the n-th order moment of noise distribution.

According to [2], by considering only the major term in R^{add}, it can be shown that $R^{add}(x) \approx \frac{E[\xi_\mu^2]}{2} \left|\frac{\partial f(x)}{\partial x}\right|^2$ for mean square-error loss; and $R^{add}(x) \approx \frac{E[\xi_\mu^2]}{2} \frac{f(x)^2 - 2tf(x) + t}{f(x)^2(1-f(x))^2} \left|\frac{\partial f(x)}{\partial x}\right|^2$ for cross-entropy loss. This indicates that R^{add} regularizes the smoothness of the network function, while the strength of smoothness is mainly controlled by the second order moment of the distribution of noise ξ_μ (i.e., $\frac{1}{m}$), which is only related to training batch size m. In Fig. 2, we illustrate the influence of additive noise on learning a classification hyperplane with different noise levels. The yellow points and blue points represent samples from two classes, and the Gaussian noise is added to the samples for data augmentation. By increasing the noise level σ to a proper level (e.g., $\sigma = 0.5$), the learned classification hyperplane becomes smoother and thus has better generalization capability. However, a too big noise level (e.g., $\sigma = 1$) will over-smooth the classification boundary and decrease the discrimination ability.

Multiplicative Noise: For multiplicative noise ξ_σ, we can use a simple logarithmic transformation $l(\frac{x}{\sqrt{\xi_\sigma}}) = l(e^{\log|x| - \frac{1}{2}\log\xi_\sigma}\text{sign}(x))$ to transform it into the form of additive noise. Then according to our analyses of additive noise:

$$E_{\xi_\sigma}[l(\frac{x}{\sqrt{\xi_\sigma}})] = l(x) + R^{mul}(x), \quad R^{mul}(x) = \sum_{n=1}^{\infty}\sum_{k=1}^{d} \mathrm{I}(x \neq 0) \frac{E[\log^n \xi_\sigma]}{(-2)^n n!} \frac{d^n l(x)}{(d\log|x|)^n}. \tag{6}$$

where $\mathrm{I}(x \neq 0)$ is an indicator function and $R^{mul}(x)$ is the residual term of Taylor expansion for multiplicative noise. Similar to the residual term of addictive noise $R^{add}(x)$, the major term of $R^{mul}(x)$ can also be viewed as a regularizer to $\left|\frac{\partial f(x)}{\partial \log|x|}\right|^2$, which controls the smoothness of network on log-scale, and $E[\log^2 \xi_\sigma]$ is related to the strength of the regularizer.

Compound Noise: In Section 3.1 we have shown that BN will introduce both addictive noise and multiplicative noise into the normalized activation input, i.e., $\widehat{x} = \frac{\widetilde{x}+\xi_\mu}{\sqrt{\xi_\sigma}}$. In the following theorem, we present the joint residual formulation for the compound of addictive noise and multiplicative noise.

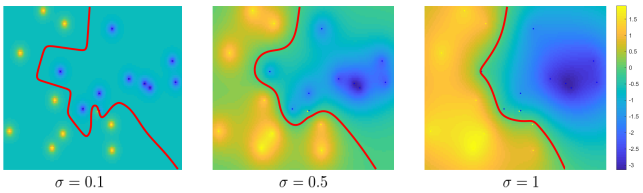


Fig. 2. The influence of noise injection on classification hyperplane (shown as red curve) learning with different noise levels. The yellow points and blue points represent samples from two classes, and the Gaussian noise with variance σ^2 is added to the samples for data augmentation. We can see that by increasing the noise level σ to a proper level (e.g., $\sigma = 0.5$), the learned classification hyperplane becomes smoother and thus has better generalization capability. However, a too big noise level (e.g., $\sigma = 1$) will over-smooth the classification boundary and decrease the discrimination ability.

Theorem 2: *If the infinite derivative of $l(x)$ exists for any x, given two random variables ξ_μ and ξ_σ (> 0), then we have the Taylor expansion for $l(\frac{x+\xi_\mu}{\sqrt{\xi_\sigma}})$:*

$$E_{\xi_\mu,\xi_\sigma}[l(\frac{x+\xi_\mu}{\sqrt{\xi_\sigma}})] = l(x) + R^{add}(x) + R^{mul}(x) + R(x),\ \ R(x) = \sum_{n=1}^{\infty} \frac{E[{\xi_\mu}^n]}{n!} \frac{d^n R^{mul}(x)}{dx^n} \tag{7}$$

where $R^{add}(x)$ and $R^{mul}(x)$ are defined in Eq.(5) and (6), respectively.

The proof of Theorem 2 can be found in the **Supplementary Material**. From Theorem 2, we can see the Taylor expansion residual can be divided into three parts: a residual term $R^{add}(x)$ for addictive noise , a residual term $R^{mul}(x)$ for multiplicative noise and a cross residual term $R(x)$. When the noise level is small, $R(x)$ can be ignored. Particularly, the distributions of ξ_μ and ξ_σ are give in Eq.(2) so that the regularizer strength parameters $E[\xi_\mu^2]$ and $E[\log^2 \xi_\sigma]$ can be easily calculated, which are only determined by training batch size m. The noise is injected into the normalized data $\widetilde{x} = \frac{x-\mu}{\sigma}$. If the introduced noise by BN is strong (e.g., when batch size is small), the training forward propagation through the DNN may accumulate and amplify noise, which leads to undesirable model performance. Therefore, it is crucial to choose a suitable batch size for training to make BN keep a proper noise level and ensure a favorable regularization function. However, in some situations of limited memory and computing resources, we can only use a small batch size for training. It is hence important to find an approach to control the noise level of BN with small batch size, which will be investigated in the next section.

4 Momentum Batch Normalization

As proved in Section 3, the batch size m directly controls the strength of the regularizer in BN so that BN is sensitive to batch size. In most previous literature [43, 18], the batch size m is set around 64 by experience. However, in some

applications the batch size may not be set big enough due to the limited memory and large size of input. How to stably train a network with small batch size in BN remains an open problem. Owe to our theoretical analyses in Section 3, we propose a simple solution to alleviate this problem by introducing a parameter to control the strength of regularizer in BN. Specifically, we replace the batch means and variances in BN by their momentum or moving average:

$$\mu_M^{(n)} = \lambda\mu_M^{(n-1)} + (1-\lambda)\mu_B, \quad \left(\sigma_M^{(n)}\right)^2 = \lambda\left(\sigma_M^{(n-1)}\right)^2 + (1-\lambda)\sigma_B^2, \tag{8}$$

where λ is the momentum parameter to controls the regularizer strength, and n refers to the number of batches (or iterations). We name our new BN method as Momentum Batch Normalization (MBN), which can make the noise level generated by using a small batch size almost the same as that by using a large batch size when the training stage ends.

4.1 Noise Estimation

At the end of the training process, all statistics of variables tend to be converged. According to Eq. (8), it can be derived that

$$\mu_M^{(n)} = (1-\lambda)\sum_{i=1}^{n}\lambda^{n-i}\mu_B, \quad \left(\sigma_M^{(n)}\right)^2 = (1-\lambda)\sum_{i=1}^{n}\lambda^{n-i}\sigma_B^2 \tag{9}$$

When n is very large, let μ_M and σ_M denote the final momentum mean and variance, we can derive that

$$\xi_\mu = \frac{\mu - \mu_M}{\sigma} \sim \mathcal{N}(0, \frac{1-\lambda}{m}) \tag{10}$$

$\xi_\sigma = \frac{\sigma_M^2}{\sigma^2}$ follows Generalized-Chi-Squared distribution, whose expectation is $E[\xi_\sigma] = \frac{m-1}{m}$ and variance is $Var[\xi_\sigma] = \frac{1-\lambda}{1+\lambda}\frac{2(m-1)}{m^2}$.

We can see that the variances of ξ_μ and ξ_σ approach to zero when λ is close to 1, MBN degenerates into standard BN when λ is zero. This implies that the noise level can be controlled by momentum parameter λ. A larger value λ will weaken the regularization function of MBN, and vice versa. Even when the batch size m is very small, we are still able to reduce the noise level through adjusting λ. This is an important advantage of MBN over conventional BN. For instance, if we want to make MBN with batch size 4 have similar noise level with batch size 16, the momentum parameter λ can be set as 3/4 to make their variances of ξ_μ similar, ($\frac{1-\frac{3}{4}}{4} = \frac{1}{16}$), and the multiplicative noise ξ_σ will also be reduced.

4.2 Momentum Parameter Setting

Dynamic Momentum Parameter for Training: Since the momentum parameter λ controls the final noise level, we need to set a proper momentum parameter to endow the network a certain generalization ability. Please note

Algorithm 1 [MBN] Momentum Batch Normalization

Input: Values of x over a training mini-batch $\mathcal{X}_b$; parameters γ, β; current training moving mean μ and variance σ^2; current inference moving mean μ_{inf} and variance σ^2_{inf}; momentum parameters λ for training and τ for inference.

Output: $\{y_i = \mathrm{MBN}(x_i)\}$; updated μ and σ^2; updated μ_{inf} and σ^2_{inf}

Training step:

$$\mu_B = \frac{1}{m}\sum_{i=1}^{m} x_i$$
$$\sigma_B^2 = \frac{1}{m}\sum_{i=1}^{m} (x_i - \mu_b)^2$$
$$\mu \leftarrow \lambda\mu + (1-\lambda)\mu_B$$
$$\sigma^2 \leftarrow \lambda\sigma^2 + (1-\lambda)\sigma_B^2$$
$$\widehat{x_i} = \frac{x_i - \mu}{\sqrt{\sigma^2+\epsilon}}$$
$$y_i = \gamma\widehat{x_i} + \beta$$
$$\mu_{inf} \leftarrow \tau\mu_{inf} + (1-\tau)\mu_B$$
$$\sigma^2_{inf} \leftarrow \tau\sigma^2_{inf} + (1-\tau)\sigma_B^2$$

Inference step: $y_i = \gamma\frac{x_i - \mu_{inf}}{\sqrt{\sigma^2_{inf}+\epsilon}} + \beta$

that our noise analysis in Section 4.1 holds only when network statistics are stable at the end of training. In the beginning of training, we cannot directly use the moving average of batch mean and variance, because the population mean and variance also change significantly. Therefore, we hope that in the beginning of training the normalization is close to BN, while at the end of it tends to be MBN. To this end, we propose a dynamic momentum parameter as follows:

$$\lambda^{(t)} = \rho^{\frac{T}{T-1}\max(T-t,0)} - \rho^T, \qquad \rho = \min(\frac{m}{m_0}, 1)^{\frac{1}{T}} \tag{11}$$

where t refers to the t-th iteration epoch, T is the number of the total epochs, m is the actual batch size and m_0 is a large enough batch size (e.g., 64).

We use the same momentum parameter within one epoch. $\lambda^{(t)}$ starts from zero. When $\frac{m}{m_0}$ is small, $\lambda^{(t)}$ tends to be a number close to 1 at the end of the training. If m is equal to or larger than m_0, $\lambda^{(t)}$ is always equal to zero, and then MBN degenerates into BN. The dynamic setting of momentum parameter ensures that at the beginning of training process, the normalization is similar to standard BN, while at the end of the training the normalization approaches to MBN with a noise level similar to that of BN with batch size m_0.

Momentum Parameter for Inference: For inference step, we also need to set a momentum parameter. For the clarity of description, here we use τ to denote this momentum parameter to differentiate it from the momentum parameter λ in the training stage. One can straightforwardly set τ as a constant, e.g. $\tau = 0.9$, which is independent of batch size. However, this setting is not very reasonable because it cannot reflect the final noise level when training is ended, which is related to batch size m. Therefore, we should set τ to be adaptive to batch size m. Denote by τ_0 the desired momentum value for an ideal batch size m_0, to make the inference momentum have the same influence on the last sample, we take $\tau^{\frac{N}{m}}$ as a reference to determine the value of τ for batch size m as follows:

$$\tau^{\frac{N}{m}} = \tau_0^{\frac{N}{m_0}} \quad \Rightarrow \quad \tau = \tau_0^{\frac{m}{m_0}} \tag{12}$$

where N is the number of samples, m_0 is an ideal batch size and τ_0 is its corresponding momentum parameter. In most of our experiments, we set $m_0 = 64$ and $\tau_0 = 0.9$ for the inference step. One can see that when the training batch size m is small, a larger inference momentum parameter τ will be used, and consequently the noise in momentum mean and variance will be suppressed.

4.3 Algorithm

The back-propagation (BP) process of MBN is similar to that of traditional BN. During training, the gradients of loss w.r.t. to activations and model parameters are calculated and back-propagated. The formulas of BP are listed as follows:

$$\begin{aligned}
&\frac{\partial L}{\partial \widehat{x}_i} = \frac{\partial L}{\partial \widehat{y}_i}\gamma, \quad \frac{\partial L}{\partial \gamma} = \sum_{i=1}^{m} \frac{\partial L}{\partial \widehat{y}_i}\widehat{x}_i, \quad \frac{\partial L}{\partial \beta} = \sum_{i=1}^{m} \frac{\partial L}{\partial \widehat{y}_i} \\
&\frac{\partial L}{\partial {\sigma_B}^2} = \sum_{i=1}^{m} \frac{\partial L}{\partial \widehat{x}_i}(\widehat{x}_i - \mu_M)\frac{-1}{2}({\sigma_M}^2 + \epsilon)^{-\frac{3}{2}}(1-\lambda) \\
&\frac{\partial L}{\partial \mu_B} = (\sum_{i=1}^{m} \frac{\partial L}{\partial \widehat{x}_i}\frac{\lambda - 1}{\sqrt{{\sigma_M}^2 + \epsilon}}) + \frac{\partial L}{\partial {\sigma_B}^2}\frac{\sum_{i=1}^{m} -2(x_i - \mu_B)}{m} \\
&\frac{\partial L}{\partial x_i} = \frac{\partial L}{\partial \widehat{x}_i}\frac{1}{\sqrt{{\sigma_M}^2 + \epsilon}} + \frac{\partial L}{\partial {\sigma_B}^2}\frac{2(x_i - \mu_B)}{m} + \frac{\partial L}{\mu_B}\frac{1}{m}
\end{aligned} \tag{13}$$

Since the current moving averages of μ_M and ${\sigma_M}^2$ are related to the mean μ_B and variance ${\sigma_B}^2$ of the current mini-batch, they also contribute to the gradient, while the previous μ_M and ${\sigma_M}^2$ can be viewed as two constants for the current mini-batch. The training and inference of MBN are summarized in Algorithm 1.

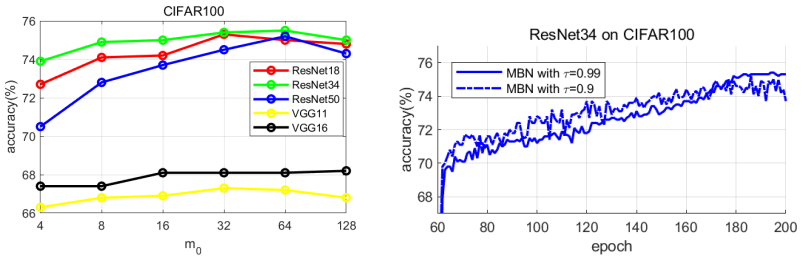
5 Experimental Results

5.1 Datasets and Experimental Setting

To evaluate MBN, we apply it to image classification tasks and conduct experiments on CIFAR10, CIFAR100 [21] and Mini-ImageNet100 datasets [40].

Datasets. CIFAR10 consists of 50k training images from 10 classes, while CIFAR100 consists of 50k training and 10k testing images from 100 classes. The resolution of sample images in CIFAR10/100 is 32×32. Mini-ImageNet is a subset of the well-known ImageNet dataset. It consists of 100 classes with 600 images each class, and the image resolution is 84×84. We use the first 500 images from each class as training data, and the rest 100 images for testing, i.e., 50k images for training and 10k images for testing.

Experimental setting. We use SGD with momentum 0.9 and weight decay 0.0001, employ standard data augmentation and preprocessing techniques, and nd decrease the learning rate when learning plateaus occur. The model is trained for 200 epochs and 100 epochs for CIFAR and Mini-ImageNet-100, respectively. We start with a learning rate of $0.1 * \frac{m}{64}$ both for CIFAR10 and CIFAR100



(a) Recognition rates of MBN with different m_0 on CIFAR100 by using different DNNs, including ResNet18, ResNet34, ResNet50, VGG11 and VGG16.

(b) Testing accuracy curves on CIFAR100 of BN and MBN with different τ for inference and training batch size 2 per GPU.

Fig. 3. Parameters tuning of MBN.

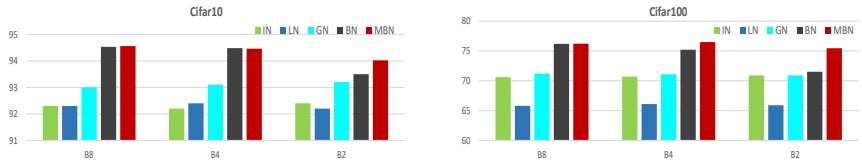
and $0.1 * \frac{m}{128}$ for Mini-ImageNet-100, and divide it by 10 for every 60 epochs and 30 epochs, respectively. We mainly employ ResNet [14] as our backbone network, and use similar experimental settings to the original ResNet paper. All the experiments are conducted on Pytorch1.0 framework.

5.2 Parameters setting

There are two hyper-parameters in our proposed MBN, m_0 and τ_0, which are used to determine the momentum parameters λ and τ for training and inference.

The setting for m_0: We first fix τ_0 (e.g., 0.9) to find a proper m_0. We adopt ResNet18 as the backbone and train it with batch size 8 and 16 on 4 GPUs, i.e., batch size 2 and 4 per GPU, to test the classification accuracy with different m_0. Particularly, we let m_0 be 4, 8, 16, 32, 64, 128 in MBN. Fig. 3(a) shows the accuracy curves on CIFAR100. We can see that if m_0 is too small (e.g., 4), MBN will be close to BN, and the performance is not very good. The accuracies of MBN are very close for m_0 from 16 to 128, which shows that MBN is not very sensitive to parameter m_0. Considering that if m_0 is too large (e.g., 128), the momentum parameter λ may change too quickly so that the training may not converge, we set it to 32 in all the experiments.

The setting for τ_0: We then fix m_0 as 32 and find a proper τ_0 based on Eq.(12). Fig. 3(b) shows the testing accuracy curves for for MBN with different values of τ. $\tau = 0.9$ is the original BN setting, and $\tau = 0.99$ is our setting based on Eq.(12) with $\tau_0 = 0.85$. We can see that when τ is small the testing accuracy curves of both BN and MBN have big fluctuations; while τ is large, the accuracy curves become more stable and the final accuracies can be improved. We set $\tau_0 = 0.85$ in the following experiments.



(a) Testing accuracy on Cifar10 of ResNet18

(b) Testing accuracy on CIFAR100 of ResNet18

Fig. 4. Testing accuracy on CIFAR10 and CIFAR100 of ResNet18 with training batch size (BS) 8, 4, and 2 per GPU.

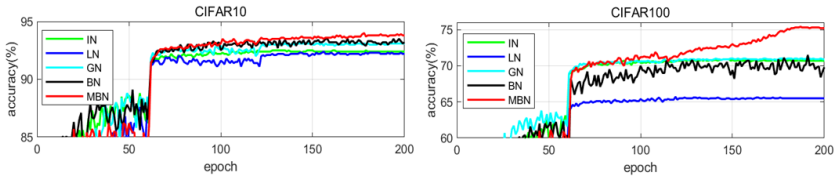


Fig. 5. Comparison of accuracy curves for different normalization methods with a batch size of 2 per GPU. We show the test accuracies vs. the epoches on CIFAR10 (left) and CIFAR100 (right). The ResNet18 is used.

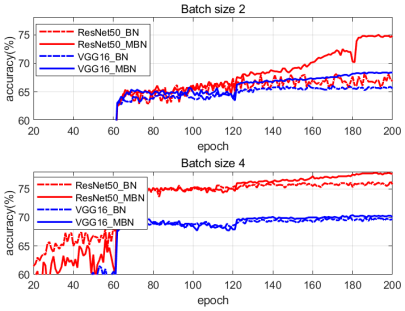
5.3 Results on CIFAR10/100

We first conduct experiments on the CIFAR10 and CIFAR100 datasets [21]. We first use ResNet18 as the backbone network to evaluate MBN with different batch sizes, and then test the performance of MBN with more networks.

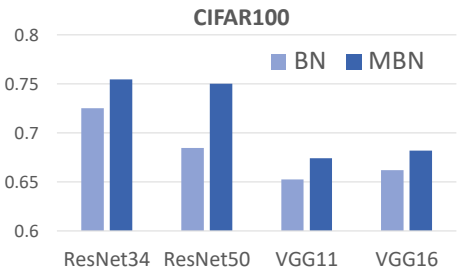
Training with Different Batch Size: To testify whether MBN is more robust than BN with small batch size, we train Resnet18 on CIFAR10 and CIFAR100 by setting the batch size m as 8, 4, 2 per GPU, respectively. We also compare the behaviors of other normalization methods, including IN [39], LN [23] and GN [43], by replacing the BN layer with them. For GN, we use 32 groups as set in [43]. And we set $T = 180$ in Eq.(11) for MBN.

Fig. 4 shows the results for different normalization methods. We can see that on both CIFAR10 and CIFAR100, when the batch size is relatively large (e.g., 8), the accuracy of MBN is similar to BN. This is in accordance to our theoretical analysis in Sections 3 and 4. However, when training batch size becomes small (e.g., 2), the accuracy of BN drops largely, while the accuracy of MBN decreases slightly. This shows that MBN is more robust than BN for training with small batch size. Meanwhile, MBN works much better than IN, LN and GN.

Fig. 5 shows the training and testing accuracy curves vs. epoch of ResNet18 with batch size 2. We can see that at the last stage of training when all statistics become stable, MBN can still achieve certain performance gain. This is because with MBN the momentum mean and variance approach to the population mean and variance, and hence the noise becomes small. Consequently, MBN can still keep improving though other methods are saturated.



(a) Testing accuracy curves on CIFAR100 for different network architectures with training batch size 2 (top) and 4 (bottom) per GPU.



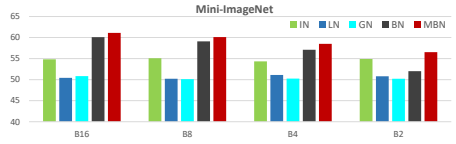
(b) Testing accuracy on CIFAR100 for different network architectures with training batch size 2 per GPU.

Fig. 6. Training with different DNN models with BN and MBN on CIFAR100.

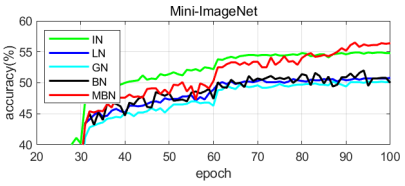
On More Network Architectures: We further test MBN with different network architectures, including ResNet34, ResNet50, VGG11 and VGG16, by using batch size 2 per GPU on CIFAR100. Fig. 6(a) shows the training and testing accuracy curves vs. epochs, and Fig. 6(b) shows the final testing accuracies. We can have the following observations. First, on all the four networks, MBN always outperforms BN. Second, under such a small batch size, the accuracy of deeper network ResNet50 can be lower than its shallower counterpart ResNet34. That is because the deeper networks have more BN layers, and each BN layer would introduce relatively large noise when batch size is small. The noise is accumulated so that the benefit of more layers can be diluted by the accumulated noise. However, with MBN the performance drop from ResNet50 to ResNet34 is very minor, where the drop by BN is significant. This again validates that MBN can suppress the noise effectively in training.

5.4 Results on Mini-ImageNet-100

On Small Batch Size: On Mini-imageNet, we use ResNet50 as our backbone network. The input size is the same as image size 84×84. The settings for MBN are the same as Section 5.3. Fig. 7(a) compares the testing accuracies of IN, LN, GN, BN and MBN with batch sizes 16, 8, 4 and 2 per GPU. Fig. 7(b) shows their testing accuracy curves with training batch size 2 per GPU. We can see that BN and MBN achieve better results than other normalization methods when batch size is larger than 2, while other normalization methods, such as IN, LN and GN, usually work not very well on Mini-imageNet. But the performance of BN drops significantly when batch size is 2, even worse that IN, while MBN still works well when batch size is 2. This clearly demonstrates the effectiveness of MBN. Furthermore, we also compare MBN with BN on full ImageNet using ResNet50 with 64 GPUs and 4 batch size per GPU. It is found that find MBN outperforms BN by 2.5% in accuracy on the validation set.



(a) Testing accuracy on Mini-ImageNet of IN, LN, GN, BN and MBN with training batch size 16, 8, 4 and 2 per GPU.



(b) Testing accuracy curves on Mini-ImageNet of IN, LN, GN, BN and MBN with training batch size 2 per GPU.

Fig. 7. Comparison of different normalization methods on Mini-ImageNet.

Comparison with BreN: BreN [18] was also proposed to make BN work for training with small batch size. It adopts a heuristic clipping strategy to control the influence of current moving average on the normalizer. Though BreN and our proposed MBN have similar goals, they are very different in theoretical development and methodology design. First, the dynamic momentum setting in MBN makes it easy to analyze the noise level in the final training stage, while in BreN it is hard to know the noise level with the heuristic clipping strategy. Second, the hyper-parameters m_0 and τ_0 are very easy to be tuned and fixed in MBN (we fixed them in all our experiments on all datasets), while the hyper-parameters (clipping bounds) in BreN are very difficult to set. Although a strategy to set the clipping bound was given in [18], we found that this setting usually leads unsatisfactory performance when the dataset or training batch size changes. We have tried various parameter settings for BreN on Mini-ImageNet when training batch size is 2, but found that in most cases the results are even worse. So we report the best result of BreN on Mini-ImageNet with $r_{max} = 1.5$ and $d_{max} = 0.5$, which is 55.47%, lower than the performance of MBN (56.50%)

6 Conclusion

Batch normalization (BN) is a milestone technique in deep learning and it largely improves the effectiveness and efficiency in optimizing various deep networks. However, the working mechanism of BN is not fully revealed yet, while the performance of BN drops much when the training batch size is small because of the inaccurate batch statistics estimation. In this work, we first revealed that the generalization capability of BN comes from its noise generation mechanism in training, and then presented the explicit regularization formulation of BN. We consequently presented an improved version of BN, namely momentum batch normalization (MBN), which uses the moving average of sample mean and variance in a mini-batch for training. By adjusting a dynamic momentum parameter, the noise level in the estimated mean and variance can be well controlled in MBN. The experimental results demonstrated that MBN can work stably for different batch sizes. In particular, it works much better than BN and other popular normalization methods when the batch size is small.

References

1. Amodei, D., Ananthanarayanan, S., Anubhai, R., Bai, J., Battenberg, E., Case, C., Casper, J., Catanzaro, B., Cheng, Q., Chen, G., et al.: Deep speech 2: End-to-end speech recognition in english and mandarin. In: International conference on machine learning. pp. 173–182 (2016)
2. An, G.: The effects of adding noise during backpropagation training on a generalization performance. Neural computation **8**(3), 643–674 (1996)
3. Arpit, D., Zhou, Y., Kota, B.U., Govindaraju, V.: Normalization propagation: A parametric technique for removing internal covariate shift in deep networks. arXiv preprint arXiv:1603.01431 (2016)
4. Bishop, C.M.: Training with noise is equivalent to tikhonov regularization. Neural Computation **7**(1), 108–116 (1995)
5. Bjorck, J., Gomes, C., Selman, B., Weinberger, K.Q.: Understanding batch normalization pp. 7694–7705 (2018)
6. Bottou, L.: Stochastic gradient learning in neural networks. Proceedings of Neuro-Nımes **91**(8), 12 (1991)
7. Bottou, L.: Large-scale machine learning with stochastic gradient descent. In: Proceedings of COMPSTAT'2010, pp. 177–186. Springer (2010)
8. Cho, M., Lee, J.: Riemannian approach to batch normalization. In: Advances in Neural Information Processing Systems. pp. 5225–5235 (2017)
9. Crocker, L., Algina, J.: Introduction to classical and modern test theory. ERIC (1986)
10. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: Imagenet: A large-scale hierarchical image database. In: 2009 IEEE conference on computer vision and pattern recognition. pp. 248–255. Ieee (2009)
11. Duchi, J., Hazan, E., Singer, Y.: Adaptive subgradient methods for online learning and stochastic optimization. Journal of Machine Learning Research **12**(Jul), 2121–2159 (2011)
12. Guo, Y., Wu, Q., Deng, C., Chen, J., Tan, M.: Double forward propagation for memorized batch normalization. In: Thirty-Second AAAI Conference on Artificial Intelligence (2018)
13. He, K., Gkioxari, G., Dollár, P., Girshick, R.: Mask r-cnn. In: Proceedings of the IEEE international conference on computer vision. pp. 2961–2969 (2017)
14. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 770–778 (2016)
15. Huang, G., Liu, Z., Van Der Maaten, L., Weinberger, K.Q.: Densely connected convolutional networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 4700–4708 (2017)
16. Huang, L., Yang, D., Lang, B., Deng, J.: Decorrelated batch normalization. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 791–800 (2018)
17. Huang, L., Zhou, Y., Zhu, F., Liu, L., Shao, L.: Iterative normalization: Beyond standardization towards efficient whitening. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 4874–4883 (2019)
18. Ioffe, S.: Batch renormalization: Towards reducing minibatch dependence in batch-normalized models. In: Advances in neural information processing systems. pp. 1945–1953 (2017)

19. Ioffe, S., Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. arXiv preprint arXiv:1502.03167 (2015)
20. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
21. Krizhevsky, A., Hinton, G., et al.: Learning multiple layers of features from tiny images. Tech. rep., Citeseer (2009)
22. LeCun, Y.A., Bottou, L., Orr, G.B., Müller, K.R.: Efficient backprop. In: Neural networks: Tricks of the trade, pp. 9–48. Springer (2012)
23. Lei Ba, J., Kiros, J.R., Hinton, G.E.: Layer normalization. arXiv preprint arXiv:1607.06450 (2016)
24. Lenth, R.V.: cumulative distribution function of the non-central t distribution. Journal of the Royal Statistical Society. Series C (Applied Statistics) **38(1)**, 185–189 (1989)
25. Luo, P., Peng, Z., Ren, J., Zhang, R.: Do normalization layers in a deep convnet really need to be distinct? arXiv preprint arXiv:1811.07727 (2018)
26. Luo, P., Ren, J., Peng, Z.: Differentiable learning-to-normalize via switchable normalization. arXiv preprint arXiv:1806.10779 (2018)
27. Luo, P., Wang, X., Shao, W., Peng, Z.: Towards understanding regularization in batch normalization (2018)
28. Luong, M.T., Pham, H., Manning, C.D.: Effective approaches to attention-based neural machine translation. arXiv preprint arXiv:1508.04025 (2015)
29. Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A.A., Veness, J., Bellemare, M.G., Graves, A., Riedmiller, M., Fidjeland, A.K., Ostrovski, G., et al.: Human-level control through deep reinforcement learning. Nature **518**(7540), 529 (2015)
30. Ren, M., Liao, R., Urtasun, R., Sinz, F.H., Zemel, R.S.: Normalizing the normalizers: Comparing and extending network normalization schemes. arXiv preprint arXiv:1611.04520 (2016)
31. Ren, S., He, K., Girshick, R., Sun, J.: Faster r-cnn: Towards real-time object detection with region proposal networks. In: Advances in neural information processing systems. pp. 91–99 (2015)
32. Salimans, T., Kingma, D.P.: Weight normalization: A simple reparameterization to accelerate training of deep neural networks. In: Advances in Neural Information Processing Systems. pp. 901–909 (2016)
33. Santurkar, S., Tsipras, D., Ilyas, A., Madry, A.: How does batch normalization help optimization? (no, it is not about internal covariate shift) pp. 2483–2493 (2018)
34. Shao, W., Meng, T., Li, J., Zhang, R., Li, Y., Wang, X., Luo, P.: Ssn: Learning sparse switchable normalization via sparsestmax. arXiv preprint arXiv:1903.03793 (2019)
35. Shekhovtsov, A., Flach, B.: Stochastic normalizations as bayesian learning. In: Asian Conference on Computer Vision. pp. 463–479. Springer (2018)
36. Silver, D., Huang, A., Maddison, C.J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., et al.: Mastering the game of go with deep neural networks and tree search. nature **529**(7587), 484 (2016)
37. Sutskever, I., Martens, J., Dahl, G., Hinton, G.: On the importance of initialization and momentum in deep learning. In: International conference on machine learning. pp. 1139–1147 (2013)
38. Teye, M., Azizpour, H., Smith, K.: Bayesian uncertainty estimation for batch normalized deep networks. arXiv preprint arXiv:1802.06455 (2018)
39. Ulyanov, D., Vedaldi, A., Lempitsky, V.: Instance normalization: The missing ingredient for fast stylization. arXiv preprint arXiv:1607.08022 (2016)

40. Vinyals, O., Blundell, C., Lillicrap, T., Wierstra, D., et al.: Matching networks for one shot learning. In: Advances in neural information processing systems. pp. 3630–3638 (2016)
41. Wang, G., Peng, J., Luo, P., Wang, X., Lin, L.: Batch kalman normalization: Towards training deep neural networks with micro-batches. arXiv preprint arXiv:1802.03133 (2018)
42. Wu, S., Li, G., Deng, L., Liu, L., Wu, D., Xie, Y., Shi, L.: L1-norm batch normalization for efficient training of deep neural networks. IEEE transactions on neural networks and learning systems (2018)
43. Wu, Y., He, K.: Group normalization. In: Proceedings of the European Conference on Computer Vision (ECCV). pp. 3–19 (2018)
44. Zhang, C., Bengio, S., Hardt, M., Recht, B., Vinyals, O.: Understanding deep learning requires rethinking generalization. arXiv preprint arXiv:1611.03530 (2016)
45. Zhang, K., Zuo, W., Chen, Y., Meng, D., Zhang, L.: Beyond a gaussian denoiser: Residual learning of deep cnn for image denoising. IEEE Transactions on Image Processing **26**(7), 3142–3155 (2017)