An Internal Covariate Shift Bounding Algorithm for Deep Neural Networks by Unitizing Layers' Outputs

You Huang
Fuzhou University
youhuang0607@gmail.com

Yuanlong Yu[*]
Fuzhou University
yu.yuanlong@fzu.edu.cn

Abstract

Batch Normalization (BN) techniques have been proposed to reduce the so-called Internal Covariate Shift (ICS) by attempting to keep the distributions of layer outputs unchanged. Experiments have shown their effectiveness on training deep neural networks. However, since only the first two moments are controlled in these BN techniques, it seems that a weak constraint is imposed on layer distributions and furthermore whether such constraint can reduce ICS is unknown. Thus this paper proposes a measure for ICS by using the Earth Mover (EM) distance and then derives the upper and lower bounds for the measure to provide a theoretical analysis of BN. The upper bound has shown that BN techniques can control ICS only for the outputs with low dimensions and small noise whereas their control is not effective in other cases. This paper also proves that such control is just a bounding of ICS rather than a reduction of ICS. Meanwhile, the analysis shows that the high-order moments and noise, which BN cannot control, have great impact on the lower bound. Based on such analysis, this paper furthermore proposes an algorithm that unitizes the outputs with an adjustable parameter to further bound ICS in order to cope with the problems of BN. The upper bound for the proposed unitization is noise-free and only dominated by the parameter. Thus, the parameter can be trained to tune the bound and further to control ICS. Besides, the unitization is embedded into the framework of BN to reduce the information loss. The experiments show that this proposed algorithm outperforms existing BN techniques on CIFAR-10, CIFAR-100 and ImageNet datasets.

1. Introduction

Deep neural networks (DNNs) show good performance in image recognition [18], speech recognition [13] and other fields [23, 32] in recent years. However, how to train DNNs is still a fundamental problem which is more complicated than training shallow networks because of deep architectures. It was commonly thought that stacking more layers suffers from the problem of vanishing or exploding gradients [7], but there are some problems of training with unclear definitions. A problem called *Internal Covariate Shift* (ICS) [15] may hinder the convergence of training DNNs.

ICS is derived from *Covariate Shift* (CS) that is caused by using data from two different distributions to respectively train and test a model, generally in the supervised learning process [28]. However, ICS mainly exists in the feed-forward networks. Considering the lth layer of a network with L layers, a stack of following $L-l$ layers forms a local network $f_{l+1:L}$, whose input is the output of the lth layer. Thus, the distribution of the input is affected by all the former l layers' weights. In detail, the objective function of $f_{l+1:L}$ is defined as

$$\mathcal{L}(\mathbf{\Theta}_{l+1:L}; p_l^{(t)}, p_{\boldsymbol{y}}) = \mathbb{E}_{\boldsymbol{x}\sim p_l^{(t)}, \boldsymbol{y}\sim p_{\boldsymbol{y}}(\cdot|\boldsymbol{x})}[h(\boldsymbol{x}, \boldsymbol{y}; \mathbf{\Theta}_{l+1:L})], \tag{1}$$

where $p_l^{(t)}$ is the distribution of the lth layer's outputs at the tth iteration; $p_{\boldsymbol{y}}$ is the conditional distribution of the ground truth at the last layer given $\boldsymbol{x}$; $\mathbf{\Theta}_{l+1:L}$ is the weights of $f_{l+1:L}$; $h(\boldsymbol{x}, \boldsymbol{y}; \mathbf{\Theta}_{l+1:L})$ is the loss for a sample pair $(\boldsymbol{x}, \boldsymbol{y})$. We use Back Propagation algorithm to train networks. However, the objective function $\mathcal{L}(\mathbf{\Theta}_{l+1:L}; p_l^{(t+1)}, p_{\boldsymbol{y}})$ at the $(t+1)$-th iteration would be different from the previous one as the distribution changes from $p_l^{(t)}$ to $p_l^{(t+1)}$. So using the gradients obtained from $\mathcal{L}(\mathbf{\Theta}_{l+1:L}; p_l^{(t)}, p_{\boldsymbol{y}})$ to update $\mathbf{\Theta}_{l+1:L}$ might not reduce $\mathcal{L}(\mathbf{\Theta}_{l+1:L}; p_l^{(t+1)}, p_{\boldsymbol{y}})$ due to the divergence between $p_l^{(t)}$ and $p_l^{(t+1)}$. Furthermore, the divergence will become much larger with increase of the number of network layers.

The technique called Batch Normalization (BN) [15] has been proposed to reduce ICS by attempting to make the distributions remain unchanged. In practice, BN normalizes the outputs to control the first two moments, *i.e.*, mean and variance, and uses two adjustable parameters to recover the information lost in normalizing outputs. Empirical results have shown that BN can speed up network training and also

[*]Corresponding author.

improve the success rate [11, 29]. However, whether BN can really reduce ICS is not clear in theory. It is obvious that the first issue is about how to measure the divergence. Furthermore, since BN techniques only control the first and second moments, the constraint imposed on distributions by BN is weak. So how to theoretically analyze the bounding of ICS imposed by BN techniques is the second issue.

Meanwhile, some experiments have shown that the performance gain of BN seems unrelated to the reduction of ICS [27]. In fact, ICS always exists when we train networks based on the fact that the gradient strategy must give the weight update in order to train the network such that the distribution of each layer varies. Furthermore, in the case of gradient vanishing, the ICS is totally eliminated. However, the network training cannot work. This case illustrates that very slight ICS cannot support effective training. Thus, it seems that controlling ICS instead of eliminating it is effective for training networks. So how to control ICS in order to improve network training is another challenge issue.

This paper proposed an ICS measure, *i.e.*, the divergence, by using the Earth Mover (EM) distance [33], inspired by the success of Wasserstein Generative Adversarial Networks (WGAN) [1]. Furthermore, this paper simplifies the measure by leveraging the Kantorovich-Rubinstein duality [33].

Based on this proposed ICS measure, this paper furthermore derives an upper bound of ICS between $p_l^{(t+\Delta t)}$ and $p_l^{(t)}$ at the $(t+\Delta t)$-th and tth iterations, respectively. The upper bound has shown that BN techniques can control the bounding of ICS in the low-dimensional case with small noise. Otherwise, the upper bound is out of control by using BN techniques. So it is required to analyze the lower bound of ICS especially for non-trivial distributions. Thus, this paper also derives a lower bound of ICS. The result has shown that the high-order moments and noise have great impact on the lower bound.

In order to control ICS, this paper proposes an algorithm that unitizes the normalized outputs. It is obvious that normalizing the outputs can introduce the moment-dependent upper bound, but such normalization would degrade when the moment estimation is not accurate. In contrast, unitizing the outputs in this proposed algorithm can lead to a constant upper bound without noise. However, by simply unitizing the outputs, the bound is very tight such that the weights cannot be updated in a reasonable range. Instead, this paper introduces a trainable parameter α in the unitization, such that the upper bound is adjustable and ICS can be further controlled by fine-tuning α. It is important that the proposed unitization is embedded into the BN framework in order to reduce the information loss. The experiments show that the proposed unitization can outperform existing BN techniques in the benchmark datasets including CIFAR-10, CIFAR-100 [17] and ImageNet [25].

2. Related work

Batch Normalization aims to reduce ICS by stabilizing the distributions of layers' outputs [15]. In fact, BN only controls the first two moments by normalizing the outputs, which is inspired by the idea of whitening the outputs to make the training faster [20]. However, BN is required to work with a sufficiently large batch size in order to reduce the noise of moments, and will degrade when the restriction on batch sizes is more demanding in some tasks [6, 9, 24]. Thus the methods including LN [2], IN [5] and GN [36] have been proposed. These variants estimate the moments within each sample, mitigating the impact of micro-batch. Besides, Kalman Normalization (KN) addresses this problem by the merits of Kalman Filtering [34], and a method called 'EvalNorm' is proposed to estimate the moments more accurately for BN during inference [31].

Other methods inspired by BN have been proposed to improve network training. Weight Normalization decouples the length of the weight from the direction by reparameterizing the weights, and speeds up convergence of the training [26]. Cho and Lee regard the weight space in a BN layer as a Riemannian manifold and provide a new learning rule following the intrinsic geometry of this manifold [4]. Cosine Normalization uses cosine similarity and bounds the results of dot product, addressing the problem of the large variance [22]. Wu *et al*. propose the algorithm normalizing the layers' inputs with l_1 norm to reduce computation and memory [35]. Huang *et al*. propose Decorrelated Batch Normalization that whitens instead of normalizing the activations [14].

However, there is no complete analysis in theory for BN. Santurkar *et al*. attempt to demonstrate that the performance gain of BN is unrelated to the reduction of ICS by experiments [27]. However, the first experiment only shows that BN can improve network training by other ways. In the second experiment, the difference between gradients is an unsuitable ICS measure since the gradients are sensitive and the accurate estimations require sufficient samples. Kohler *et al*. provide the theoretical analysis of BN [16], but it requires strong assumptions. In addition, Cai *et al*. focus on ordinary least squares regression and analyze the effects of gradient descent with BN in stability and convergence [3]. On the other hand, Yang *et al*. find that the maximum trainable depth of networks with BN is limited due to the gradient explosion caused by BN [37]. In general, the reason why BN works still remains unclear.

3. Unitization

The EM distance requires weak assumptions, and has been empirically proved to be effective in improving Generative Adversarial Networks (GANs) [8], which replaces the traditional KL-divergence in making the objective function [1]. According to the EM distance, the ICS measure for the lth layer's outputs is defined as

$$W(p_l^{(t+\Delta t)}, p_l^{(t)}) = \inf_{\gamma \in \prod(p_l^{(t+\Delta t)}, p_l^{(t)})} \mathbb{E}_{(\boldsymbol{x},\boldsymbol{y})\sim\gamma}\big[||\boldsymbol{x}-\boldsymbol{y}||\big], \tag{2}$$

where $\prod(p_l^{(t+\Delta t)}, p_l^{(t)})$ denotes the set of all joint distributions whose marginals are $p_l^{(t+\Delta t)}$ and $p_l^{(t)}$, respectively [1]. Then, by the Kantorovich-Rubinstein duality [33], the EM distance Eq. (2) can be rewritten as

$$W(p_l^{(t+\Delta t)}, p_l^{(t)}) = \sup_{||f||_L \le 1} \mathbb{E}_{\boldsymbol{x}\sim p_l^{(t+\Delta t)}}[f(\boldsymbol{x})] - \mathbb{E}_{\boldsymbol{y}\sim p_l^{(t)}}[f(\boldsymbol{y})], \tag{3}$$

where the distance is obtained by optimizing f over the 1-Lipschitz function space (see the algorithm that estimates the EM distance in the supplementary material).

3.1. The Upper Bound

For d-dimensional outputs of the lth layer, denote by $\boldsymbol{\mu}^{(t)} = (\mu_1^{(t)}, \mu_2^{(t)}, \ldots, \mu_d^{(t)})$ and $(\boldsymbol{\sigma}^{(t)})^2 = ((\sigma_1^{(t)})^2, (\sigma_2^{(t)})^2, \ldots, (\sigma_d^{(t)})^2)$ the mean and variance of the distribution $p_l^{(t)}$, respectively. The upper bound over $W(p_l^{(t+\Delta t)}, p_l^{(t)})$ is formed by the first two moments (see the proofs of all the theorems in the supplementary material).

Theorem 1. *Suppose that* $|\mu_i^{(t)}| < \infty, |\mu_i^{(t+\Delta t)}| < \infty, 1 \le i \le d$. *Then,*

$$\begin{aligned} W(p_l^{(t+\Delta t)}, p_l^{(t)}) \le & \sum_{i=1}^d (\sigma_i^{(t+\Delta t)})^2 + \sum_{i=1}^d (\sigma_i^{(t)})^2 \\ & + \Big(\sum_{i=1}^d (\mu_i^{(t+\Delta t)} - \mu_i^{(t)})^2\Big)^{\frac{1}{2}} + 2. \end{aligned} \tag{4}$$

In BN, the output is normalized by the estimated mean $\hat{\mu_i}$ and standard deviation $\hat{\sigma_i}$. Thus, for the normalized output, assume that $\mu_i^{(t)} = \epsilon_{\mu,i}^{(t)}, (\sigma_i^{(t)})^2 = 1 + \epsilon_{\sigma^2,i}^{(t)}, 1 \le i \le d$, where $\epsilon_{\mu,i}^{(t)}, \epsilon_{\sigma^2,i}^{(t)}, 1 \le i \le d$ are noise. According to the above theorem, the upper bound is

$$\begin{aligned} W(p_l^{(t+\Delta t)}, p_l^{(t)}) \le & \sum_{i=1}^d \epsilon_{\sigma^2,i}^{(t+\Delta t)} + \sum_{i=1}^d \epsilon_{\sigma^2,i}^{(t)} + 2d \\ & + \Big(\sum_{i=1}^d (\epsilon_{\mu,i}^{(t+\Delta t)} - \epsilon_{\mu,i}^{(t)})^2\Big)^{\frac{1}{2}} + 2. \end{aligned} \tag{5}$$

It's obvious that normalizing the outputs by noise-free moments will lead to a constant upper bound and impose a constraint on ICS. In contrast, the distance for the unnormalized outputs is unbounded (see an example of the unbounded distance in the supplementary material). Nevertheless, the noise cannot be controlled in practice, and for high-dimensional outputs, the bound in Eq. (5) might be too loose to constraint the distance due to a large d. In this case, the ICS for the non-trivial distributions cannot be bounded effectively by controlling the first two moments as BN techniques have done. Then, the analysis of the lower bound is required.

3.2. The Lower Bound

For convenience, let $\boldsymbol{x} = (x_1, x_2, \ldots, x_d)$ and $\boldsymbol{y} = (y_1, y_2, \ldots, y_d)$. Then, the lower bound on the distance is obtained by constructing a 1-Lipschitz function.

Theorem 2. *Suppose that* $C > 0$ *is a real number, and* $n \ge 2$ *is an integer. Then,*

$$\begin{aligned} W(p_l^{(t+\Delta t)}, p_l^{(t)}) = & \sup_{||f||_L \le 1} \mathbb{E}_{\boldsymbol{x}\sim p_l^{(t+\Delta t)}}[f(\boldsymbol{x})] - \mathbb{E}_{\boldsymbol{y}\sim p_l^{(t)}}[f(\boldsymbol{y})] \\ \ge & \big|\mathbb{E}_{\boldsymbol{x}\sim p_l^{(t+\Delta t)}}[f_{n,C}(\boldsymbol{x})] - \mathbb{E}_{\boldsymbol{y}\sim p_l^{(t)}}[f_{n,C}(\boldsymbol{y})]\big|, \end{aligned} \tag{6}$$

where $f_{n,C}$ *is the* 1*-Lipschitz function defined as*

$$f_{n,C}(\boldsymbol{x}) = \frac{1}{nC^{n-1}d^{\frac{1}{2}}}\Big(\sum_{|x_i|\le C} x_i^n + \sum_{x_i < -C} (-C)^n + \sum_{x_i > C} C^n\Big). \tag{7}$$

To simplify the analysis, assume that the supports of the distributions are subsets of $[-C_0, C_0]^d$ for some $C_0 > 0$. Then, the lower bound is formed by the nth-order moments. For $n > 2$, it's straightforward that the high-order moments affect the lower bound, which cannot be controlled by BN especially in the case of the relaxed upper bound. On the other hand, for $n = 2$ and the normalized output, the lower bound is

$$\begin{aligned} & W(p_l^{(t+\Delta t)}, p_l^{(t)}) \\ \ge & \frac{1}{2C_0 d^{\frac{1}{2}}}\Big|\sum_{i=1}^d (\epsilon_{\mu,i}^{(t+\Delta t)})^2 + \epsilon_{\sigma^2,i}^{(t+\Delta t)} - (\epsilon_{\mu,i}^{(t)})^2 - \epsilon_{\sigma^2,i}^{(t)}\Big|. \end{aligned} \tag{8}$$

The lower bound in Eq. (8) is dominated by the noise. Thus, BN degrades in such case especially for micro-batches. Some methods have been proposed, *e.g.*, GN, to reduce the noise of the moments rather than eliminating the noise. So the lower bound is still dependent on moments.

Based on such analysis of BN, this paper proposes an algorithm with an adjustable upper bound that is noise-free and moment-independent to further bound the distance.

3.3. Vanilla Unitization

The proposed algorithm unitizes layers' outputs, and the vanilla unitization transformation is defined as

$$g(\boldsymbol{x}) = \begin{cases} \dfrac{\boldsymbol{x}}{||\boldsymbol{x}||_2} & , ||\boldsymbol{x}||_2 \ne 0 \\ \boldsymbol{c} & , ||\boldsymbol{x}||_2 = 0 \end{cases}, \tag{9}$$

where $\boldsymbol{c}$ is a constant unit vector. Similarly, the upper bound for the unitized output is given. In fact, the EM distance for $g(\boldsymbol{x})$ is defined as

$$\begin{aligned} &W(p_U^{(t+\Delta t)}, p_U^{(t)}) \\ &= \sup_{||f||_L \leq 1} \mathbb{E}_{\boldsymbol{x}\sim p_l^{(t+\Delta t)}}[f(g(\boldsymbol{x}))] - \mathbb{E}_{\boldsymbol{y}\sim p_l^{(t)}}[f(g(\boldsymbol{y}))], \end{aligned} \tag{10}$$

where $p_U^{(t)}$ is the distribution of the unitized outputs.

Theorem 3.1. *Suppose that for $\boldsymbol{x} \sim p_l^{(t)}$, $g(\boldsymbol{x}) \sim p_U^{(t)}$. Then,*

$$W(p_U^{(t+\Delta t)}, p_U^{(t)}) \leq 2. \tag{11}$$

For g, the upper bound is absolutely a constant independent of all parameters including d. Then, ICS for the unitized outputs is exactly bounded in spite of the distribution $p_l^{(t)}$. Hence, by unitizing the outputs, ICS is fully controlled by this constant bound in fact. However, the constant upper bound leads to the other problem. For $t = 0$ and any $\Delta t > 0$, the distribution $p_U^{(\Delta t)}$ is constrained such that the distance between $p_U^{(\Delta t)}$ and $p_U^{(0)}$ is no more than the constant 2. This might be a severe problem, especially when the network is poorly initialized. Thus, the unitization has to be modified.

3.4. Modified Unitization

To alleviate the problem of the very tight bound, define the transformation, which partly unitizes the outputs, as

$$g(\boldsymbol{x}; \alpha) = \begin{cases} \boldsymbol{c} & , ||\boldsymbol{x}||_2 = 0, \alpha = 1 \\ \dfrac{\boldsymbol{x}}{\alpha ||\boldsymbol{x}||_2 + (1-\alpha)} & , other \end{cases}, \tag{12}$$

where $\alpha \in [0, 1]$ is a parameter. Analogously, the upper bound w.r.t. $g(\boldsymbol{x}; \alpha)$ is given.

Theorem 3.2. *Suppose that for $\alpha \in [0, 1]$ and $\boldsymbol{x} \sim p_l^{(t)}$, $g(\boldsymbol{x}; \alpha) \sim p_U^{(t)}$. Then,*

$$\begin{aligned} W(p_U^{(t+\Delta t)}, p_U^{(t)}) \leq &\mathbb{I}_{\alpha=0}(\alpha) \cdot (\mathbb{E}_{\boldsymbol{x}\sim p_l^{(t+\Delta)}}[||\boldsymbol{x}||_2] \\ &+ \mathbb{E}_{\boldsymbol{y}\sim p_l^{(t)}}[||\boldsymbol{y}||_2]) + \mathbb{I}_{\alpha>0}(\alpha) \cdot \frac{2}{\alpha}. \end{aligned} \tag{13}$$

Note that $\alpha = 0$ implies $g(\boldsymbol{x}; \alpha)$ is an identity mapping, and $\alpha > 0$ implies the distance is exactly bounded by $2/\alpha$. Thus, the bound is dominated by α, and the very bound is obtained by fine-tuning α over $[0, 1]$.

Furthermore, considering a set of parameters $\boldsymbol{\alpha} = (\alpha_1, \alpha_2, \ldots, \alpha_d) \in [0, 1]^d$, the general unitization is defined as

$$g(\boldsymbol{x}; \boldsymbol{\alpha}) = \begin{cases} \boldsymbol{0} & , ||\boldsymbol{x}||_2 = 0 \\ \left((||\boldsymbol{x}||_2 - 1) \cdot \text{diag}(\boldsymbol{\alpha}) + E\right)^{-1} \boldsymbol{x} & , ||\boldsymbol{x}||_2 > 0 \end{cases}, \tag{14}$$

where diag$(\boldsymbol{\alpha})$ is a diagonal matrix of $\boldsymbol{\alpha}$ and E is a identity matrix. Likewise, the upper bound for $g(\boldsymbol{x}; \boldsymbol{\alpha})$ is given.

Theorem 3.3. *Suppose that for $\boldsymbol{\alpha} \in [0, 1]^d$ and $\boldsymbol{x} \sim p_l^{(t)}$, $g(\boldsymbol{x}; \boldsymbol{\alpha}) \sim p_U^{(t)}$. Then,*

$$\begin{aligned} W(p_U^{(t+\Delta t)}, p_U^{(t)}) \leq &\mathbb{I}_{\min_j \alpha_j > 0}(\boldsymbol{\alpha}) \cdot \frac{2}{\min_j \alpha_j} \\ &+ \mathbb{I}_{\min_j \alpha_j = 0}(\boldsymbol{\alpha}) \cdot (\mathbb{E}_{\boldsymbol{x}\sim p_l^{(t+\Delta)}}[||\boldsymbol{x}||_2] \\ &+ \mathbb{E}_{\boldsymbol{y}\sim p_l^{(t)}}[||\boldsymbol{y}||_2] + 2). \end{aligned} \tag{15}$$

The minimum $\alpha_* = \min_j \alpha_j$ dominates the upper bound. If $\alpha_* = 0$, then there exists i such that the scale of x_i remains unchanged after the unitization, and the EM distance for the marginal distribution of x_i is unbounded. In contrast, the constant bound $2/\alpha^*$ is obtained by $\alpha_* > 0$. Furthermore, if α_i is fixed for some i, the other parameters $\alpha_j, j \neq i$ can be freely fine-tuned over $[\alpha_i, 1]$ without changing the bound. Thus, $g(\boldsymbol{x}; \boldsymbol{\alpha})$ is more flexible, and used in the proposed algorithm. However, the unitized outputs lose some information, *e.g.*, similarity between the samples, which cannot be recovered by an affine transformation like that in BN. The smaller bound leads to more information loss, and a trade-off is required. Then, the unitization algorithm is given.

3.5. Algorithm

For a network, $\boldsymbol{\alpha}$ in each unitization layer is trained with the weights to reduce the objective function. However, since $\boldsymbol{\alpha} \in [0, 1]^d$, the training would lead to a constrained optimization problem. To avoid the problem and make the training stable, this paper uses a simple interpolation method for Eq. (14). The practical transformation is defined as

$$g(\boldsymbol{x}; \boldsymbol{\alpha}, \epsilon) = \left[\frac{1}{\sqrt{||\boldsymbol{x}||_2^2 + \epsilon}} \boldsymbol{\alpha} + (\boldsymbol{1} - \boldsymbol{\alpha})\right] \odot \boldsymbol{x}, \tag{16}$$

where $\epsilon > 0$ makes the non-zero denominator, and $\odot$ represents element-wise production. Likewise, we provide the upper bound for the practical unitization.

Theorem 3.4. *Suppose that for $\boldsymbol{\alpha} \in \mathbb{R}^d, \epsilon > 0$ and $\boldsymbol{x} \sim p_l^{(t)}$, $g(\boldsymbol{x}; \boldsymbol{\alpha}, \epsilon) \sim p_g^{(t)}$. Then,*

$$\begin{aligned} W(p_g^{(t+\Delta t)}, p_g^{(t)}) \leq &2||\boldsymbol{\alpha}||_\infty + ||\boldsymbol{1} - \boldsymbol{\alpha}||_\infty (\mathbb{E}_{\boldsymbol{x}\sim p_l^{(t+\Delta t)}}[||\boldsymbol{x}||_2] \\ &+ \mathbb{E}_{\boldsymbol{y}\sim p_l^{(t)}}[||\boldsymbol{y}||_2]). \end{aligned} \tag{17}$$

Intuitively, the theoretical unitization (14) and practical unitization (16) tune the bounds (15) and (17) respectively by $\boldsymbol{\alpha}$ in a similar way that $\boldsymbol{\alpha} \to \boldsymbol{1}^-$ yields the tight bound while $\boldsymbol{\alpha} \to \boldsymbol{0}^+$ yields the loose bound, though the bound

(17) for the practical one is relatively relaxed. In addition, $||\mathbf{1}-\boldsymbol{\alpha}||_\infty$ in the second term of the bound (17) requires $\boldsymbol{\alpha} \in [0,2]^d$ for the reduction of ICS, which is a weaker constraint, compared with $\boldsymbol{\alpha} \in [0,1]^d$ in the theoretical bound (15). Therefore, the effects of the practical unitization are similar to those of theoretical one.

In the proposed algorithm, the unitization Eq. (16) is embedded into the framework of BN to reduce the information loss. In fact, only the unitization might require large $\boldsymbol{\alpha}$ to bound the EM distance with more information loss. In contrast, there is less information loss in BN since similarity between the normalized outputs remains unchanged and the affine transformation in BN can recover some information. Thus, the proposed algorithm integrates these two techniques to bound ICS with reasonable information loss. The algorithm is presented in Algorithm 1, where element-wise division is also denoted by $/$. The moments $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}^2$ in inference is computed in the same way [15].

Algorithm 1 Unitization Algorithm

```
Input: dataset $\{\boldsymbol{x}_i\}_{i=1}^n$, trainable parameters $\boldsymbol{\alpha}, \boldsymbol{\gamma}$ and $\boldsymbol{\beta}$
Output: unitized results $\{\boldsymbol{y}_i\}_{i=1}^n$
1: $\boldsymbol{\mu} \leftarrow \frac{1}{n}\sum_i \boldsymbol{x}_i$
2: $\boldsymbol{\sigma}^2 \leftarrow \frac{1}{n}\sum_i (\boldsymbol{x}_i - \boldsymbol{\mu})^2$
3: for $i \leftarrow 1$ to $n$ do
4:     $\hat{\boldsymbol{x}}_i \leftarrow (\boldsymbol{x}_i - \boldsymbol{\mu})/\sqrt{\boldsymbol{\sigma}^2 + \epsilon}$
5:     $p \leftarrow 1/\sqrt{||\hat{\boldsymbol{x}}_i||_2^2 + \epsilon}$
6:     $\overline{\boldsymbol{x}}_i \leftarrow [p\boldsymbol{\alpha} + (\mathbf{1} - \boldsymbol{\alpha})] \odot \hat{\boldsymbol{x}}_i$
7:     $\boldsymbol{y}_i \leftarrow \boldsymbol{\gamma} \odot \overline{\boldsymbol{x}}_i + \boldsymbol{\beta}$
8: end for
```

3.6. Unitized Convolutional Layers

To take into account the spatial context of image data, this paper also proposes the unitized convolutional layers. As recommended by [15], the moments $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}^2$ in Algorithm 1 are computed over the whole mini-batch at different locations w.r.t. a feature map, and they are shared within the same feature map (Figure 1(a)). But the norm in the unitization is computed in a different way. How to compute the norm is determined by the definition of a single sample for image data. A simple algorithm follows the idea of BN in convolutional layers, where the pixels at the same location in all channels is regarded as a single sample (Figure 1(b)), and then this sample will be unitized by its norm. However, this algorithm scales the pixels by location-dependent norms, ignoring the spatial context.

Thus, the computation of the norms has to be modified. Instead, the pixels at all locations in all feature maps within an image form a single sample (Figure 1(c)). The sample's norm will be location-independent and shared within these pixels. However, this will lead to a very large norm for a pixel. As the inverse of the norm, p in Algorithm 1 will be relatively small and make $p\boldsymbol{\alpha}$ be ignored when $\boldsymbol{\alpha}$ is being fine-tuned. Then $\boldsymbol{\alpha}$ only scales $\hat{\boldsymbol{x}}_i$ by $1-\boldsymbol{\alpha}$, but the scale has been controlled by $\boldsymbol{\gamma}$. Hence, the norm is divided by a constant related to the number of the pixels before the unitization.

Algorithm 2 Unitization Algorithm for Convolutional Layers

```
Input: dataset $\mathcal{D} = \{x_{ij}^{(k)} | 1 \le k \le N, 1 \le i \le C, 1 \le j \le HW\}$, trainable parameters $\boldsymbol{\alpha}, \boldsymbol{\gamma}$ and $\boldsymbol{\beta}$
Output: unitized results $\{y_{ij}^{(k)} | 1 \le k \le N, 1 \le i \le C, 1 \le j \le HW\}$
1: for $k \leftarrow 1$ to $N$ do
2:     $s \leftarrow 0$
3:     for $i \leftarrow 1$ to $C$ do
4:         for $j \leftarrow 1$ to $HW$ do
5:             $\hat{x}_{ij}^{(k)} = \mathrm{BN}(x_{ij}^{(k)}; \mathcal{D})$
6:             $s \leftarrow s + \hat{x}_{ij}^{(k)2}$
7:         end for
8:     end for
9:     $s \leftarrow s/(nHW)$
10:    $p \leftarrow 1/\sqrt{s+\epsilon}$
11:    for $i \leftarrow 1$ to $C$ do
12:        for $j \leftarrow 1$ to $HW$ do
13:            $\bar{x}_{ij}^{(k)} \leftarrow [p\alpha_i + (1-\alpha_i)]\hat{x}_{ij}^{(k)}$
14:            $y_{ij}^{(k)} \leftarrow \gamma_i \bar{x}_{ij}^{(k)} + \beta_i$
15:        end for
16:    end for
17: end for
```

The modified unitization is presented in Algorithm 2, where $x_{ij}^{(k)}$ denotes the value at the jth location in the ith feature map, generated from the kth training sample; $\hat{x}_{ij}^{(k)}$, $\bar{x}_{ij}^{(k)}$ and $y_{ij}^{(k)}$ are defined in the same way; $\mathrm{BN}(\cdot;\mathcal{D})$ denotes the normalization transformation for convolutional layers [15] using the dataset $\mathcal{D}$ without the affine transformation; α_i, γ_i and β_i denote the ith element of $\boldsymbol{\alpha}$, $\boldsymbol{\gamma}$ and $\boldsymbol{\beta}$, respectively; the n in Line 9 is a hyper-parameter that is set to HW by default.

4. Experiments

4.1. Estimated Moments

To verify the ability of the unitization controlling higher moments, we train simple neural networks on the MNIST dataset [19] and estimate the moments of a certain layer's outputs.

Network Architectures: The inputs of the networks are 28×28 images, followed by a stack of fully-connected layers with ReLU activations, consisting of 10 100-unit layers and one 8-unit layer. A BN/unitization layer follows

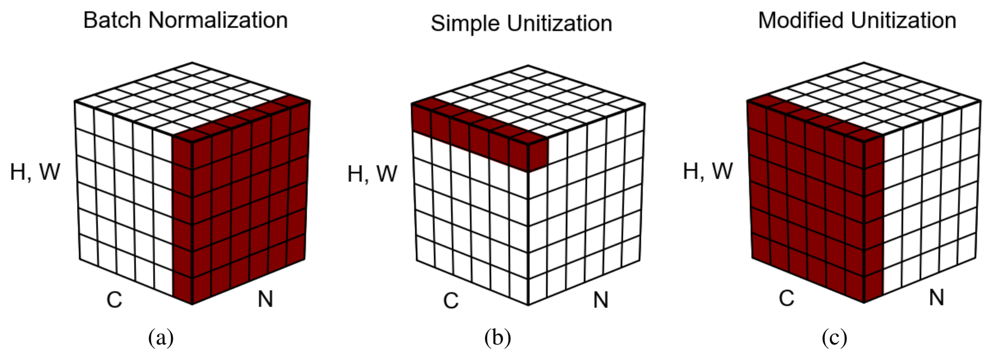

Figure 1: Different methods for estimating the statistics. Like the visualization of normalization methods [36], each subplot shows a feature map tensor, with N, C and (H, W) as the batch axis, the channel axis and the spatial axes, respectively. (a) shows the values of the pixels in red are used to compute the moments $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$ in BN, while (b) and (c) show the values of the pixels are used to obtain the norm. The estimated moments and norms are shared within these pixels.

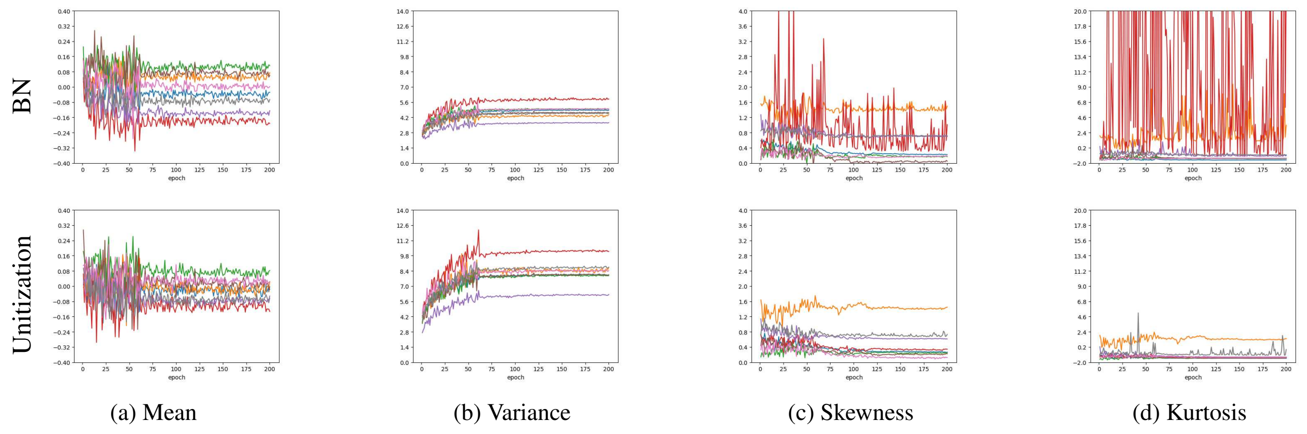

Figure 2: Moments estimated over the 8-unit layers' outputs. Each line in a subfigure represents the moment w.r.t. a unit's output. In general, more stable moments are obtained by using the unitization.

each fully-connected layer. The networks end with a fully-connected layer for 10 classes.

Implementation Details: The networks are trained using Mini-batch Gradient Descent for 200 epochs, and the batch size is 128. The learning rate starts from 0.05 and is divided by 5 at the 61th, 121th and 161th epochs. At the end of each epoch, the training samples are fed into the networks to get the normalized/unitized 8-unit layer's outputs. Then, we estimate the mean, variance, skewness and kurtosis over the outputs.

Results: As is shown in Figure 2, the estimated mean and variance for both BN and the unitization layers are stable through the training. However, the estimated skewness and kurtosis are unstable w.r.t. the BN layer, with large fluctuation in the red line. By the lower bound in Eq. (6), the EM distance will be large. In contrast, there are more stable skewness and kurtosis of the unitized outputs. The proposed unitization controls the high-order moments

4.2. Classification Results on CIFAR

For the tasks of image recognition, we run the experiments on CIFAR-10 and CIFAR-100 datasets [17], following the data augmentation recommended by [21] (the experiments of comparing the unitization with GN [36] are provided in the supplementary material).

Network Architectures: The ResNet-20, ResNet-110 and ResNet-164 [11, 12] with BN or the unitization are trained to compare the performance. The ResNets follow the general architecture [12] with full pre-activation blocks.

Implementation Details: In each experiment, for BN and the unitization, the networks are initialized by the same weights that are generated using the method of [10] to reduce impacts of initialization. Each network is trained by Mini-batch Gradient Descent with Nesterov's momentum, and the same learning rate in the moment experiments is used. The momentum is 0.9 and the weight decay is 0.0005. The mini-batch sizes are 128 and 64 in training {ResNet-20, ResNet-110} and ResNet-164, respectively. Each network is evaluated after 200 epochs and the median accuracy of 5 runs is reported. The parameter $\boldsymbol{\alpha}$ is initialized to $\mathbf{0}$. The parameters $\boldsymbol{\gamma}$ and $\boldsymbol{\beta}$ are initialized to $\mathbf{1}$ and $\mathbf{0}$, respectively, as suggested by [15].

Table 1
Classification accuracy on the CIFAR-10 testing dataset.

Network	Mini-batch size	Accuracy
ResNet-20 (BN)	128	91.79%
ResNet-20 (Unitization)	128	**92.21**%
ResNet-110 (BN) [12]	128	93.63%
ResNet-110 (BN)	128	93.99%
ResNet-110 (Unitization)	128	**94.12**%
ResNet-164 (BN) [12]	128	94.54%
ResNet-164 (BN)	64	94.34%
ResNet-164 (Unitization)	64	**94.62**%

Results: Table 1 shows the results on the CIFAR-10 dataset, where the results in [12] are also presented for comparison. The proposed algorithm has better performance, compared with BN, which raises the classification accuracy for each ResNet. But there is less improvement in the accuracy of the deeper network, which might be explained by the less benefit from stacking more layers in deeper networks. Actually, the ResNet-1001 [12] only achieves an accuracy of 95.08% that is the limit of these ResNets with BN. The accuracy of ResNet-164 in [12] is only 0.54% less than that of the ResNet-1001, but using the unitization raises the accuracy by 0.08%.

The results on the CIFAR-100 dataset are reported in Table 2, where the unitization still outperforms BN, with increase of over 1% in the accuracy of each experiment.

Table 2
Classification accuracy on the CIFAR-100 testing dataset.

Network	Mini-batch size	Accuracy
ResNet-20 (BN)	128	66.43%
ResNet-20 (Unitization)	128	**67.49**%
ResNet-110 (BN)	128	72.27%
ResNet-110 (Unitization)	128	**73.31**%
ResNet-164 (BN) [12]	128	75.67%
ResNet-164 (BN)	64	76.56%
ResNet-164 (Unitization)	64	**77.58**%

4.3. Classification Results on ImageNet

We further evaluate the unitization on the ImageNet 2012 classification dataset [25]. The networks are trained on the 1.28M training images, and evaluated on the 50k validation images. Only the scale augmentation [11, 30] is used.

Network Architectures: Only the ResNet-101 with BN or the unitization is trained to compare the performance. The network follows the architecture [11] but with full pre-activation blocks [12]. Besides, the outputs of each shortcut connection and the final block are not normalized or unitized.

Implementation Details: Like the experiments on CIFAR datasets, the weights are initialized by the method [10], shared between the experiments and trained by the gradient descent with the same momentum, but the weight decay is 0.0001. The learning rate starts from 0.01 and is divided by 5 at the 31th, 61th and 91th epochs. The batch size is 64 for a single GPU. After 120 epochs, the networks are evaluated on the validation data by two methods. The first method resizes the images with the sorter side in $\{224, 256, 384, 480, 640\}$ and averages the scores over 42 crops at all scales (2 central crops for 224-scale images, 10 standard crops for other resized images). The second method adopts the fully-convolutional form and averages the scores over the same multi-scale images [11]. Besides, the n in Line 9 of Algorithm 2 is fixed and set to the same value in the training for the multi-scale images.

Table 3
Classification accuracy on the ImageNet dataset.

Algorithm	Method/Mini-batch size	Top-1	Top-5
BN	multi-scale crops/64	78.12%	93.45%
Unitization	multi-scale crops/64	**78.33**%	93.22%
BN	fully-convolution/64	76.47%	93.02%
Unitization	fully-convolution/64	**77.84**%	**93.33**%
BN [11]	fully-convolution/256	80.13%	95.40%

Results: In the results, the unitization outperforms BN in general, with only the top-5 accuracy for the first method less than that of BN. However, there is a performance gap between the reproduced result and the accuracy in [11], which might be explained by the different implementation details including the data augmentation, the architecture and hyper-parameters like the batch size and the learning rate. But for the evaluation method of fully-convolution recommended by [11], the accuracy increases by over 1% using the unitization. In general, the unitization shows higher performance for classification tasks.

5. Conclusion

This paper proposes an ICS measure by using the EM distance, and provides a theoretical analysis of BN through the upper and lower bounds. The moment-dependent upper bound has shown that BN techniques can effectively control ICS only for the low-dimensional outputs with small noise in the moments, but would degrade in other cases. Meanwhile, the high-order moments and noise that are out of BN's control have great impact on the lower bound. Then, this paper proposes the unitization algorithm with the noise-free and moment-independent upper bound. By training the parameter in the unitization, the bound can be fine-tuned to further control ICS. The experiments demonstrate the proposed algorithm's control of high-order moments and performance on the benchmark datasets including CIFAR-10, CIFAR-100 and ImageNet.

References

[1] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In *International Conference on Machine Learning*, pages 214–223, 2017.

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

[3] Yongqiang Cai, Qianxiao Li, and Zuowei Shen. A quantitative analysis of the effect of batch normalization on gradient descent. *arXiv preprint arXiv:1810.00122*, 2018.

[4] Minhyung Cho and Jaehyung Lee. Riemannian approach to batch normalization. In *Advances in Neural Information Processing Systems*, pages 5225–5235, 2017.

[5] Victor Lempitsky Dmitry Ulyanov, Andrea Vedaldi. Instance normalization: The missing ingredient for fast stylization. *arXiv:1607.08022*, 2016.

[6] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015.

[7] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256, 2010.

[8] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[9] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Computer Vision (ICCV), 2017 IEEE International Conference on*, pages 2980–2988. IEEE, 2017.

[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015.

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

[12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *European conference on computer vision*, pages 630–645. Springer, 2016.

[13] Geoffrey Hinton, Li Deng, Dong Yu, George E Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N Sainath, et al. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal processing magazine*, 29(6):82–97, 2012.

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

[15] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[16] Jonas Kohler, Hadi Daneshmand, Aurelien Lucchi, Ming Zhou, Klaus Neymeyr, and Thomas Hofmann. Exponential convergence rates for batch normalization: The power of length-direction decoupling in non-convex optimization. *arXiv preprint arXiv:1805.10694*, 2018.

[17] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.

[18] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[19] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

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

[21] Chen-Yu Lee, Saining Xie, Patrick Gallagher, Zhengyou Zhang, and Zhuowen Tu. Deeply-supervised nets. In *Artificial Intelligence and Statistics*, pages 562–570, 2015.

[22] Chunjie Luo, Jianfeng Zhan, Lei Wang, and Qiang Yang. Cosine normalization: Using cosine similarity instead of dot product in neural networks. *arXiv preprint arXiv:1702.05870*, 2017.

[23] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[24] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.

[25] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015.

[26] Tim Salimans and Diederik P Kingma. Weight normalization: A simple reparameterization to accelerate training of deep neural networks. In *Advances in Neural Information Processing Systems*, pages 901–909, 2016.

[27] Shibani Santurkar, Dimitris Tsipras, Andrew Ilyas, and Aleksander Madry. How does batch normalization help optimization? In *Advances in Neural Information Processing Systems*, pages 2483–2493, 2018.

[28] Hidetoshi Shimodaira. Improving predictive inference under covariate shift by weighting the log-likelihood function. *Journal of statistical planning and inference*, 90(2):227–244, 2000.

[29] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *Nature*, 550(7676):354, 2017.

[30] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[31] Saurabh Singh and Abhinav Shrivastava. Evalnorm: Estimating batch normalization statistics for evaluation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3633–3641, 2019.

[32] Jonathan J Tompson, Arjun Jain, Yann LeCun, and Christoph Bregler. Joint training of a convolutional network and a graphical model for human pose estimation. In *Advances in neural information processing systems*, pages 1799–1807, 2014.

[33] Cédric Villani. *Optimal transport: old and new*, volume 338. Springer Science & Business Media, 2008.

[34] Guangrun Wang, Ping Luo, Xinjiang Wang, Liang Lin, et al. Kalman normalization: Normalizing internal representations across network layers. In *Advances in Neural Information Processing Systems*, pages 21–31, 2018.

[35] 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. *IEEE transactions on neural networks and learning systems*, 2018.

[36] Yuxin Wu and Kaiming He. Group normalization. *arXiv preprint arXiv:1803.08494*, 2018.

[37] Greg Yang, Jeffrey Pennington, Vinay Rao, Jascha Sohl-Dickstein, and Samuel S Schoenholz. A mean field theory of batch normalization. *arXiv preprint arXiv:1902.08129*, 2019.