Re-Thinking the Effectiveness of Batch Normalization and Beyond

Hanyang Peng, Yue Yu, and Shiqi Yu

Abstract—Batch normalization (BN) is used by default in many modern deep neural networks due to its effectiveness in accelerating training convergence and boosting inference performance. Recent studies suggest that the effectiveness of BN is due to the Lipschitzness of the loss and gradient, rather than the reduction of internal covariate shift. However, questions remain about whether Lipschitzness is sufficient to explain the effectiveness of BN and whether there is room for vanilla BN to be further improved. To answer these questions, we first prove that when stochastic gradient descent (SGD) is applied to optimize a general non-convex problem, three effects will help convergence to be faster and better: (i) reduction of the gradient Lipschitz constant, (ii) reduction of the expectation of the square of the stochastic gradient, and (iii) reduction of the variance of the stochastic gradient. We demonstrate that vanilla BN only with ReLU can induce the three effects above, rather than Lipschitzness, but vanilla BN with other nonlinearities like Sigmoid, Tanh, and SELU will result in degraded convergence performance. To improve vanilla BN, we propose a new normalization approach, dubbed complete batch normalization (CBN), which changes the placement position of normalization and modifies the structure of vanilla BN based on the theory. It is proven that CBN can elicit all the three effects above, regardless of the nonlinear activation used. Extensive experiments on benchmark datasets CIFAR10, CIFAR100, and ILSVRC2012 validate that CBN makes the training convergence faster, and the training loss converges to a smaller local minimum than vanilla BN. Moreover, CBN helps networks with multiple nonlinear activations (Sigmoid, Tanh, ReLU, SELU, and Swish) achieve higher test accuracy steadily. Specifically, benefitting from CBN, the classification accuracies for networks with Sigmoid, Tanh, and SELU are boosted by more than 15.0%, 4.5%, and 4.0% on average, respectively, which is even comparable to the performance for ReLU.

Index Terms—Normalization, Batch Normalization, Accelerating Convergence.

1 Introduction

Batch normalization (BN) [1] is a key technique in deep neural network (DNN) development, known to accelerate training and significantly improve generalization performance during inference. Its enormous success has inspired a range of normalization approaches for other learning scenarios, such as layer normalization (LN) for Recurrent Neural Network (RNN) and Transformer architectures [2], [3], instance normalization (IN) for neural stylization [4], spectral normalization for generative adversarial networks (GANs) [5] , adaptive batch normalization (AdaBN) for fine-tune leaning [6], and shuffle batch normalization (ShuffleBN) for contrastive learning [7]. BN normalizes the features along the batch dimension, so the performance of BN is influenced by the batch size. Hence, various other normalization approaches, group normalization (GN) [8], batch renormalization (BRN) [9], and moving average batch normalization (MABN) [10], are proposed to restore the performance of BN in small cases. However, when the batch size is sufficiently large, these newly proposed normalization methods exhibit no superiority to BN.

The practical success of BN is indisputable, but the roots of its effectiveness remain largely unexplored [11]. The reduction in internal covariate shift (ICS) induced by BN was originally thought to interpret the effectiveness of BN. Recently, however, recent studies by [12] have challenged this view, arguing that the link between ICS reduction and performance improvement with BN is weak. Instead, from the perspective of landscape smoothness, [12] demonstrated that BN results in the Lipschitzness of both the loss and the gradient, which induces more predictive and stable gradients, thus allowing for faster training; then, some other works further explained the success of BN with the view of improving Hessian conditioning (that is also a type of Lipschitzness) [13], [14].

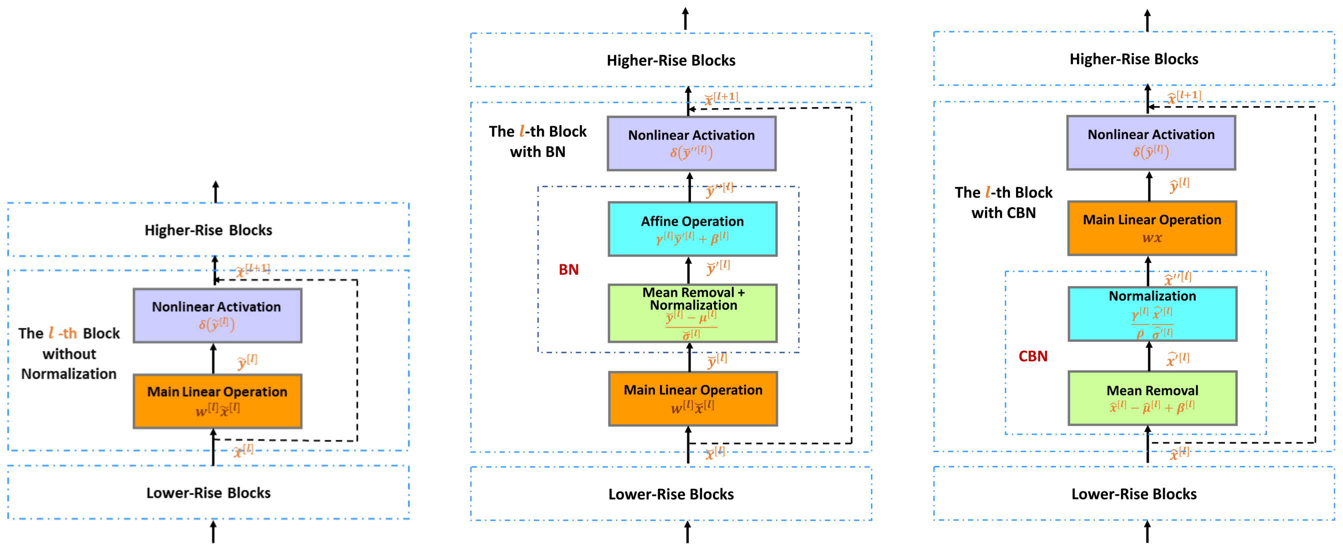
However, two open questions about BN that still need to be further explored,

Is Lipschitzness sufficient to explain the effectiveness of BN? Is there room to further improve vanilla BN, rather than mere explanation?

The efficacy of BN is primarily embodied by two aspects – accelerating the training process and boosting the inference performance, which are closely related to *faster* optimizing convergence and a final *smaller* converged minimum. More fundamentally, BN can be viewed as an optimization algorithm, so we adopt a top-down strategy to answer both questions above from the perspective of optimization. Specifically, we first deduce sufficient conditions for faster convergence and smaller converged minima, and then we identify whether BN can induce some of these conditions to explain its effectiveness. Moreover, we try to modify the structure of BN to elicit all of the above conditions to complete BN.

Following this strategy, we prove that three conditions

- *Hanyang Peng is with Peng Cheng Laboratory, Shenzhen, Guangdong, China.*
- *Yu Yue is with Peng Cheng Laboratory, Shenzhen, Guangdong, P.R. China, and the College of Computer, National University of Defense Technology, Changsha, China.*
- *Shiqi Yu is with the Research Institute of Trustworthy Autonomous Systems, and the Department of Computer Science and Engineering at Southern University of Science and Technology, Shenzhen, China. (Corresponding author:Shiqi Yu, E-mail: sqyu@sustech.edu.cn)*



(a) Network w/o Normalization (b) Network with vanilla BN (c) Network with CBN

Fig. 1. The structures of a standard block embedded with (a) no normalization, (b) vanilla BN and (c) CBN. The shortcut is optional. Vanilla BN is placed behind the linear activation layer, while CBN is located in the front of the linear activation layer. Moreover, mean removal and normalization with ℓ_2 norm are tangled in vanilla BN, but the two parts are decoupled in CBN. Additionally, CBN adds ρ to tune the magnitude of normalization for better performance.

- reduction in the gradient Lipschitz constant, reduction in the stochastic gradient squared expectation, and reduction in the stochastic gradient variance - can make the convergence faster and the converged minimum smaller, when stochastically optimizing a general nonconvex problem. The first two conditions are responsible for the ”faster” aspect, while the last two conditions are responsible for the ”smaller” aspect. We then propose a modified version of BN, called complete batch normalization (CBN), which can theoretically induce all three of these conditions. CBN involves four key modifications, as shown in Figure 1. First, we place normalization before the linear activation, which justifies the use of a wider range of activation functions in DNNs, including Tanh and Sigmoid. Second, we decouple the mean removal and normalization processes. Third, we add a new parameter to tune the magnitude of normalization, which can lower the gradient Lipschitz constant and the stochastic gradient squared expectation. Fourth, we substitute the statistics determined over a single batch with the adaptive moving average statistics that utilize historical batch information, thus lowering the variance of the stochastic gradient. Meanwhile,we also reveal the limitations of BN, showing that it can induce the three conditions only with ReLU, and may perform poorly with other activation functions such as Tanh, Sigmoid and SELU. Overall, CBN provides a more complete and theoretically justified approach to batch normalization, with the potential to improve the performance of DNNs.

Our contributions are summarized as follows:

- We we view BN-like normalization as an optimization algorithm and provide theoretical evidence for the three key effects that can accelerate convergence and decrease the converged minima while stochastically optimizing a nonconvex problem. This theoretical foundation enables us to explain and enhance vanilla BN’s performance.
- We propose a novel normalization approach that is theoretically justified by theory and can elicit all the effects described above.
- We identify the root cause for the effectiveness of vanilla BN with ReLU and the inefficacy of vanilla BN with other nonlinear activation functions.

2 Related Work

Inspired by the success of BN, various normalization variants have been proposed to address specific learning scenarios. For instance, Layer Normalization (LN) [2] and Recurrent Batch Normalization (RBN) [15] were developed for use in recurrent neural networks, while Instance Normalization (IN) [4] was designed to enhance neural stylization. Adaptive Batch Normalization (AdaBN) [6] was proposed for practical domain adaptation. Spectral Normalization (SN) [5] helps prevent model collapse in generative adversarial networks. Stochastic Normalization (SN) [16] aims to enhance fine-tuning performance, and Shuffle Batch Normalization (ShuffleBN) [7] replaces the current batch statistics with the statistics of other batches to improve unsupervised contrastive learning.

Batch normalization (BN) is widely known to be ineffective in small batch size cases, and then many methods have been proposed to alleviate this problem. Synchronized batch normalization (SyncBN) [17] computes statistics across multiple GPUs. It actually does not essentially solve this issue but transforms it into an engineering task. Several methods mimic T the principle of BN while decoupling the computational batch from the normalization batch . Layer normalization (LN) [2] exploits instance-level statistics along the channel dimension, and group normalization (GN) [8] further divides all channels into predefined groups and uses group-wise statistics. Weight normalization (WN) [18] is a simple reparameterization of the weight vectors in a neural network that can also accelerate the convergence process. Another line of approaches utilizes moving average statistics in both forward and backward passes to restore the performance when the batch size is insufficient, such as batch renormalization (BRN) [9], memorized batch normalization (MBN) [19], online normalization (ON) [20] and moving averaged batch normalization (MABN) [10]. Although these normalization methods are more effective than vanilla BN in small batch size cases, vanilla BN still outperforms them when the batch size is sufficiently large. Recently, instance level Meta Normalization (ILM) [21], instance enhancement batch normalization (IEBN) [22] and representative batch normalization (RBN) [23] have added an extra subnet or more steps with learning parameter to vanilla BN to introduce instance-level statistical information, and such approaches have achieved more competitive performance in large-size batch cases. However, these methods with instance-level statistical information still have to compute extra nonlinear operations during the inference procedure, while vanilla BN can be merged to the convolution layer or the fully-connected layer in the inference stage. Hence, these methods with instance-level statistics are inappropriate for applications in time-sensitive industrial scenarios. [24] extra introduces Proxy Normalization to GN to restore BNs beneficial properties that are not retained when solely using the prototypical batch-independent norm.

Despite ubiquitous use and the practical benefits of vanilla BN, the community has not yet reached a broad consensus on the theoretical explanations for its empirical tremendous success. The original paper claimed that BN was designed to reduce ICS to boost the training and inference performance [1]. However, [12] provided strong evidence supporting the idea that the link between the ICS reduction and the performance gain of BN is weak. [12] argued that the effectiveness of BN is attributed to smoothing the landscape of both the loss and the gradient, but its theoretical analysis only simply considers a single block with no nonlinearity activations. Afterward, several papers [13], [14] further uncovered that BN is beneficial to improving the Hessian conditioning, which can help to accelerate the convergence. In [25], the authors thought the larger learning rate brought by BN was the main reason behind its success. [26] believes the acceleration for BN is due to the fact that BN decouples the optimizing length and direction of the parameters, and theoretically proves that this decoupling will make the gradient-based descent algorithm achieve exponential convergence rates when optimizing a learning Halfspace problem and a multilayer perceptron (MLP) with one hidden layer. [27] demonstrates that scale-invariance of BN allows (stochastic) gradient descent to succeed with less tuning of learning rates. [28] theoretically and experimentally shows BN can provably prevent rank collapse for linear networks. [29] gives a uniform understanding of normalization in DNNs from the perspectives of stable forward propagation, informative forward propagation, and stable backward propagation.

Although these previous studies have contributed valuable insights into the mechanisms underlying vanilla BN from different perspectives, their theoretical analyses commonly focus on a single factor and often rely on idealized assumptions that do not accurately reflect the complexities of training modern DNNs. More importantly, they do not provide guidance on how to improve it. In contrast, our approach takes a top-down strategy, first identifying three effects that are sufficient to accelerate the training process and improve inference performance. We then propose a modification to vanilla BN that has been shown to achieve these effects.

3 Preliminary

In this section, we theoretically deduce vital factors that are responsible for accelerating convergence and leading the objective function to converge to a smaller minimum when stochastically optimizing a general nonconvex problem.

In a machine learning task, given a set of samples $\{x_i\}_{i=1}^n$, the optimization objective is commonly the empirical risk loss, *i.e.*,

$$F\left(w;\{x_i\}_{i=1}^n\right) = \frac{1}{n}\sum_{i=1}^{n} f_i(w;x_i), \tag{P}$$

where $f_i(w;x_i)$ is the loss of the i-th sample with respect to w. Note that $F\left(w;\{x_i\}_{i=1}^n\right)$ and $f_i(w;x_i)$ are abbreviated as $F(w)$, F and $f_i(w)$, f, respectively.

For large-scale machine learning, such as deep learning, SGD is commonly applied to minimize the empirical risk loss (P) due to its low computational cost. At the k-th iteration, an index set $\mathcal{B}_k$ is randomly sampled in a batch where $1 \leq |\mathcal{B}_k| \ll n$, and then w is then updated with

$$w_{k+1} \leftarrow w_k - \alpha_k v_k, \tag{1}$$

where α_{k+1} is the learning rate. For the basic SGD, $v_k = \frac{1}{|\mathcal{B}_k|}\sum_{i_k\in\mathcal{B}_k}\nabla f_{i_k}(w_k;x_{i_k})$. In deep learning, modern SGD algorithms are applied [30], [31], [32]. They can be uniformly expressed as a weighted gradient sum: $v_k = \beta_{1_k}v_{k-1} + \beta_{2_k}\frac{1}{|\mathcal{B}_k|}\sum_{i_k\in\mathcal{B}_k}\nabla f_{i_k}(w_k;x_{i_k})$, where the weights β_{1_k} and β_{2_k} are predefined constants or functions with respect to $\frac{1}{|\mathcal{B}_k|}\sum_{i_k\in\mathcal{B}_k}\nabla f_{i_k}(w_k;x_{i_k})$ and the historical gradient $\frac{1}{|\mathcal{B}_t|}\sum_{i_t\in\mathcal{B}_t}\nabla f(w_t;x_{i_t})$ $(0 \leq t < k)$.

We adopt the following standard assumptions regarding the objective function in problem (P) when investigating its convergence behaviors.

Assumption 1. *The objective function in problem (P) satisfies:*

1). Function $F(w)$ is continuously differentiable and bounded, i.e., $F^* := \inf F(w) > -\infty$;

2). Each gradient $\nabla f_i(w)$ $(i = 1,2,3,...,n)$ is Lipschitz continuous, i.e., for any w_1, w_2 ,

$$\|\nabla f_i(w_1) - \nabla f_i(w_2)\| \leq \mathcal{L}\|w_1 - w_2\|, \tag{2}$$

where $\mathcal{L}$ is also called the gradient Lipschitz constant.

Below, under Assumption 1, we present a convergence analysis of the general objective function in problem (P).

Theorem 1. *Let Assumption 1 hold and let SGD in Eq. (2) be applied to optimize problem (P). Suppose that the stepsizes (the* *learning rates)* $\{\alpha_k\}$ *satisfy* $\alpha_k \leq \frac{1}{\mathcal{L}}$ *and* $\alpha_{k+1} \leq \alpha_k$. *Then, for any number of iterations* K*, we have*

$$\mathbb{E}\left[\frac{1}{K}\sum_{k=1}^{K}\|\nabla F(w_k)\|\right]^2 \leq \underbrace{\frac{2(\mathbb{E}[F(w_0)] - F^*)}{\alpha_K K}}_{T_1} + \underbrace{\frac{1}{K\alpha_K}\sum_{k=1}^{K}\alpha_k \mathbb{E}[\|\nabla v_k\|^2]}_{T_2} + \underbrace{\frac{1}{K\alpha_K}\sum_{k=1}^{K}\alpha_k \mathbb{E}\left[\|v_k - \nabla F(w_k)\|^2\right]}_{T_3}, \tag{3}$$

The proof of Theorem 1 can be found in Appendix A. blueIt is important to note that everal theoretical convergence analyses have been conducted on stochastically optimizing non-convex problems [33], [34], [35], [36], [37], [38], [39], and some intermediate results of these analyses is somewhat similar to those in Theorem 1. However, However, their final formulations are either impractical or have unfavorable performance for deep learning tasks. For instance, the analyses in [33], [34], [35] require large batch sizes (i.e., $b \propto \sqrt{n}$ where b is the batch size and n is the total number of samples), and the gradients of the universal samples should be computed periodically. Alternatively, the learning rate scenarios in [36], [37] are closely related to the total number of samples (i.e., $\alpha_k \propto \frac{1}{\sqrt{n}}$), or the learning rate decays with the number of steps (i.e., $\alpha_k \propto \frac{1}{\sqrt{k}}$) [38], [39]. In contrast, our convergence analysis in Theorem 1 does not impose strict conditions on the batch size and the learning rate. Therefore, it is more in line with the actual situation when training deep neural networks.

According to Theorem 1, we know that the convergence speed depends on the vanishing rate of T_1 in Eq. (3). It can be easily concluded that a large learning rate α_{k+1} will directly speed up the vanishing rate, which is in accordance with the conclusion in [25]. However, the learning rate is constrained by $\alpha_{k+1} < \frac{1}{\mathcal{L}}$. Thus, to accelerate the convergence of problem (P), we should decrease the gradient Lipschitz constant $\mathcal{L}$.

From Theorem 1, we also know that as $k \to \infty$, T_1 in Eq. (3) will approach to zero, and the final value of $\mathbb{E}\left[\frac{1}{K}\sum_{k=1}^{K}\|\nabla F(w_k)\|\right]^2$ depends on the value of T_2 and Term T_3 in Eq. (3). It is known that a final small value for the gradient is more likely to enjoy favorable performance in machine learning tasks. To achieve the goal of a small value for the gradient in Eq. (3), we need to lower the value of T_2 and T_3. v_k is an additive weighted gradient sum: $v_k = \beta_{1_k} v_{k-1} + \beta_{2_k}\frac{1}{|\mathcal{B}_k|}\sum_{i_k\in\mathcal{B}_k}\nabla f_{i_k}(w_k; x_{i_k})$. Thus, at each iteration reducing the gradient squared expectation $\mathbb{E}[\|\nabla f(w_k; x_{i_k})\|^2]$ is helpful to decrease $\mathbb{E}[\|v_k\|^2]$ and ultimately reduce the value of T_2. As for T_3, we known that $\mathbb{E}[(x-\mathcal{C})^2] = \mathbb{V}[x] + (\mathbb{E}[a]-\mathcal{C})^2$ where x is a random variable and $\mathcal{C}$ is a constant, so lowering the stochastic gradient variance $\mathbb{V}[\|v_k\|]$ helps to reduce $\mathbb{E}[\|v_k - \nabla F(w_k)\|^2]$ and finally bring down the value of T_3. Notably, the conclusion that gradient variance reduction is beneficial for convergence was initially demonstrated in [40], and a wealth of later ingenious optimization algorithms [33], [34], [35] were proposed to lower the variance of the stochastic gradient, resulting in great achievements in recent years.

In summary, if an approach ensures *fast* convergence and a *small* converged minimum when applying SGD, it should elicit at least one of the following effects:

- Reducing the gradient Lipschitz constant (*fast*);
- Reducing the expectation of the squared stochastic gradient (*small*);
- Reducing the variance of the stochastic gradient (*small*).

4 Complete Batch Normalization

In this section, we present a new normalization approach that can be easily embedded in the existing DNNs. It is proven to induce all three effects during training: reduction in the gradient Lipschitz constant, reduction in the expectation of the squared stochastic gradient, and reduction in the variance of the stochastic gradient. Note that the new approach induces these effects to make convergence faster and better by changing the structure of the network rather than devising ingenious optimization algorithms [33], [34], [35], [40].

4.1 Reduction in the Gradient Lipschitz Constant and Stochastic Gradient Squared Expectation

Inspired by the basic form of vanilla BN, we construct a new normalization approach, referred to as CBN. The standard block that embeds CBN layers of a network can be formulated as

$$\left.\begin{aligned} \mu_{\mathcal{B}_k}^{[l]} &= \frac{1}{|\mathcal{B}_k|}\sum_{i_k\in\mathcal{B}_k} x_{i_k}^{[l]}, \\ x'^{[l]}_{i_k} &= x_{i_k}^{l} - \mu_{\mathcal{B}_k}^{[l]} + \beta_k^{[l]}, \end{aligned}\right\}\text{(Mean Removal)} \tag{4}$$

$$\left.\begin{aligned} \sigma_{\mathcal{B}_k}^{[l]} &= \sqrt{\frac{1}{|\mathcal{B}_k|}\sum_{i_k\in\mathcal{B}_k}(x'^{[l]}_{i_k})^2}, \\ x''^{[l]}_{i_k} &= \frac{\gamma_k^{[l]}}{\rho}\cdot\frac{x'^{[l]}_{i_k}}{\sigma_{\mathcal{B}_k}^{[l]}}, \end{aligned}\right\}\text{(Normalization via } \ell_2 \text{ Norm)} \tag{5}$$

$$y_{i_k}^{[l]} = w_k^{[l]} x''^{[l]}_{i_k}, \qquad \text{(Linear Activation)} \tag{6}$$

$$x_{i_k}^{[l+1]} = \mathbb{I}(\text{SC}) x_{i_k}^{[l]} + \delta(y_{i_k}^{[l]}), \quad \text{(Nonlinear Activation)} \tag{7}$$

where CBN consists of two steps – mean removal and normalization. The subscript k is the iteration index and the superscript $[l]$ is the block index, $\beta_k^{[l]}$ and $\gamma_k^{[l]}$ are the shift parameter and the scale parameter to be learned, respectively, and ρ is a hyperparameter to tune the normalization magnitude. $\delta(\cdot)$ is an activation function. $\mathbb{I}(\text{SC})$ indicates that if there is a shortcut, it is 1, otherwise, it is 0. Note that we sometimes omit the block index $[l]$ in the following equations for simplicity.

We now compare a plain non-normalized network with the same network after inserting a CBN in each block to demonstrate the effectiveness of CBN. The standard network with no normalization and the standard network with CBN are respectively shown in Figure 1(a) and Figure

1(c). The inputs of both networks are fed with the same raw data and the outputs of both networks are fed to the same empirical risk loss. Moreover, the weight $w_k^{[l]}$ to be learned at any l-th block in both networks are identical. We have an additional CBN term that includes removal and normalization to process the inputs of any l-th block. In the following, we will theoretically demonstrate the two parts of CBN lower the stochastic gradient squared expectation and the gradient Lipschitz constant. It is worth noting that we employ $\tilde{(\cdot)}$ for the symbols in the network without normalization(such as $\tilde{x}_{i_k}^{[l]}$ and $\tilde{y}_{i_k}^{[l]}$), and utilize $\hat{(\cdot)}$ for the symbols in the network with normalization(such as $\hat{x}_{i_k}^{[l]}$ and $\hat{y}_{i_k}^{[l]}$) to better better distinguish them, and then we have the following conclusions.

Theorem 2. **[*Justification of Mean Removal, Eq. (4)*]** *A standard network with no normalization and a standard network with mean removal, the inputs of the first block and the outputs of the last block for the both networks satisfy* $\hat{x}_{i_k}^{[1]} = \tilde{x}_{i_k}^{[1]}$, $\|\nabla_{\hat{x}_{i_k}^{[L]}} \hat{f}_{i_k}\|_2 = \|\nabla_{\tilde{x}_{i_k}^{[L]}} \tilde{f}_{i_k}\|_2$ *and* $\|\nabla^2_{\tilde{x}_{i_k}^{[L]}} \hat{f}_{i_k}\|_2 = \|\nabla^2_{\tilde{x}_{i_k}^{[L]}} \tilde{f}_{i_k}\|_2$. *Suppose that at l-th block with mean removal* $0 \leq \hat{\eta}_k^{[l]} \leq 2$ *where* $\hat{\eta}_k^{[l]} = \frac{\hat{\mu}_{\mathcal{B}_k}^{[l]} - \beta_k^{[l]}}{\mathbb{E}[\hat{x}_{i_k}^{[l]}]}$. *For the arbitrary l-th block, we then have the following:*

(1) The upper bound of the gradient Lipschitz constant of $\hat{F}$ *with respect to* ${w_k}^{[l]}$ *for the network with mean removal is lower than the upper bound of the gradient Lipschitz constant of* $\tilde{F}$ *with respect to* ${w_k}^{[l]}$ *for the non-normalized network.*

(2) The upper bound of the stochastic gradient squared expectation $\mathbb{E}[\|\nabla_{w_k}^{[l]} \hat{f}_{i_k}\|^2]$ *for the network with mean removal is lower than the upper bound of the stochastic gradient squared expectation* $\mathbb{E}[\|\nabla_{w_k}^{[l]} \tilde{f}_{i_k}\|^2]$ *for the non-normalized network .*

Theorem 3. **[*Justification of Normalization via ℓ_2-Norm, Eq. (5)*]** *For the standard network with no normalization and a standard network with normalization via l_2-Norm , the inputs of the first block and the outputs of the last block for the both networks satisfy* $\hat{x}_{i_k}^{[1]} = \tilde{x}_{i_k}^{[1]}$, $\|\nabla_{\hat{x}_{i_k}^{[L]}} \hat{f}_{i_k}\|_2 = \|\nabla_{\tilde{x}_{i_k}^{[L]}} \tilde{f}_{i_k}\|_2$ *and* $\|\nabla^2_{\tilde{x}_{i_k}^{[L]}} \hat{f}_{i_k}\|_2 = \|\nabla^2_{\tilde{x}_{i_k}^{[L]}} \tilde{f}_{i_k}\|_2$. *Suppose that at any l-th block* $0 < \tau_k^{[l]} < 1$ *where* $\tau_k^{[l]} = \frac{\gamma_k^{[l]}}{\rho \hat{\sigma}_{\mathcal{B}_k}^{[l]}}$ *, for the arbitrary l-th block, we then have the following:*

(1) The upper bound of the gradient Lipschitz constant of $\hat{F}$ *with respect to* ${w_k}^{[l]}$ *for the network with normalization via ℓ_2-Norm is lower than the upper bound of the gradient Lipschitz constant of* $\tilde{F}$ *with respect to* ${w_k}^{[l]}$ *for the non-normalized network.*

(2) The upper bound of the stochastic gradient squared expectation $\mathbb{E}[\|\nabla_{w_k}^{[l]} \hat{f}_{i_k}\|^2]$ *for the network with normalization via ℓ_2-Norm is lower than the upper bound of the stochastic gradient squared expectation* $\mathbb{E}[\|\nabla_{w_k}^{[l]} \tilde{f}_{i_k}\|^2]$ *for the non-normalized network.*

The proof of Theorem 2 and Theorem 3 is provided in Appendix E and Appendix H. As shown in Figure 2, the assumption $0 \leq \eta_k \leq 2$ in Theorem 2 and the assumption $0 \leq \tau_k \leq 1$ in Theorem 3 can be commonly satisfied in practice.

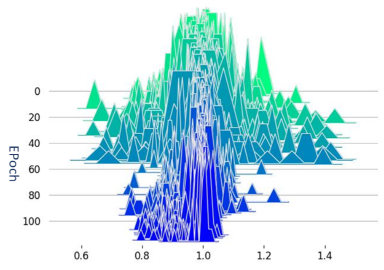
Remark 1. In [1], the positioning of BN was discussed, but the paper empirically concluded that placement before a non-linearity activation is better. [41] and [42] experimentally placed BN after a non-linearity activation as an optional experimental scheme, without making it a component of their proposed methods. In contrast, in our CBN, placing normalization after non-linearity is compulsory, which is theoretically justified. Note that empirical "try-and-error" methods are easily misled by superficial and incomplete phenomena. Vanilla BN placed before widely-used ReLU commonly performs better than vanilla BN placed after ReLU when the magnitude ρ is not tuned(the reason for this can be found in Section 5), and this phenomenon made prior works tend to place BN before a non-linearity activation. However, as our theory is guided without empirical distraction, placing normalization after a non-linearity activation with properly tuned magnitude enjoys significant improvements; for details, please refer to our experiments.

Remark 2. The proofs of Theorem 2 and Theorem 3 indicate that the upper bounds of the gradient Lipschitz constant and the stochastic gradient squared expectation are proportional to $\frac{1}{\rho^2}$, which means that a larger ρ speeds up convergence more quickly. However, ρ cannot be too large; otherwise, $\hat{x}_{i_k}$ will become too small, which will lower the dynamic range and even lead to underflow since machine precision is finite. Hence, ρ should be tuned moderately in practice. In a broad sense, adding ρ is simply equivalent to changing $f(x^\top w)$ to $f(\frac{x^\top}{\rho} w)$. Thus, it seems that the optimization trajectory remains the same when w is magnified by an exact factor ρ. w is not more likely to learn to be magnified exactly by a large factor ρ. The number of local minima of a DNN is very large, and they are scattered throughout the entire space. Therefore, adding ρ is likely to help w in the network to more quickly converge to a local minimum near the initial value rather than follow a long and indirect trajectory to converge to the original local minimum, since overparameterized neural networks commonly converge to lower training losses while their parameters hardly vary from their initial value [43], [44], [45]. It is important to note that in the case of using a deep and straight network, the prerequisite $0 < \frac{\gamma[l]}{\rho \hat{\sigma} \mathcal{B}^{[l]}} < 1$ stated in Theorem 3 is often not satisfied during initialization if we simply set $\rho = 1$. As a result, the norm of the gradient will exponentially increase with depth, in accordance with theoretical predictions [46]. Therefore, in such scenarios, it becomes necessary to tune the parameter ρ to ensure that $0 < \frac{\gamma[l]}{\rho \hat{\sigma} \mathcal{B}^{[l]}} < 1$. Furthermore, by adopting the gradient independence assumption presented in [46], we have the potential to achieve more robust theoretical results, thereby eliminating the need for the upper bounds in both Theorem 2 and Theorem 3.

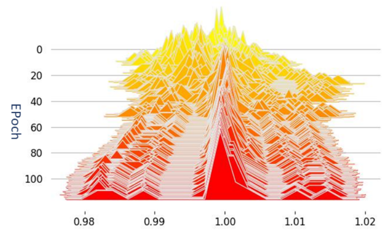
4.2 Reduction in the Variance of the Stochastic Gradient

We now turn our attention to a technique that can be embedded into CBN to reduce the stochastic gradient variance. Its core idea is substituting the statistics $\mu_{\mathcal{B}_k}$ and $\sigma_{\mathcal{B}_k}$ determined over the current batch with the adaptive moving average statistics during training.

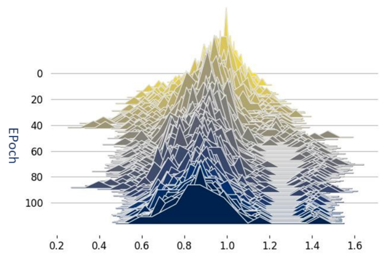
Recalling Eqs. (4-7), it is easy to know that $\nabla_{w_k} f_{i_k} = (\nabla_{y_{i_k}} f_{i_k}) {x''}_{i_k}^\top$. Then, we conclude that reducing the variance of ${x''}_{i_k}$ is helpful to minimize the variance of the stochastic gradient $\nabla_{w_k} f_{i_k}$ due to the facts $\mathbb{V}[ab] =$



(a) Layer #2

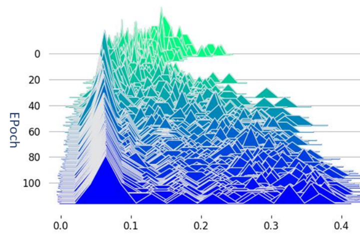


(b) Layer #8

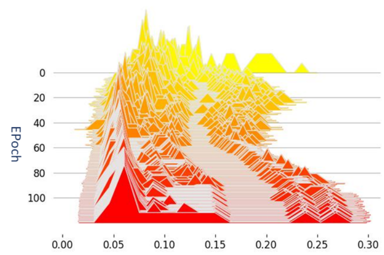


(c) Layer #14

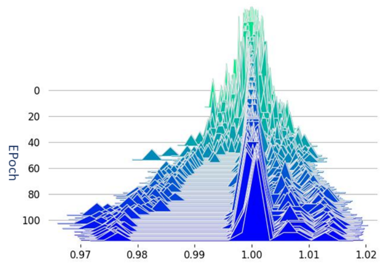
Fig. 2. The distribution cross channels of η_k of (a) layer#2.cbn (stage1.0.cbn1), (b) layer#8.cbn (stage2.0.cbn1) and (c) layer#14.cbn (stage3.0.cbn1) in ResNet-20 during training on CIFAR10, where $\eta_k = (\mu_{\mathcal{B}_k} - \beta_k)/\mathbb{E}[x_{i_k}^{[l]}]$ is defined in Theorem 2. $\mu_{\mathcal{B}_k}$ and β_k have been shown in Eq.(4). $\mathbb{E}[x_{i_k}^{[l]}]$ depending on the full sets make it difficult to compute, so we practically substitute it with the exponential moving average $\mu_{\mathcal{A}_k}$ in Eq.(8). In practice, the assumption $0 < \eta_k < 2$ in Theorem 2 can be commonly satisfied.



(a) Layer #2



(b) Layer #8



(c) Layer #14

Fig. 3. The distribution cross channels of τ_k of (a) layer#2.cbn (stage1.0.cbn1), (b) layer#8.cbn (stage2.0.cbn1) and (c) layer#14.cbn (stage3.0.cbn1) in ResNet-20 during training on CIFAR10, where $\tau_k^{[l]} = \gamma_k^{[l]}/\rho\hat{\sigma}_{\mathcal{B}_k}^{[l]}$ is defined in Theorem 3. γ_k and $\sigma_{\mathcal{B}_k}$ have been shown in Eq.(5), and we set $\rho = 1$ in the experiment. In practice, the assumption $0 \leq \tau_k \leq 1$ in Theorem 3 can be commonly satisfied.

```
**Algorithm 1.** Training in the $l$-th layer of a standard neural network with Complete Batch Normalization
**Input**: the training mini-batch data $x_k^{[l]} = [x_{1_k}^{[l]}, x_{2_k}^{[l]}, x_{|\mathcal{B}|_k}^{[l]}]$ from the $(l-1)$ block at the $k$-th iteration, and the buffer data $\mu_{|\mathcal{A}|_k}^{[l]}, \sigma_{|\mathcal{A}|_k}^{[l]}$, and the learning rate $\alpha_k$ at the $k$-th iteration, and the positive constant $\omega$.
**Output**: $x_k^{[l+1]}$
    $\mu_{\mathcal{B}_k}^{[l]} \leftarrow \frac{1}{|\mathcal{B}_k|}\sum_{i_k\in\mathcal{B}_k} x_{i_k}^{[l]}$
    $r_{\mu_k} \leftarrow \frac{\mu_{\mathcal{A}_k}^{[l]}}{\mu_{\mathcal{B}_k}^{[l]}}$
    ${x'}_{i_k}^{[l]} \leftarrow x_{i_k}^{l} - r_{\mu_k}\mu_{\mathcal{B}_k}^{[l]} + \beta_k^{[l]}$
    $(\sigma_{\mathcal{B}_k}^{[l]})^2 \leftarrow \frac{1}{|\mathcal{B}_k|}\sum_{i_k\in\mathcal{B}_k} ({x'}_{i_k}^{[l]})^2$
    $r_{\sigma_k} \leftarrow \frac{\sigma_{\mathcal{A}_k}^{[l]}}{\sigma_{\mathcal{B}_k}^{[l]}}$
    ${x''}_{i_k}^{[l]} \leftarrow \frac{\gamma_k^{[l]}}{\rho} \cdot \frac{{x'}_{i_k}^{[l]}}{r_{\sigma_k}\sigma_{\mathcal{B}_k}^{[l]}}$
    $y_{i_k}^{[l]} \leftarrow w_k^{[l]}{x''}_{i_k}^{[l]}$
    $x_{i_k}^{[l+1]} \leftarrow \mathbb{I}(\mathrm{SC})x_{i_k}^{[l]} + \delta(y_{i_k}^{[l]})$
    $\eta_k \leftarrow \max(1 - \omega\alpha_k, 0)$
    $\mu_{\mathcal{A}_{k+1}}^{[l]} \leftarrow \eta_k\mu_{\mathcal{A}_k}^{[l]} + (1-\eta_k)\mu_{\mathcal{B}_k}$
    $(\sigma_{\mathcal{A}_{k+1}}^{[l]})^2 \leftarrow (\eta_k\sigma_{\mathcal{A}_k}^{[l]})^2 + (1-\eta_k)(\sigma_{\mathcal{B}_k}^{[l]})^2$
```

$\frac{1}{2}\left(\mathbb{V}[a](\mathbb{E}[b^2] + \mathbb{E}^2[b]) + (\mathbb{E}[a^2] + \mathbb{E}^2[a])\mathbb{V}[b]\right)$ if the random variables a and b are independent. From Eqs.(4-5), we know that ${x''}_{i_k} = {\gamma(x_k - \mu_{\mathcal{B}_k} + \beta_k)}/{\rho\sigma_{\mathcal{B}_k}}$. Again exploiting $\mathbb{V}[ab] = \frac{1}{2}\left(\mathbb{V}[a](\mathbb{E}[b^2] + \mathbb{E}^2[b]) + (\mathbb{E}[a^2] + \mathbb{E}^2[a])\mathbb{V}[b]\right)$ and another fact $\mathbb{V}[a+b] = \mathbb{V}[a] + \mathbb{V}[b]$ when a and b are independent, minimizing the variance of the batch statistics $\mu_{\mathcal{B}_k}$ and $\sigma_{\mathcal{B}_k}$ is beneficial for decreasing the variance of ${x''}_{i_k}$ and will ultimately lower the variance of the stochastic gradient.

Because the batch statistics $\mu_{\mathcal{B}_k}$ and $\sigma_{\mathcal{B}_k}$ are Monte Carlo estimators, their variances are inversely proportional to the number of given examples. The exponential moving average statistics, $\mu_{\mathcal{A}_{k+1}} = \eta\mu_{\mathcal{A}_k} + (1-\eta)\mu_{\mathcal{B}_{k+1}}$ and $\sigma^2_{\mathcal{A}_k} = \eta\sigma^2_{\mathcal{A}_k} + (1-\eta)\sigma^2_{\mathcal{B}_k}$, are used in the testing stage in vanilla BN. Actually, the modest exploitation of exponential moving average statistics during training will reduce the variance, but applying historical statistics will also inevitably bring bias. Thus, we should strike a wise balance between variance reduction and bias increase by controlling η. If the bias is large, we should decrease η, and vice versa. Note that the bias is inversely proportional to the learning rate (the theoretical analysis is provided in Appendix E). Thus, adaptively adjusting η by carefully tracking the learning rate helps to reduce the variances of $\mu_{\mathcal{B}_k}$ and $\sigma_{\mathcal{B}_k}$. We present the *adaptive moving average technique* (AMAT) as follows:

$$\mu_{\mathcal{A}_{k+1}} = \eta_k\mu_{\mathcal{A}_k} + (1-\eta_k)\mu_{\mathcal{B}_k}, \tag{8}$$

$$\sigma^2_{\mathcal{A}_{k+1}} = \eta_k\sigma^2_{\mathcal{A}_k} + (1-\eta_k)\sigma^2_{\mathcal{B}_k}, \tag{9}$$

where η_k is a decreasing function of the learning rate α_k. The batch statistics $\mu_{\mathcal{B}_k}$ and $\sigma_{\mathcal{B}_k}$ are substituted by adaptive moving average statistics $\mu_{\mathcal{A}_k}$ and $\sigma_{\mathcal{A}_k}$ during training.

Notably, during training, if we use only the moving average historical statistics to normalize data, the current minibatch statistics will have less impact on backpropagation. This can result in model parameter growth without any improvement in loss. To solve this problem, as shown in Algorithm 1, we use a technique from [9] where we introduce ratios between the moving average historical statistics and the current minibatch statistics, treating them as a constant during computation. This allows us to use the historical statistics for the forward pass, while the backpropagation focuses on the current minibatch statistics.

Remark 3. BRN [9], MBN [19], ON [20] and MABN [10] also leverage moving average historical statistics during training. However, there are two differences between these existing techniques and our proposed AMAT.First, the previous normalization techniques adopted moving average historical statistics to correct the current statistics in cases with insufficient batch size and fill the gap between the training and inference statistics, while AMAT aims to reduce the variance of the stochastic gradient. Second, the previous normalization techniques adopted fixed η or those that diminished with the number of iterations, while the weighted parameter η in AMAT is set to adapt to the learning rate.

5 Re-Explaining the Effectiveness of BN

A standard block with BN is formulated as follows:

$$y_{i_k}^{[l]} = w_{i_k}^{[l]} x_{i_k}^{[l]}, \tag{10}$$

$$y'^{[l]}_{i_k} = y_{i_k}^{[l]} - \frac{1}{|\mathcal{B}_k|} \sum_{i_k \in \mathcal{B}_k} y_{i_k}^{[l]}, \tag{11}$$

$$y''^{[l]}_{i_k} = y'^{[l]}_{i_k} \Big/ \sqrt{\frac{1}{|\mathcal{B}_k|} \sum_{i_k \in \mathcal{B}_k} \left(\bar{y}_{i_k}^{[l]}\right)^2}, \tag{12}$$

$$x_{i_k}^{[l+1]} = \mathbb{I}(\mathrm{SC}) x_{i_k}^{[l]} + \delta(y''^{[l]}_{i_k}), \tag{13}$$

where $\delta(\cdot)$ is an activation function. The shift and scale parameters are omitted for simplicity.

Although CBN is similar to vanilla BN, there are still significant distinctions, as shown in Figure 1. *First*, the placement positions are different. Vanilla BN is placed behind the linear activation layer, while CBN is located in the front of the linear activation layer. *Second*, mean removal and normalization with the ℓ_2 norm are tangled in vanilla BN, but the two parts are decoupled in CBN. *Third*, we add a hyperparameter ρ in CBN to tune the magnitude of normalization to achieve better performance. *Fourth*, we adopt adaptive moving average statistics in CBN to exploit historical information rather than statistics over the single current batch.

Since each step of CBN in Eqs. (4-7) is theoretically justified, we can analyze the effectiveness and weakness of BN via the differences between BN and CBN. BN is applied after the linear activation and before the nonlinear activation function. Hence, from our theoretical analysis in Section 4, BN actually takes effect for the linear activation in the next block. Due to the nonlinearity of the activation function, the benefits of normalization are impaired. However, when ReLU is used, the degradation is not so serious. The reason for this is presented as follows. Due to mean removal in Eq. (11), the intermediate features $\{y''^{[l]}_{i_k}\}_{i_k \in \mathcal{B}_k}$ are commonly symmetrical around zero, such that the following holds:

$$x_{i_k}^{[l+1]} = \delta\left(\frac{y''^{[l]}_{i_k}}{\sqrt{\frac{1}{|\mathcal{B}_k|}\sum_{i_k\in\mathcal{B}_k}\left(y''^{[l]}_{i_k}\right)^2}}\right) = \frac{\delta(y''^{[l]}_{i_k})}{\sqrt{\frac{2}{|\mathcal{B}_k|}\sum_{i_k\in\mathcal{B}_k}\left(\delta(y''^{[l]}_{i_k})\right)^2}}. \tag{14}$$

When used in conjunction with ReLU, vanilla BN can be seen as equivalent to performing ℓ_2-norm normalization with a normalization magnitude of $\rho = \sqrt{2}$ for CBN. This means that regardless of whether the normalization is performed before or after the ReLU activation, the effects of ℓ_2-norm normalization are the same, but with a scaling factor difference. According to Theorem 3, this approach can effectively reduce the gradient Lipchitz constant and the stochastic gradient squared expectation. Moreover, placing vanilla BN before ReLU usually outperforms placing it after ReLU without tuning ρ, and this is because the scale factor $\sqrt{2}$ is naturally suited to ReLU-like nonlinearities. However, this coincidence does not hold for other types of nonlinearities. On the other hand, the position and construction of CBN is designed based on theory and offers a more general guarantee of effectiveness. This is confirmed by experimental results in Section 6, which show that the more a nonlinearity differs from ReLU, the greater the improvement CBN provides over vanilla BN.

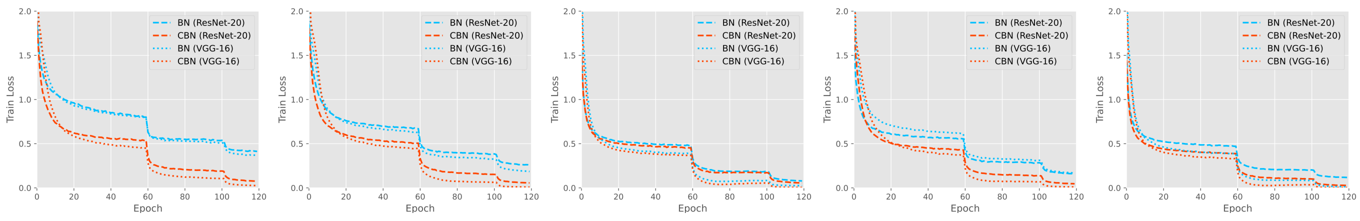
Remark 4. The prior work of Santurkar et al [12] demonstrated that the effectiveness of vanilla BN is due to its ability to improve the Lipschitzness and "effective" β-smoothness of the loss. However, our work makes new contributions to deepen the understanding of batch normalization and identify a direction for further improvements. *Firstly*, while [12] shows that vanilla BN enhances the Lipschitzness and β-smoothness of the loss, it remains unclear how these effects lead to faster convergence and smaller converged minima, which are the primary benefits of vanilla BN. In contrast, we adopt a top-down strategy, deducing sufficient conditions for faster convergence and smaller converged minima when stochastically optimizing a general non-convex problem. We then construct CBN, which can elicit all these effects. *Secondly*, in [12], the authors omit non-linear activation when performing theoretical analysis. However, our theoretical analysis considers non-linear activation and shows that normalization should be placed after the non-linearity. Our extensive experiments have demonstrated the ineffectiveness of pre-activation vanilla BN and the validity of post-activation CBN. *Thirdly*, [12] only analyzes a fixed block with or without vanilla BN, assuming that the input and output of a fixed block are the same in both cases. However, this idealized assumption deviates from reality, as the outputs of the block with or without BN will be different, and the inputs of the next block with or without BN will also be inequable. In comparison, we analyze the entire network with multiple blocks, and our analysis does not require the inputs and outputs of the intermediate blocks to be the same.

6 Experiments

6.1 General Settings

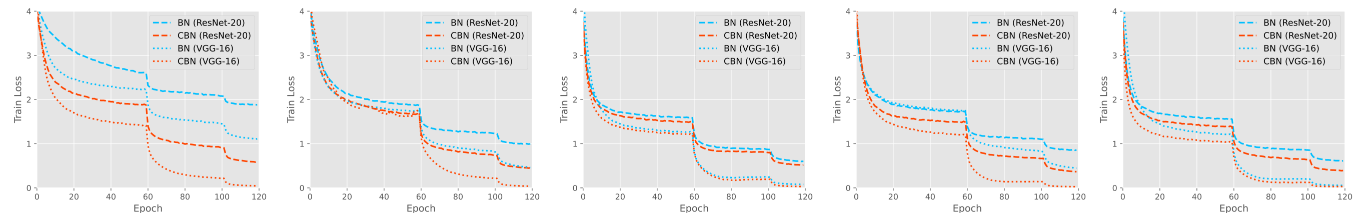
We assess the performances of vanilla BN and CBN when embedding in widely-used VGG-Net [47], ResNet [48] and EfficientNet [49] on the CIFAR10, CIFAR100 and ILSVRC2012 benchmark datasets in this section.

For the experiments on CIFAR10 and CIFAR100, the experimental settings are the same as that in the original paper proposing ResNet [48] except for the total iterations. We sample a set of 128 examples with replacement for each batch. SGD is adopted with a momentum of 0.9 and a



(a) Sigmoid (b) Tanh (c) ReLU (d) SELU (e) Swish

Fig. 4. Visual comparison of training losses for BN and CBN embedded ResNet-20 and VGG-16 on CIFAR10 with different activation functions: (a) Sigmoid, (b) Tanh, (c) ReLU, (d) SELU, and (e) Swish.



(a) Sigmoid (b) Tanh (c) ReLU (d) SELU (e) Swish

Fig. 5. Visual comparison of training losses for BN and CBN embedded ResNet-20 and VGG-16 on CIFAR100 with different active functions: (a) Sigmoid, (b) Tanh, (c) ReLU, (d)SELU, and (e) Swish.

TABLE 1
Comparison of mean test accuracy and standard deviation of 10 trials for BN and CBN embedded ResNet-20 with different activation functions on CIFAR10.

Methods	Before Opera.	Decouple	Tune ρ	AMAT	ResNet-20				
					Sigmoid	Tanh	ReLU	SELU	Swish
① BN (Baseline)	-	-	-	-	78.97 $\pm$ 0.53	87.59 $\pm$ 0.12	91.13 $\pm$ 0.09	89.69 $\pm$ 0.11	91.24 $\pm$ 0.11
② CBN	✓	✓	✓	✓	91.20 $\pm$ 0.06	91.73 $\pm$ 0.07	91.98 $\pm$ 0.06	91.57 $\pm$ 0.08	92.16 $\pm$ 0.04
Improv.					(+12.13)	(+4.14)	(+0.85)	(+1.88)	(+0.92)
③	✓	-	-	-	88.78 $\pm$ 0.11	89.85 $\pm$ 0.06	91.27 $\pm$ 0.16	91.08 $\pm$ 0.16	91.72 $\pm$ 0.13
					(+9.81)	(+2.26)	(+0.14)	(+1.39)	(+0.48)
④	✓	✓	-	-	89.75 $\pm$ 0.13	91.29 $\pm$ 0.05	91.50 $\pm$ 0.11	91.17 $\pm$ 0.04	91.89 $\pm$ 0.06
					(+10.78)	(+3.70)	(+0.37)	(+1.48)	(+0.65)
⑤	✓	✓	✓	-	90.93 $\pm$ 0.11	91.39 $\pm$ 0.10	91.58 $\pm$ 0.07	91.45 $\pm$ 0.12	91.93 $\pm$ 0.08
					(+11.96)	(+3.80)	(+0.45)	(+1.76)	(+0.69)
⑥	-	-	✓	-	79.75 $\pm$ 0.21	87.64 $\pm$ 0.08	91.53 $\pm$ 0.18	90.13 $\pm$ 0.06	91.32 $\pm$ 0.11
					(+0.98)	(+0.05)	(+0.40)	(+0.44)	(+0.08)
⑦	-	-	✓	✓	80.27 $\pm$ 0.16	87.77 $\pm$ 0.06	91.67 $\pm$ 0.08	90.31 $\pm$ 0.04	91.52 $\pm$ 0.09
					(+1.30)	(+0.18)	(+0.54)	(+0.62)	(+0.28)

TABLE 2
Comparison of mean test accuracy and standard deviation of 10 trials for BN and CBN embedded VGG-16 with different activation functions on CIFAR10.

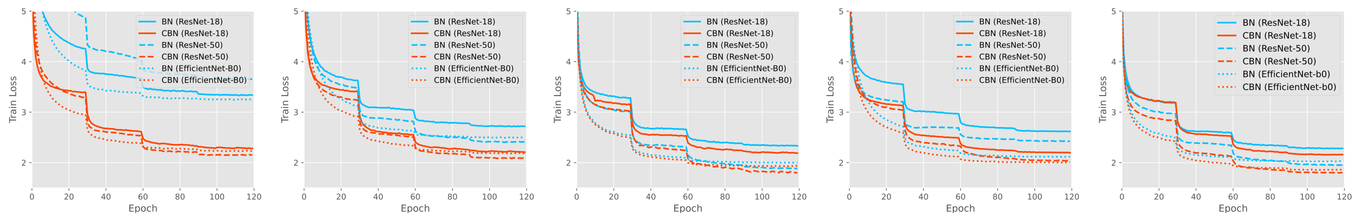
Methods	Before Opera.	Decouple	Tune ρ	AMAT	VGG-16				
					Sigmoid	Tanh	ReLU	SELU	Swish
① BN, Baseline	-	-	-	-	82.38 $\pm$ 0.56	88.80 $\pm$ 0.08	93.12 $\pm$ 0.09	89.60 $\pm$ 0.05	92.58 $\pm$ 0.11
② BN	✓	✓	✓	✓	92.66 $\pm$ 0.08	92.94 $\pm$ 0.08	93.86 $\pm$ 0.06	93.81 $\pm$ 0.08	94.07 $\pm$ 0.07
Improv.					(+10.26)	(+4.15)	(+0.73)	(+4.21)	(+1.49)
③	✓	-	-	-	91.40 $\pm$ 0.16	92.81 $\pm$ 0.08	93.24 $\pm$ 0.05	93.02 $\pm$ 0.08	93.48 $\pm$ 0.06
					(+9.02)	(+4.01)	(+0.12)	(+3.42)	(+0.90)
④	✓	✓	-	-	92.13 $\pm$ 0.10	92.77 $\pm$ 0.11	93.32 $\pm$ 0.13	93.21 $\pm$ 0.05	93.63 $\pm$ 0.11
					(+9.75)	(+3.97)	(+0.20)	(+3.61)	(+1.05)
⑤	✓	✓	✓	-	92.31 $\pm$ 0.11	92.85 $\pm$ 0.13	93.51 $\pm$ 0.11	93.44 $\pm$ 0.12	93.71 $\pm$ 0.13
					(+9.93)	(+4.05)	(+0.39)	(+3.84)	(+1.13)
⑥	-	-	✓	-	86.57 $\pm$ 0.11	89.70 $\pm$ 0.09	93.44 $\pm$ 0.06	89.86 $\pm$ 0.09	92.76 $\pm$ 0.11
					(+4.19)	(+0.90)	(+0.32)	(+0.26)	(+0.18)
⑦	-	-	✓	✓	86.69 $\pm$ 0.08	89.84 $\pm$ 0.06	93.63 $\pm$ 0.05	89.98 $\pm$ 0.05	92.90 $\pm$ 0.12
					(+4.31)	(+1.04)	(+0.51)	(+0.39)	(+0.32)

TABLE 3
Comparison of mean test accuracy and standard deviation of 10 trials for BN and CBN embedded ResNet-20 with different activation functions on CIFAR100.

Methods	Before Opera.	Decouple	Tune ρ	AMAT	ResNet-20 Sigmoid	Tanh	ReLU	SELU	Swish
① BN (Baseline)	-	-	-	-	46.05 $\pm$ 0.20	63.15 $\pm$ 0.09	67.21 $\pm$ 0.12	63.81 $\pm$ 0.14	67.82 $\pm$ 0.10
② CBN	✓	✓	✓	✓	66.92 $\pm$ 0.08	67.13 $\pm$ 0.07	68.55 $\pm$ 0.05	67.43 $\pm$ 0.08	69.52 $\pm$ 0.16
Improv.					(+20.87)	(+3.98)	(+1.34)	(+3.62)	(+1.70)
③	✓	-	-	-	65.06 $\pm$ 0.09	66.42 $\pm$ 0.11	66.78 $\pm$ 0.12	66.13 $\pm$ 0.09	67.25 $\pm$ 0.12
					(+19.01)	(+3.09)	(-0.43)	(+2.32)	(-0.57)
④	✓	✓	-	-	66.64 $\pm$ 0.08	66.24 $\pm$ 0.10	67.90 $\pm$ 0.08	66.22 $\pm$ 0.16	68.31 $\pm$ 0.13
					(+20.59)	(+3.27)	(+0.69)	(+2.41)	(+0.49)
⑤	✓	✓	✓	-	66.82 $\pm$ 0.11	67.13 $\pm$ 0.15	68.28 $\pm$ 0.09	66.40 $\pm$ 0.09	69.17 $\pm$ 0.15
					(+20.77)	(+3.98)	(+1.07)	(+2.59)	(+1.35)
⑥	-	-	✓	-	48.82 $\pm$ 0.24	63.26 $\pm$ 0.14	67.38 $\pm$ 0.11	65.03 $\pm$ 0.14	67.93 $\pm$ 0.10
					(+2.77)	(+0.11)	(+0.17)	(+1.22)	(+0.11)
⑦	-	-	✓	✓	49.88 $\pm$ 0.15	63.45 $\pm$ 0.10	67.51 $\pm$ 0.09	65.25 $\pm$ 0.08	68.01 $\pm$ 0.11
					(+3.83)	(+0.40)	(+0.30)	(+1.44)	(+0.19)

TABLE 4
Comparison of mean test accuracy and standard deviation of 10 trials for BN and CBN embedded VGG-16 with different activation functions on CIFAR100.

Methods	Before Opera.	Decouple	Tune ρ	AMAT	VGG-16 Sigmoid	Tanh	ReLU	SELU	Swish
① BN (Baseline)	-	-	-	-	60.37 $\pm$ 0.10	66.54 $\pm$ 0.15	72.56 $\pm$ 0.07	65.52 $\pm$ 0.17	71.78 $\pm$ 0.08
② CBN	✓	✓	✓	✓	72.41 $\pm$ 0.07	72.12 $\pm$ 0.08	73.72 $\pm$ 0.06	71.39 $\pm$ 0.07	73.93 $\pm$ 0.08
Improv.					(+12.04)	(+5.58)	(+1.16)	(+5.87)	(+2.15)
③	✓	-	-	-	70.03 $\pm$ 0.10	70.89 $\pm$ 0.09	72.09 $\pm$ 0.07	70.18 $\pm$ 0.11	72.15 $\pm$ 0.10
					(+9.66)	(+4.35)	(-0.47)	(+4.66)	(+0.37)
④	✓	✓	-	-	71.01 $\pm$ 0.08	71.16 $\pm$ 0.10	72.86 $\pm$ 0.09	70.27 $\pm$ 0.17	72.79 $\pm$ 0.11
					(+10.64)	(+4.62)	(+0.30)	(+4.75)	(+1.01)
⑤	✓	✓	✓	-	71.44 $\pm$ 0.11	71.53 $\pm$ 0.12	73.45 $\pm$ 0.12	70.60 $\pm$ 0.09	73.11 $\pm$ 0.12
					(+11.07)	(+4.99)	(+0.89)	(+5.08)	(+1.33)
⑥	-	-	✓	-	63.33 $\pm$ 0.19	66.89 $\pm$ 0.17	72.77 $\pm$ 0.14	66.57 $\pm$ 0.25	71.97 $\pm$ 0.15
					(+2.96)	(+0.35)	(+0.21)	(+1.05)	(+0.09)
⑦	-	-	✓	✓	64.09 $\pm$ 0.08	66.98 $\pm$ 0.08	72.91 $\pm$ 0.06	66.80 $\pm$ 0.15	72.05 $\pm$ 0.08
					(+3.72)	(+0.44)	(+0.35)	(+1.28)	(+0.17)



(a) Sigmoid (b) Tanh (c) ReLU (d) SELU (e) Swish

Fig. 6. Visual comparison of training losses for BN and CBN embedded in ResNet-18, ResNet-50, and EfficientNet-B0 on ILSVRC2012 with different active functions: (a) Sigmoid, (b) Tanh, (c) ReLU, (d)SELU, and (d) Swish.

TABLE 5
Comparison of top-1 test accuracy for BN and CBN embedded ResNet-18, ResNet-50 and EfficientNet-B0 with different activation activation functions on ILSVRC2012.

	ResNet-18					ResNet-50					EfficientNet-B0				
	Sigmoid	Tanh	ReLU	SELU	Swish	Sigmoid	Tanh	ReLU	SELU	Swish	Sigmoid	Tanh	ReLU	SELU	Swish
BN	52.03	64.45	70.02	65.46	70.81	44.38	69.35	76.02	69.68	76.21	55.04	67.95	73.71	69.51	74.61
CBN	70.15	70.12	70.43	71.10	71.39	72.82	73.93	76.38	74.69	76.75	70.36	71.47	74.05	72.57	75.08
Imp.	(+18.12)	(+5.67)	(+0.41)	(+5.64)	(+0.58)	(+28.44)	(+4.58)	(+0.36)	(+5.01)	(+0.54)	(+15.32)	(+3.52)	(+0.34)	(+3.06)	(+0.47)

weight decay of 0.0005. To simplify the tuning process and ensure fair comparisons, in each case, we start with the same learning rate of 0.1, divide the learning rate by 10 after 60 and 100 epochs, and finally terminate the procedure after 120 epochs. For the experiments on ILSVRC2012, SGD is adopted with a momentum of 0.9 and a weight decay of 0.0001. We sample a set of 256 examples with replacement for each batch, and each batch is uniformly distributed across 8 GPUs. The learning rate starts at 0.1 and is divided by 10 after 30, 60 and 90 epochs; training is finally terminated after 120 epochs. The data augmentation and implementation follow the PyTorch official codes at https://github.com/pytorch/examples/tree/main/imagenet.

6.2 Comparison with Vanilla BN

We evaluate the performance of the proposed CBN in comparison with that of vanilla BN based on the training convergence speeds and the inference classification accuracies achieved on CIFAR10, CIFAR100 and ILSVRC2012. Note that when applying AMAT, we simply set the exponential parameter in Eq. (9) to $\eta_k = \max(1 - 10 \times lr, 0)$. The results are reported in Figures 4-6 and Tables 1-5.

In comparison with vanilla BN, CBN can make the training loss converge *faster* and the final converged value *smaller* (Figures 4-5), and then it enjoys *better* inference classification accuracy (please see the first two lines in Tables 1-4). Specifically, benefitting from CBN, the training convergence for networks with Sigmoid, Tanh and SELU improves considerably in terms of speed the final converged values, with the test accuracy improvements of more than 13%, 4%, 3.5% on average, respectively. Compared with vanilla BN, the improvement for CBN with ReLU is not so substantial as that with Sigmoid, Tanh and SELU. The reason for this has been elaborately demonstrated in Section 4. When we apply ReLU, placing normalization before or after the linear activation is roughly equivalent, while applying Sigmoid or Tanh, the position of normalization plays a key role in boosting performance. However, when ReLU changes slightly, such as Swish, the equivalence of placing normalization before or after the linear activation is also slightly impaired. Therefore, as illustrated in Figures Fig.2-5 and Tables 1-4, CBN with Swish can receive greater improvement in terms of its training convergence speed and inference accuracy than CBN with ReLU (approximately 1.6% versus 1.0% on average). In other words, our extensive experiments have confirmed that the less the nonlinearity resembles ReLU, the more obvious the improvement of CBN over vanilla BN becomes; this situation is in accordance with the theoretical analysis in Section 5. Notably, with the help of CBN, the performance of the networks with Sigmoid and Tanh are fairly close to that of the networks with ReLU, thus further verifying the power of the proposed approach.

There are four main different features in CBN – placing normalization before the linear activation, decoupling the mean removal and normalization process, tuning the best normalization parameter ρ and adopting AMAT during training. To validate the effectiveness of each feature, we also implement experiments with intermediate methods in which BN with one or more features. As displayed in Tables 1-4, placing BN before the linear activation (③) achieves substantial improvements when Sigmoid, Tanh or SELU is equipped. However, when ReLU or Swish is applied, the performance of ③ is even inferior to vanilla BN (comparing ① (vanilla BN) and ③). No matter whether the normalization approach is performed before or after ReLU, the effects of normalization with the ℓ_2 norm are the same (except for a scaled factor $\sqrt{2}$), and only the scaling factor makes BN slightly outperform ③. Decoupling the mean removal and the normalization with ℓ_2-norm yields further improvement(comparing ③ and ④). With the assistance of the tuned ρ, a significant gain in test accuracy is achieved (comparing ① (vanilla BN) and ⑥, ④ and ⑤), which experimentally demonstrates the effectiveness of tuning normalization magnitude. When exploiting AMAT, the modified method steadily increases the test accuracy (comparing ② (CBN) and ⑤, ⑥ and ⑦). Interestingly, compared with ① (vanilla BN) and ⑥, the standard deviations for ② (CBN) and ⑦ are decreased, which implies that adaptive moving average can reduce the variance.

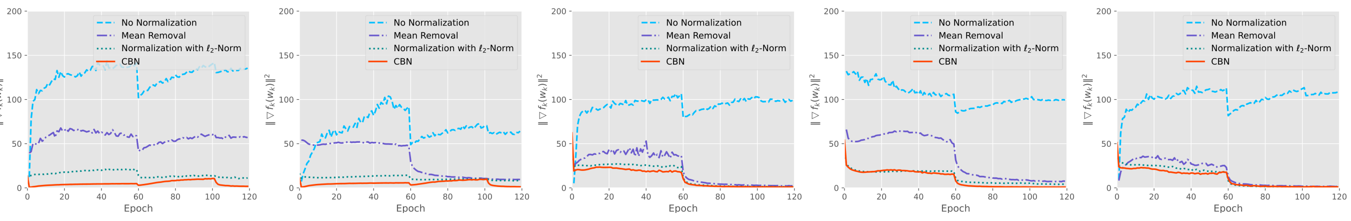
We also report the results of classification experiments performed on the large-scale ILSVRC2012 dataset. As shown in Figure 6 and Table 5, the training convergence rate for CBN is also consistently faster than that for BN, and the test accuracy achieved with CBN is also higher than that achieved with BN, demonstrating that CBN is also effective on large-scale datasets. Specifically, when Sigmoid, Tanh or SELU is applied, CBN substantially boosts the classification performance by *more than* 20, 5% *and* 5%, *respectively* on average, respectively. When ReLU is employed, the test accuracy improvement is not so obvious (approximately 0.4%), but it is still completive to the-state-of-art modified batch normalization [9], [19], [42]. Similar to the experimental results obtained on CIFAR10 and CIFAR100, compared CBN with ReLU, the gains of CBN with Swish in terms of the training convergence and the inference accuracy are also boosted, which experimentally validate theoretical analysis in Section 5. Notably, ReLU is originally employed in ResNet [48], and Swish is specifically equipped for EfficientNet [49], but the gains from the use of CBN with ReLU and Swish for both ResNet and EfficietNet are not obviously distinguishable, which indicates the benefits of CBN may be independent of the use of specific network architectures.

6.3 Performance Analysis for Components

We perform ablation experiments with ResNet-20 on CIFAR100 to clarify the contributions of mean removal and the normalization with the ℓ_2-norm of CBN. We find the maximal learning rate (lr) from the candidate set $\{0.0001, 0.001, 0.01, 0.1, 1\}$ that ensures that training does not collapse for the four approaches (no normalization, mean removal, normalization with ℓ_2-norm and CBN). From Theorem 1, we know that the maximal lr is inversely proportional to the gradient Lipschitz constant, *i.e.*, $lr_{\max} = \frac{1}{L}$. Hence, the results in Table 6 imply that both mean removal and normalization with the ℓ_2-norm can help to reduce the gradient Lipschitz constant, but normalization with ℓ_2-norm contributes more, which experimentally validates the first conclusion in Theorem 2 and Theorem 3. Moreover, as shown in Figure 7, mean removal and normalization with ℓ_2-norm are also beneficial to reduction in the stochastic gradient squared expectation, experimentally validating the second

TABLE 6
Ablation study on mean test accuracy and standard deviation of 10 trials with ResNet-20 on CIFAR10.

Mean Removal	Normalization	Max. lr	Test Accuracy Sigmoid	Tanh	ReLU	SELU	Swish
-	-	0.001	35.96 $\pm$ 0.24	69.67 $\pm$ 0.06	80.40 $\pm$ 0.55	77.15 $\pm$ 0.42	81.37 $\pm$ 0.23
✓	-	0.01	63.99 $\pm$ 0.36	87.30 $\pm$ 0.13	89.99 $\pm$ 0.06	87.61 $\pm$ 0.14	90.01 $\pm$ 0.07
-	✓	0.1	81.22 $\pm$ 0.23	88.65 $\pm$ 0.17	91.33 $\pm$ 0.12	90.76 $\pm$ 0.08	91.17 $\pm$ 0.06
✓	✓	0.1	91.20 $\pm$ 0.05	91.43 $\pm$ 0.07	91.98 $\pm$ 0.06	91.57 $\pm$ 0.08	92.16 $\pm$ 0.04



(a) Sigmoid (b) Tanh (c) ReLU (d) SELU (e) Swish

Fig. 7. Visual comparison of gradient norm for CBN components embedded ResNet-20 on CIFAR10 with different active functions: (a) Sigmoid, (b) Tanh, (c) ReLU, (d) SELU, and (e) Swish.

conclusion in Theorem 2 and Theorem 3. This then leads to the higher testing accuracy shown in Table 6. Normalization with the ℓ_2-norm still performs better than mean removal in this case. More importantly, the results of CBN (mean removal + normalization with ℓ_2-norm) demonstrate mean removal and normalization together can generate the effect of "$1 + 1 > 2$", especially when Sigmoid, Tanh or SELU is applied.

6.4 Sensitivity Analysis For ρ

We add a scaling factor ρ to magnify the normalization magnitude for CBN. In this subsection, we analyze influences of different values of the scaling factor ρ: $\{0.25, 0.5, 1, 2, 4, 8\}$ for CBN embedded ResNet-20.

As shown in Figures 8-9, when ρ is moderately large, it is not only beneficial for speeding up training but also helps the training loss to ultimately converge to a smaller value so that CBN achieves higher test accuracy. However, if we further increase ρ to a large value, although the convergence speed is further accelerated, the training loss is more likely to converge to a larger local minimum. The reason for this is discussed in Remark 2 of Section 4. An excessively large ρ causes the activation function (tanh or sigmoid) to operate predominantly in the linear regime, which further harms the representation capability of the neural network, making it difficult for the training loss to reach a better local minimum. An excessively large ρ also lowers the dynamic range since machine precision is finite. On the other hand, as displayed in Figure 8, when ρ is too small, the training convergence becomes slow and unstable.

Another interesting phenomenon is that the best values of ρ among CBN with ReLU, Tanh and Sigmoid become larger in order. ReLU, Tanh and Sigmoid compress the inputs greater in order, so the outputs of ReLU, Tanh and Sigmoid, which are also the inputs of CBN, successively become smaller. The best value of ρ in CBN with Sigmoid is even smaller than 1.

From Figures 8-9, we know the performance for CBN with ReLU, SELU or Swish is robust to the varying range of ρ. Although the performance for CBN with Sigmoid or Tanh is sensitive to the value of ρ, CBN with Sigmoid or Tanh achieves better performance in the worst-case scenario than vanilla with Sigmoid or Tanh. Therefore, if we simply set $\rho = 1$ as the default value, CBN will still outperform vanilla BN.

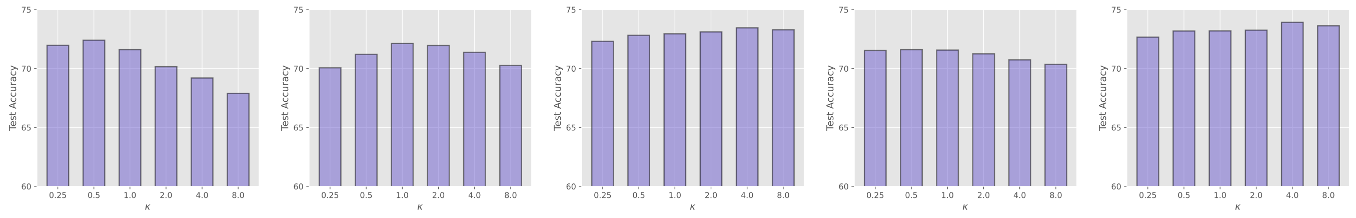
6.5 Variance Reduction Analysis for AMAT

Previous works have leveraged moving average statistics during training. Our contribution is not initially exploiting moving average statistics. The concept "adaptive" is just the core of our method to receive better performance due to resulting in stochastic gradient variance reduction. From our theoretical analysis, AMAT adaptively adjusts the moving parameter η by carefully tracking the learning rate to help reduce the variance the stochastic gradient. In contrast, fixing the moving parameter η may degrade rather than improve the performance. To verify this claim, we implement comparison experiments including vanilla BN, CBN without AMAT, CBN with a fixed moving parameter and CBN with AMAT embedded in ResNet-18 on ILSVRC2012. Notably, when applying the adaptive moving average scheme, we simply set the exponential parameter $\eta_k = \max(1 - 10 \times lr, 0)$, and when adopting the fixed moving average parameter η, we simply set $\eta = 0.9$. We also set $\eta = 0.9$ for vanilla BN.

igure 10 illustrates that CBN with AMAT offers significant advantages over vanilla BN and CBN without AMAT. Specifically, using CBN with AMAT reduces both the stochastic gradient variance and the gradient itself. However, it is worth noting that fixing the moving parameter η does not necessarily decrease the stochastic gradient variance. In fact, fixing η may increase bias, which negatively affects performance. These results suggest that CBN with AMAT is a valuable technique for improving the performance of deep learning models.

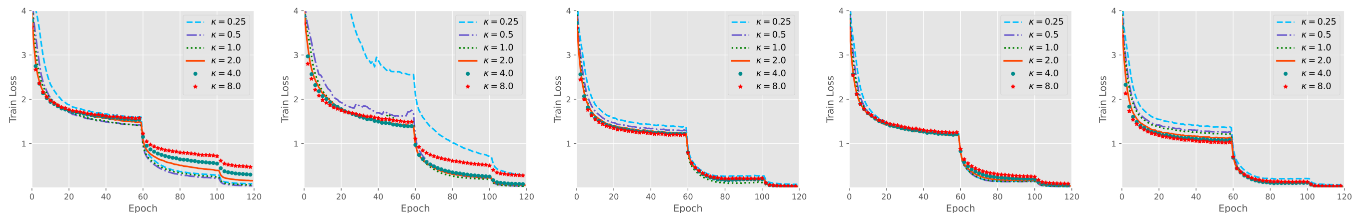
7 Conclusion and Discussion

The practical success of vanilla BN in accelerating training convergence and improving inference performance is indis-



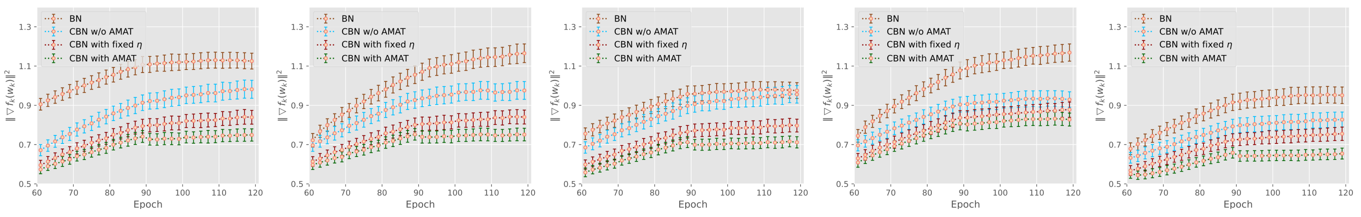
(a) Sigmoid (b) Tanh (c) ReLU (d) SELU (e) Swish

Fig. 8. Sensitivity analysis of training losses for CBN with different ρ embedded VGG-16 on CIFAR100 combining with different active functions: (a) Sigmoid, (b) Tanh, (c) ReLU, (d) SELU, and (e) Swish.



(a) Sigmoid (b) Tanh (c) ReLU (d) SELU (e) Swish

Fig. 9. Sensitivity analysis of training losses for CBN with different ρ embedded VGG-16 on CIFAR100 combining with different active functions: (a) ReLU, (b) Tanh, (c) ReLU, (d) SELU, and (e) Swish.



(a) Sigmoid (b) Tanh (c) ReLU (d) SELU (e) Swish

Fig. 10. Visual comparison of the variance of the stochastic gradient for CBN without AMAT, with fixed moving parameter and with AMAT embedded ResNet-18 on ILSVRC2012 combining different active functions: (a) Sigmoid, (b) Tanh, (c) ReLU, (d) SELU, and (e) Swish.

putable. However, the mechanism behind its effectiveness is not fully understood. In this paper, we do not directly try to theoretically justify vanilla BN and then improve it. In contrast, we derive the factors that influence convergence rate and the final converged minimum, and used this knowledge to construct a new normalization approach, called CBN, which is proven to induce these factors. We then compared CBN to vanilla BN, which allowed us to assess the effectiveness of vanilla BN. Our analysis show that vanilla BN is only effective when used with ReLU, and that even minor changes to ReLU, such as Swish, can result in a decline in performance.We conducted extensive experiments that validate our theoretical analyses and demonstrate that CBN is superior to vanilla BN. Specifically, when Sigmoid Tanh, SELU is applied, CBN boost network classification accuracy by an average of 15%, 4% and 3.5%, respectively, compared to vanilla BN.

Theoretical work by Lipschitz (2018) shows that it is NP-hard to exactly estimate the Lipschitz constant, which has important implications for estimating the gradient Lipschitz constant and the stochastic gradient squared expectation. Therefore, it is more realistic to estimate their upper bounds, which are still theoretically and practically significant. Tightening these upper bounds is an avenue for future research.

Normalization approaches can be thought of as optimization algorithms that modify the structure of machine learning models to accelerate convergence. This is in contrast to conventional gradient descent algorithms that work on gradients. By modulating the structure of the model, we can potentially develop promising new optimization approaches, which we plan to explore further in future research.

Acknowledgments

This work is partially supported by the National R&D Program of China (Grant No. 2022ZD0115301), the Major Key Project of PCL (Grant No. PCL2023AS7-1), the National Natural Science Foundation of China (Grant No. 61976144), and the Shenzhen International Research Cooperation Project (Grant No. GJHZ20220913142611021).

References

[1] S. Ioffe and C. Szegedy, "Batch normalization: accelerating deep network training by reducing internal covariate shift," in *International Conference on International Conference on Machine Learning*, 2015.

[2] J. L. Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," *arXiv preprint: 1607.06450v1*, 2016.

[3] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in neural information processing systems*, 2017, pp. 5998–6008.

[4] D. Ulyanov, A. Vedaldi, and V. Lempitsky, "Instance normalization: The missing ingredient for fast stylization," *arXiv preprint: 1607.08022*, 2016.

[5] T. Miyato, T. Kataoka, M. Koyama, and Y. Yoshida, "Spectral normalization for generative adversarial networks," *arXiv preprint arXiv:1802.05957*, 2018.

[6] Y. Li, N. Wang, J. Shi, X. Hou, and J. Liu, "Adaptive batch normalization for practical domain adaptation," *Pattern Recognition*, vol. 80, pp. 109–117, 2018.

[7] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick, "Momentum contrast for unsupervised visual representation learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 9729–9738.

[8] Y. Wu and K. He, "Group normalization," in *European Conference on Computer Vision*, 2018.

[9] S. Ioffe, "Batch renormalization: Towards reducing minibatch dependence in batch-normalized models," in *Advances in Neural Information Processing Systems*, 2017.

[10] J. Yan, R. Wan, X. Zhang, W. Zhang, Y. Wei, and J. Sun, "Towards stabilizing batch statistics in backward propagation of batch normalization," in *International Conference on Learning Representations*, 2020.

[11] B. R. Ali Rahimi, "Back when we were kid," *NIPS Test of Time Award*, 2016.

[12] S. Santurkar, D. Tsipras, A. Ilyas, and A. Madry, "How does batch normalization help optimization? (no, it is not about internal covariate shift)," in *Advances in Neural Information Processing Systems*, 2018.

[13] B. Ghorbani, S. Krishnan, and Y. Xiao, "An investigation into neural net optimization via hessian eigenvalue density," in *International Conference on Machine Learning*. PMLR, 2019, pp. 2232–2241.

[14] L. Huang, J. Qin, L. Liu, F. Zhu, and L. Shao, "Layer-wise conditioning analysis in exploring the learning dynamics of dnns," in *European Conference on Computer Vision*. Springer, 2020, pp. 384–401.

[15] T. Cooijmans, N. Ballas, C. Laurent, Ç. Gülçehre, and A. Courville, "Recurrent batch normalization," *arXiv preprint arXiv:1603.09025*, 2016.

[16] Z. Kou, K. You, M. Long, and J. Wang, "Stochastic normalization," in *Advances in Neural Information Processing Systems*, vol. 33, 2020.

[17] C. Peng, T. Xiao, Z. Li, Y. Jiang, X. Zhang, K. Jia, G. Yu, and J. Sun, "Megdet: A large mini-batch object detector," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 6181–6189.

[18] T. Salimans and D. P. Kingma, "Weight normalization: A simple reparameterization to accelerate training of deep neural networks," in *Advances in Neural Information Processing Systems*, 2016.

[19] Y. Guo, Q. Wu, C. Deng, J. Chen, and M. Tan, "Double forward propagation for memorized batch normalization," in *Association for the Advancement of Artificial Intelligence*, 2018.

[20] V. Chiley, I. Sharapov, A. Kosson, U. Koster, R. Reece, S. Samaniego de la Fuente, V. Subbiah, and M. James, "Online normalization for training neural networks," in *Advances in Neural Information Processing Systems*, vol. 32. Curran Associates, Inc., 2019, pp. 8433–8443.

[21] S. Jia, D.-J. Chen, and H.-T. Chen, "Instance-level meta normalization," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[22] S. Liang, Z. Huang, M. Liang, and H. Yang, "Instance enhancement batch normalization: An adaptive regulator of batch noise," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 34, 2020, pp. 4819–4827.

[23] S.-H. Gao, Q. Han, D. Li, M.-M. Cheng, and P. Peng, "Representative batch normalization with feature calibration," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2021, pp. 8669–8679.

[24] A. Labatie, D. Masters, Z. Eaton-Rosen, and C. Luschi, "Proxy-normalizing activations to match batch normalization while removing batch dependence," in *Advances in Neural Information Processing Systems*, vol. 34, 2021, pp. 16 990–17 006.

[25] J. Bjorck, C. Gomes, B. Selman, and K. Q. Weinberger, "Understanding batch normalization," *arXiv preprint arXiv:1806.02375*, 2018.

[26] J. Kohler, H. Daneshmand, A. Lucchi, T. Hofmann, M. Zhou, and K. Neymeyr, "Exponential convergence rates for batch normalization: The power of length-direction decoupling in non-convex optimization," in *The 22nd International Conference on Artificial Intelligence and Statistics*, 2019, pp. 806–815.

[27] S. Arora, Z. Li, and K. Lyu, "Theoretical analysis of auto rate-tuning by batch normalization," *arXiv preprint arXiv:1812.03981*, 2018.

[28] H. Daneshmand, J. Kohler, F. Bach, T. Hofmann, and A. Lucchi, "Batch normalization provably avoids ranks collapse for randomly initialised deep networks," in *Advances in Neural Information Processing Systems*, vol. 33, 2020, pp. 18 387–18 398.

[29] E. S. Lubana, R. Dick, and H. Tanaka, "Beyond batchnorm: towards a unified understanding of normalization in deep learning," in *Advances in Neural Information Processing Systems*, vol. 34, 2021, pp. 4778–4791.

[30] M. D. Zeiler, "ADADELTA: an adaptive learning rate method," *arXiv preprint: 1212.5710*, 2012.

[31] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *International Conference on Learning Representations*, 2015.

[32] S. J. Reddi, S. Kale, and S. Kumar, "On the convergence of adam and beyond," in *International Conference on Learning Representations*, 2018.

[33] Z. Allen-Zhu and E. Hazan, "Variance reduction for faster non-convex optimization," in *International Conference on Machine Learning*, 2016.

[34] C. Fang, C. J. Li, Z. Lin, and T. Zhang, "Spider: Near-optimal non-convex optimization via stochastic path-integrated differential estimator," in *Advances in Neural Information Processing Systems*, 2018.

[35] Z. Wang, K. Ji, Y. Zhou, Y. Liang, and V. Tarokh, "Spiderboost and momentum: Faster variance reduction algorithms," in *Advances in Neural Information Processing Systems*, 2019.

[36] J. Bernstein, Y.-X. Wang, K. Azizzadenesheli, and A. Anandkumar, "Signsgd: Compressed optimisation for non-convex problems," in *International Conference on Machine Learning*, 2018, pp. 560–569.

[37] A. Reisizadeh, A. Mokhtari, H. Hassani, A. Jadbabaie, and R. Pedarsani, "Fedpaq: A communication-efficient federated learning method with periodic averaging and quantization," in *International Conference on Artificial Intelligence and Statistics*, 2020, pp. 2021–2031.

[38] X. Chen, S. Liu, R. Sun, and M. Hong, "On the convergence of a class of adam-type algorithms for non-convex optimization," in *International Conference on Learning Representations*, 2019.

[39] A. Cutkosky and F. Orabona, "Momentum-based variance reduction in non-convex sgd," in *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[40] R. Johnson and T. Zhang, "Accelerating stochastic gradient descent using predictive variance reduction," in *Advances in Neural Information Processing Systems*, 2013.

[41] D. Mishkin and J. Matas, "All you need is a good init," in *International Conference on Learning Representations*, 2016.

[42] L. Huang, D. Yang, B. Lang, and J. Deng, "Decorrelated batch normalization," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018.

[43] A. Jacot, C. Hongler, and F. Gabriel, "Neural tangent kernel: Convergence and generalization in neural networks," in *Advances in Neural Information Processing Systems* , 2018, pp. 8580–8589.

[44] S. Du, J. Lee, H. Li, L. Wang, and X. Zhai, "Gradient descent finds global minima of deep neural networks," in *Proceedings of International Conference on Machine Learning*, K. Chaudhuri and R. Salakhutdinov, Eds., vol. 97, 2019, pp. 1675–1685.

[45] L. Chizat, E. Oyallon, and F. Bach, "On lazy training in differentiable programming," in *Advances in Neural Information Processing Systems*, vol. 32, 2019, pp. 2937–2947.

[46] G. Yang, J. Pennington, V. Rao, J. Sohl-Dickstein, and S. S. Schoenholz, "A mean field theory of batch normalization," in *International Conference on Learning Representations*, 2019.

[47] A. Z. Karen Simonyan, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint: 1409.1556*, 2016.

[48] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 770–778.

[49] M. Tan and Q. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," in *International Conference on Machine Learning*. PMLR, 2019, pp. 6105–6114.

Appendix A Proof of Theorem 1

Proof. Since Assumption 1 holds, at the k-th iteration we have

$$F(w_{k+1}) \leq F(w_k)+\langle \nabla F(w_k), w_{k+1}-w_k\rangle+\frac{L}{2}\|w_{k+1}-w_k\|^2. \tag{15}$$

It is known $w_{k+1}=w_k-\alpha_k v_k$ in Eq. (2), which further implies that

$$\begin{aligned} F(w_{k+1}) \leq& F(w_k)-\alpha_k\langle \nabla F(w_k), v_k\rangle+\frac{L\alpha_k^2}{2}\|v_k\|^2 \\ =& F(w_k)-\alpha_{k+1}\langle \nabla F(w_k), v_k-\nabla F(w_k)\rangle \\ & -\alpha_k\|\nabla F(w_k)\|^2+\frac{L\alpha_k^2}{2}\|v_k\|^2 \\ \leq& F(w_k)-\frac{\alpha_k}{2}\|\nabla F(w_k)\|^2+\frac{\alpha_k}{2}\|v_k-\nabla F(w_k)\|^2 \\ & +\frac{L\alpha_k^2}{2}\|v_k\|^2, \end{aligned} \tag{16}$$

where the last equality follows the fact that $-\langle x,y\rangle \leq \frac{\|x\|^2+\|y\|^2}{2}$. Taking expectation on both sides of the above inequality, we have

$$\begin{aligned} \mathbb{E}[F(w_k)] \leq& F(w_k)-\frac{\alpha_k}{2}\|\nabla F(w_k)\|^2 \\ & +\frac{\alpha_k}{2}\mathbb{E}\left[\|v_k-\nabla F(w_k)\|^2\right]+\frac{L\alpha_k^2}{2}\mathbb{E}\left[\|v_k\|^2\right]. \end{aligned} \tag{17}$$

Rearranging the terms yields

$$\begin{aligned} \frac{\alpha_k}{2}\|\nabla F(w_k)\|^2 \leq& F(w_k)-\mathbb{E}[F(w_k)]+\frac{L\alpha_k^2}{2}\mathbb{E}\left[\|v_k\|^2\right] \\ & +\frac{\alpha_k}{2}\mathbb{E}\left[\|v_k-\nabla F(w_k)\|^2\right]. \end{aligned} \tag{18}$$

Taking the total expectation and summing over 1 to K and combining $F^* \leq F(w_K)$, we obtain

$$\begin{aligned} \frac{1}{2}\sum_{k=1}^{K}\alpha_k\mathbb{E}\left[\|\nabla F(w_k)\|^2\right] \leq& \mathbb{E}[F(w_0)]-F^* \\ & +\sum_{k=1}^{K}\frac{L}{2}\alpha_k^2\mathbb{E}[\|\nabla v_k\|^2] \\ & +\frac{1}{2}\sum_{k=1}^{K}\alpha_k\mathbb{E}\left[\|v_k-\nabla F(w_k)\|^2\right]. \end{aligned} \tag{19}$$

It is known the the stepsizes (the learning rates) $\{\alpha_{k+1}\}$ satisfy $\alpha_k \leq \frac{1}{L}$ and $\alpha_k \leq \alpha_k$, and then we we have

$$\begin{aligned} \frac{\alpha_K}{2}\sum_{k=1}^{K}\mathbb{E}\left[\|\nabla F(w_k)\|^2\right] \leq& (\mathbb{E}[F(w_0)]-F^*) \\ & +\sum_{k=1}^{K}\frac{\alpha_k}{2}\mathbb{E}[\|\nabla v_k\|^2] \\ & +\sum_{k=1}^{K}\frac{\alpha_k}{2}\mathbb{E}\left[\|v_k-\nabla F(w_k)\|^2\right]. \end{aligned} \tag{20}$$

Observing the fact $\mathbb{E}[A+B]=\mathbb{E}[A]+\mathbb{E}[B]$ and the Cauchy-Schwarz Inequality $(\frac{1}{K}\sum_{i=1}^{K}\|\nabla F(w_k)\|)^2 \leq \frac{1}{K}\sum_{i=1}^{K}\|\nabla F(w_k)\|^2$ and dividing the both sides of Eq. (20) with $\frac{\alpha_K}{2}$, we further obtain

$$\begin{aligned} \mathbb{E}\left[\frac{1}{K}\sum_{k=1}^{K}\|\nabla F(w_k)\|\right]^2 \leq& \frac{2(\mathbb{E}[F(w_0)]-F^*)}{\alpha_K K} \\ & +\frac{1}{K\alpha_K}\sum_{k=1}^{K}\alpha_k\mathbb{E}[\|\nabla v_k\|^2] \\ & +\frac{1}{K\alpha_K}\sum_{k=1}^{K}\alpha_k\mathbb{E}\left[\|v_k-\nabla F(w_k)\|^2\right], \end{aligned} \tag{21}$$

and finally we arrive at the desired result. □

Appendix B Lemma 1

Lemma 1. *Given $L:\mathcal{R}^{m\times p}\to\mathcal{R}$, $A\in\mathcal{R}^{m\times n}$, $B\in\mathcal{R}^{p\times q}$, $C\in\mathcal{R}^{m\times q}$ and $Z\in\mathcal{R}^{n\times p}$, we define $f(U)=f(AZB+C)$; then, the Hessian matrix with respect to Z is:*

$$\nabla_Z^2 f(Z)=(B\otimes A^\top)\nabla_U^2 f(U)(B^\top\otimes A), \tag{22}$$

where ∇^2 is the second-order derivative operator, $\otimes$ is the Kronecker product operator.

Proof. Since $f(U)=f(AZB+C)$, we know

$$\begin{aligned} \partial L =& \mathrm{Tr}\left((\nabla_U f(U))^\top \partial U\right) \\ =& \mathrm{Tr}\left((\nabla_U f(U))^\top \partial(AZB)\right) \\ =& \mathrm{Tr}\left((\nabla_U f(U))^\top A\partial ZB\right) \\ =& \mathrm{Tr}\left(B(\nabla_U f(U))^\top A\partial Z\right), \end{aligned} \tag{23}$$

where the last equality results from $\mathrm{Tr}(XY)=\mathrm{Tr}(YX)$

Hence, we obtain

$$\nabla_Z L(Z)=A^\top\nabla_U f(U)B^\top. \tag{24}$$

Further, we have

$$\begin{aligned} \mathrm{vec}(\partial(\nabla_Z f(Z))) =& \mathrm{vec}\left(\partial\left(A^\top\nabla_U f(U)B^\top\right)\right) \\ =& (B\otimes A^\top)\mathrm{vec}\left(\partial(\nabla_U f(U))\right) \\ =& (B\otimes A^\top)\frac{\mathrm{vec}(\partial(\nabla_U f(U)))}{\mathrm{vec}(\partial U)}\mathrm{vec}(\partial U) \\ =& (B\otimes A^\top)\nabla_U^2 f(U)\mathrm{vec}(A\partial ZB) \\ =& (B\otimes A^\top)\nabla_U^2 f(U)(B^\top\otimes A)\mathrm{vec}(\partial Z) \end{aligned} \tag{25}$$

where the second equality and the fifth equality is due to the fact that $\mathrm{vec}(XYW)=(W^\top\otimes X)\mathrm{vec}(Y)$.

It implies that

$$\nabla_Z^2 f(Z)=\frac{\mathrm{vec}((\partial\nabla_Z f(Z))}{\mathrm{vec}(\partial Z)}=(B\otimes A^\top)\nabla_U^2 f(U)(B^\top\otimes A), \tag{26}$$

and finally we achieve the desired result. □

Appendix C
Lemma 2

Lemma 2. *Given that $a \in \mathcal{R}^d$ is a vector and it satisfied $\|a\| = 1$, we obtain $\|I - aa^\top\|_2 = 1$.*

Proof. $aa^\top$ and $a^\top a$ have the same non-zero eigenvalues, and we know $a^\top a$ is actually a scalar and the value is $a^\top a = \|a\|^2 = 1$, so that the eigenvalue of $a^\top a$ is 1. Therefore, the eigenvalues of $aa^\top$ is either 1 or 0, and then the eigenvalues of $I - aa^\top$ is either 1 or 0, so we obtain the conclusion $\|I - aa^\top\|_2 = 1$. □

Appendix D
Lemma 3

Lemma 3. *A standard network with no normalization and a standard network with CBN are respectively shown in Figure 1(a) and Figure 1(b). The inputs of the first block and the outputs of the last block for the both networks satisfy $\tilde{x}^{[1]}_{i_k} = \hat{x}^{[1]}_{i_k}$, $\|\nabla_{\hat{x}^{[L]}_{i_k}} \hat{f}_{i_k}\|_2 = \|\nabla_{\tilde{x}^{[L]}_{i_k}} \tilde{f}_{i_k}\|_2$ and $\|\nabla^2_{\hat{x}^{[L]}_{i_k}} \hat{f}_{i_k}\|_2 = \|\nabla^2_{\tilde{x}^{[L]}_{i_k}} \tilde{f}_{i_k}\|_2$. Suppose that at l-th block with mean removal $0 \le \hat{\eta}^{[l]}_k \le 2$ where $\hat{\eta}^{[l]}_k = \frac{\hat{\mu}^{[l]}_{\mathcal{B}_k} - \beta^{[l]}_k}{\mathbb{E}[\hat{x}^{[l]}_{i_k}]}$, we then have the following at any block:*

(1). The upper bound of $\mathbb{E}[\|\hat{x}^{[l]}_{i_k}\|_2^2]$ is less than the upper bound of $\mathbb{E}[\|\tilde{x}^{[l]}_{i_k}\|_2^2]$.

(2). The upper bound of $\mathbb{E}[\|\nabla_{\hat{x}^{[l]}_{i_k}} \hat{f}_{i_k}\|_2]$ is less than the upper bound of $\mathbb{E}[\|\nabla_{\tilde{x}^{[l]}_{i_k}} \tilde{f}_{i_k}\|_2]$.

(3). The upper bound of $\|\nabla^2_{\hat{x}^{[l]}_k} \hat{F}_k\|_2$ is less than the upper bound of $\|\nabla^2_{\tilde{x}^{[l]}_k} \tilde{F}_k\|_2$.

Proof. We use mathematical induction to prove the propositions.

(1) We first to comparative analysis for the forward paths for the network with mean removal and the network without normalization.

For the inductive step, we assume the following inequality holds, *i.e.*,

$$\sup(\mathbb{E}[\|\hat{x}^{[l]}_{i_k}\|_2^2)]) \le \sup(\mathbb{E}[\|\tilde{x}^{[l]}_{i_k}\|_2^2]), \tag{27}$$

We analyze the network with mean removal. It is known $\hat{x'}^{[l]}_{i_k} = \hat{x}^{[l]}_{i_k} - \mu^{[l]}_{\mathcal{B}_k} + \beta^{[l]}_k$ where $\mu^{[l]}_{\mathcal{B}_k} - \beta^{[l]}_k = \hat{\eta}^{[l]}_k \mathbb{E}[\hat{x}^{[l]}_{i_k}]$, and then

$$\begin{aligned}\mathbb{E}[\|\hat{x'}^{[l]}_{i_k}\|_2^2] =& \mathbb{E}[\|\hat{x}^{[l]}_{i_k} - \mu^{[l]}_{\mathcal{B}_k} + \beta^{[l]}_k\|_2^2] \\ =& \mathbb{E}[\|\hat{x}^{[l]}_{i_k} - \hat{\eta}^{[l]}_k \mathbb{E}[\hat{x}^{[l]}_{i_k}]\|_2^2] \\ =& \mathbb{E}[\|\hat{x}^{[l]}_{i_k}\|_2^2] - \hat{\eta}^{[l]}_k(2 - \hat{\eta}^{[l]}_k)\mathbb{E}[\|\hat{x}^{[l]}_{i_k}\|]^2,\end{aligned} \tag{28}$$

It is known that $0 \le \hat{\eta}^{[l]}_k \le 2$, so $0 \le \hat{\eta}^{[l]}_k(2 - \hat{\eta}^{[l]}_k) \le 1$. Then, we have

$$\mathbb{E}[\|\hat{x'}^{[l]}_{i_k}\|_2^2] \le \mathbb{E}[\|\hat{x}^{[l]}_{i_k}\|_2^2]. \tag{29}$$

$\hat{y}^{[l]}_{i_k} = w^{[l]}_k \hat{x'}^{[l]}_{i_k}$ implies that

$$\|\hat{y}^{[l]}_{i_k}\|_2^2 \le \|w^{[l]}_k\|_2^2 \|\hat{x'}^{[l]}_{i_k}\|_2^2 \tag{30}$$

Since the nonlinear activation $\delta(\cdot)$ is a shrinkage function, we obtain

$$\hat{x}^{[l+1]}_{i_k} = \mathbb{I}(\mathrm{SC})\hat{x}^{[l]}_{i_k} + \hat{\zeta}^{[l]}_{i_k}\hat{y}^{[l]}_{i_k} \tag{31}$$

where $\mathbb{I}(\mathrm{SC})$ indicates whether there is a shortcut, and $0 \le \hat{\zeta}^{[l]}_{i_k} \le 1$.

Combining Eq.(28-31), we obtain

$$\begin{aligned}\mathbb{E}[\|\hat{x}^{[l+1]}_{i_k}\|_2^2] &\le \left(\mathbb{I}(\mathrm{SC}) + (\hat{\zeta}^{[l]}_{i_k})^2\|w^{[l]}_k\|_2^2\right)\mathbb{E}[\|\hat{x}^{[l]}_{i_k}\|_2^2] \\ &\le \left(\mathbb{I}(\mathrm{SC}) + \|w^{[l]}_k\|_2^2\right)\mathbb{E}[\|\hat{x}^{[l]}_{i_k}\|_2^2]\end{aligned} \tag{32}$$

Similar to Eq.(30-32), for the l-th layer in the network without normalization, we also have

$$\begin{aligned}\mathbb{E}[\|\tilde{x}^{[l+1]}_{i_k}\|_2^2] &\le \left(\mathbb{I}(\mathrm{SC}) + (\tilde{\zeta}^{[l]}_{i_k})^2\|w^{[l]}_k\|_2^2\right)\mathbb{E}[\|\tilde{x}^{[l]}_{i_k}\|_2^2] \\ &\le \left(\mathbb{I}(\mathrm{SC}) + \|w^{[l]}_k\|_2^2\right)\mathbb{E}[\|\tilde{x}^{[l]}_{i_k}\|_2^2]\end{aligned} \tag{33}$$

Combining Eq.(27), Eq.(32) and Eq.(33), we obtain

$$\sup(\mathbb{E}[\|\hat{x}^{[l+1]}_{i_k}\|_2^2]) \le \sup(\mathbb{E}[\|\tilde{x}^{[l+1]}_{i_k}\|_2^2]), \tag{34}$$

where this completes the inductive step. We know that the inputs of the network with mean removal and the non-normalized network are the the same, *i.e.*, $\hat{x}^{[1]}_{i_k} = \tilde{x}^{[1]}_{i_k}$. Since both the base case and the induction step have been proved as true, by mathematical induction the statement $\sup(\mathbb{E}[\|\hat{x}^{[s]}_{i_k}\|_2^2]) \le \sup(\mathbb{E}[\|\tilde{x}^{[s]}_{i_k}\|_2^2])$ holds for any layer s $(1 \le s \le L)$ [1]

(2) We first comparatively analyze the backward paths of the network with mean removal and the network without normalization.

For the inductive step, we assume that the upper bound of $\mathbb{E}[\|\nabla_{\hat{x}^{[l+1]}_k} \hat{f}_k\|]^2$ is less than the upper bound of $\mathbb{E}[\|\nabla_{\tilde{x}^{[l+1]}_k} \tilde{f}_{i_k}\|^2]$ at the l-th layer holds, *i.e.*,

$$\sup(\mathbb{E}[\|\nabla_{\hat{x}^{[l+1]}_k} \hat{f}_{i_k}\|^2]) \le \sup(\mathbb{E}[\|\nabla_{\tilde{x}^{[l+1]}_k} \tilde{f}_{i_k}\|^2]) \tag{35}$$

We analyze the network with mean removal. We know $\hat{x}^{[l+1]}_k = \delta(\hat{y}^{[l]}_{i_k})$. Then, for the derivative with respect to $\hat{y}^{[l]}_{i_k}$, we have

$$\|\nabla_{\hat{y}^{[l]}_k} \hat{f}_{i_k}\|^2 = \|\delta'(\hat{y}^{[l]}_{i_k}) \odot \nabla_{\hat{x}^{[l+1]}_k} \hat{f}_{i_k}\|^2 \le \|\nabla_{\hat{x}^{[l+1]}_k} \hat{f}_{i_k}\|^2, \tag{36}$$

where $\|\delta'(\hat{y}^{[l]}_{i_k})\| \le 1$ since the derivative of the nonlinear activation $\delta(u)$ with respect to u is less than 1.

It is known $\hat{y}^{[l]}_k = w^{[l]}_k \hat{x'}^{[l]}_{i_k}$, and then according to Lemma 1, we have

$$\|\nabla_{\hat{x'}^{[l]}_{i_k}} \hat{f}_{i_k}\|^2 = \|(w^{[l]}_k)^\top \nabla_{\hat{y'}^{[l]}_{i_k}} \hat{f}_{i_k}\|^2 \le \|w^{[l]}_k\|^2 \|\nabla_{\hat{y'}^{[l]}_{i_k}} \hat{f}_{i_k}\|^2. \tag{37}$$

1. *The equality of Eq.(30) can be basically satisfied in practice. Because we commonly employ Xavier Initialization for the network, so that at the beginning, the parameter $w^{[l]}_0$ at the l-th layer make sure $\mathbb{E}(\|y^{[l]}_{i_0}\|_2^2) = \mathbb{E}(\|x^{[l]}_{i_0}\|_2^2)$ satisfied. According to the latest neural tangent kernel(NTK) theory, overparameterized neural networks make their parameters hardly vary from their initial value in optimizing process. Furthermore, the nonlinear activation function $\delta(\cdot)$ can make the following statement satisfied that when $\|\hat{y}^{[l]}_{i_k}\|_2 \le \|\tilde{y}^{[l]}_{i_k}\|$, $\|\hat{x}^{[l+1]}_{i_k}\| \le \|\tilde{x}^{[l+1]}_{i_k}\|_2$. Therefore, loosely speaking, $\mathbb{E}[\|\hat{x}^{[l+1]}_{i_k}\|_2^2] \le \mathbb{E}[\|\tilde{x}^{[l+1]}_{i_k}\|_2^2]$ can also hold without the upper bound symbol.*

It is known $\hat{x'}_{i_k}^{[l]} = \hat{x}_{i_k}^{[l]} - \frac{1}{|\mathcal{B}|}\sum_{i_k\in\mathcal{B}}\hat{x}_{i_k}^{[l]} + \beta_k^{[l]}$, and we define

$$\hat{x}_{\mathcal{B}_k}^{[l]} = \left[\hat{x}_{1_k}^{[l]}, \hat{x}_{2_k}^{[l]}, ..., \hat{x}_{|\mathcal{B}|_k}^{[l]}\right] \tag{38}$$

and

$$\hat{x'}_{\mathcal{B}_k}^{[l]} = \left[\hat{x'}_{1_k}^{[l]}, \hat{x'}_{2_k}^{[l]}, ..., \hat{x'}_{|\mathcal{B}|_k}^{[l]}\right]. \tag{39}$$

Then, we have

$$\left(\hat{x'}_{\mathcal{B}_k}^{[l]}\right)_j = \left(\hat{x}_{\mathcal{B}_k}^{[l]}\right)_j\left(I - \frac{1}{|\mathcal{B}|}\mathbf{1}\mathbf{1}^\top\right) + (\beta_k^{[l]})_j, \tag{40}$$

where $j \in [1, 2, ..., d]$ and d is the number of dimension of $\hat{x}_{1_k}^{[l]}$ and $\hat{x'}_{1_k}^{[l]}$, and I is the identity matrix and $\mathbf{1} = [\underbrace{1, 1, ..., 1}_{|\mathcal{B}|}]$.

Next, we obtain

$$\begin{aligned}\left\|\left(\nabla_{\hat{x}_{\mathcal{B}_k}^{[l]}}\hat{f}_{\mathcal{B}_k}\right)_j\right\|^2 &= \left\|\left(\nabla_{\hat{x'}_{\mathcal{B}_k}^{[l]}}\hat{f}_{\mathcal{B}_k}\right)_j\left(I - \frac{1}{|\mathcal{B}|}\mathbf{1}\mathbf{1}^\top\right)^\top\right\|^2\\ &\leq \left\|\left(\nabla_{\hat{x'}_{\mathcal{B}_k}^{[l]}}\hat{f}_{\mathcal{B}_k}\right)_j\right\|^2\left\|I - \frac{1}{|\mathcal{B}|}\mathbf{1}\mathbf{1}^\top\right\|^2\\ &= \left\|\left(\nabla_{\hat{x'}_{\mathcal{B}_k}^{[l]}}\hat{f}_{\mathcal{B}_k}\right)_j\right\|^2,\end{aligned} \tag{41}$$

where the first equality is due to Lemma 1, and the last inequality is owing to Lemma 2.

We further have

$$\begin{aligned}\mathbb{E}\left[\left\|\nabla_{\hat{x}_{i_k}^{[l]}}\hat{f}_{i_k}\right\|^2\right] &= \frac{1}{|\mathcal{B}|}\mathbb{E}\left[\left\|\nabla_{\hat{x}_{\mathcal{B}_k}^{[l]}}\hat{f}_{i_k}\right\|^2\right]\\ &= \frac{1}{|\mathcal{B}|}\mathbb{E}\left[\sum_{j=1}^{d}\left\|\left(\nabla_{\hat{x}_{\mathcal{B}_k}^{[l]}}\hat{f}_{\mathcal{B}_k}\right)_j\right\|^2\right]\\ &\leq \frac{1}{|\mathcal{B}|}\mathbb{E}\left[\sum_{j=1}^{d}\left\|\left(\nabla_{\hat{x'}_{\mathcal{B}_k}^{[l]}}\hat{f}_{\mathcal{B}_k}\right)_j\right\|^2\right]\\ &= \frac{1}{|\mathcal{B}|}\mathbb{E}\left[\left\|\nabla_{\hat{x'}_{\mathcal{B}_k}^{[l]}}\hat{f}_{\mathcal{B}_k}\right\|^2\right]\\ &= \mathbb{E}\left[\left\|\nabla_{\hat{x'}_{i_k}^{[l]}}\hat{f}_{i_k}\right\|^2\right].\end{aligned} \tag{42}$$

Combining Eq.(36-42) and the fact $\hat{x}_{i_k}^{[l+1]} = \mathbb{I}(\text{SC})\hat{x}_{i_k}^{[l]} + \delta(\hat{y}_{i_k}^{[l]})$ where $\mathbb{I}(\text{SC})$ indicates whether there is a shortcut, we have

$$\begin{aligned}\mathbb{E}\left[\|\nabla_{\hat{x}_{i_k}^{[l]}}\hat{f}_{i_k}\|^2\right] =& \mathbb{E}[\|\mathbb{I}(\text{SC}) + (w_k^{[l]})^\top(\delta'(\hat{y}_{i_k}^{[l]}) \odot \nabla_{\hat{x}_{i_k}^{[l+1]}}\hat{f}_{i_k})\\ &(I - \frac{1}{|\mathcal{B}|}\mathbf{1}\mathbf{1}^\top)^\top\|^2]\\ \leq& \left(\mathbb{I}(\text{SC}) + \|w_k^{[l]}\|^2\right)\mathbb{E}[\|\nabla_{\hat{x}_{i_k}^{[l+1]}}\hat{f}_{i_k}\|^2]\end{aligned} \tag{43}$$

Similar to Eq.(36-37)and Eq.(43), we can obtain the following inequality for the l-th layer in the network without normalization,

$$\begin{aligned}\mathbb{E}[\|\nabla_{\tilde{x}_{i_k}^{[l]}}\tilde{f}_{i_k}\|^2] =& \mathbb{E}[\|\mathbb{I}(\text{SC}) + (w_k^{[l]})^\top(\delta'(\tilde{y}_{i_k}^{[l]}) \odot \nabla_{\tilde{x}_{i_k}^{[l+1]}}\tilde{f}_{i_k})\|^2]\\ \leq& \left(\mathbb{I}(\text{SC}) + \|w_k^{[l]}\|^2\right)\mathbb{E}[\|\nabla_{\tilde{x}_{i_k}^{[l+1]}}\tilde{f}_{i_k}\|^2].\end{aligned} \tag{44}$$

Combining Eq.(35), Eq.(43) and Eq.(44), we have

$$\sup(\mathbb{E}[\|\nabla_{\hat{x}_{i_k}^{[l]}}\hat{f}_{i_k}\|_2]) \leq \sup(\mathbb{E}[\|\nabla_{\tilde{x}_{i_k}^{[l]}}\tilde{f}_{i_k}\|]), \tag{45}$$

It completes the inductive step. We know that l_2-norm of the derivative of $\hat{f}_{i_k}$ with respect to $\hat{x}_{i_k}$ for the network with mean removal and $\tilde{f}$ with respect to $\tilde{x}_{i_k}$ for the network without normalization are equal, *i.e.*, $\|\nabla_{\hat{x}_{i_k}^{[L]}}\hat{f}_{i_k}\|^2 = \|\nabla_{\tilde{x}_{i_k}^{[L]}}\tilde{f}_{i_k}\|^2$. Since both the base case and the induction step have been proved as true, by mathematical induction the statement $\sup(\mathbb{E}[\|\nabla_{\hat{x}_{i_k}^{[s]}}\hat{f}_{i_k}\|^2]) \leq \sup(\mathbb{E}[\|\nabla_{\tilde{x}_{i_k}^{[s]}}\tilde{f}_{i_k}\|^2])$ holds for any layer s $(1 \leq s \leq L)$ [2].

(3) We comparatively analyze the backward paths of the network with mean removal and the network without normalization.

For the inductive step, we assume that the upper bound of $\|\nabla^2_{\hat{x}_k^{[l+1]}}\hat{F}_k\|_2$ for the network with mean removal is less than the upper bound of $\|\nabla^2_{\hat{x}_k^{[l+1]}}\tilde{F}_k\|$ for the network without normalization at the l-th layer holds, *i.e.*,

$$\sup(\|\nabla^2_{\hat{x}_k^{[l+1]}}\hat{F}_k\|_2) \leq \sup(\|\nabla^2_{\hat{x}_k^{[l+1]}}\tilde{F}_k\|), \tag{46}$$

where $\hat{F} = \frac{1}{|\mathcal{U}|}\sum_{i\in\mathcal{U}}\hat{f}_i$ where $\mathcal{U}$ is the universal sample set and $x_k^{[l]} = [x_{1_k}^{[l]}, x_{2_k}^{[l]}, \cdots, x_{|\mathcal{U}_k|_k}^{[l]}]$.

We analyze the network with mean removal. For the second-order derivative with respect to any sample $\hat{y}_{i_k}^{[l]}$, we have

$$\|\nabla^2_{\hat{y}_k^{[l]}}\hat{f}_{i_k}\|_2 = \|\delta''(\hat{y}_{i_k}^{[l]}) \odot \nabla^2_{\hat{x}_k^{[l+1]}}\hat{f}_{i_k}\|_2 \leq \|\nabla^2_{\hat{x}_k^{[l+1]}}\hat{f}_{i_k}|_2, \tag{47}$$

where$\|\delta''(\hat{y}_{i_k}^{[l]})\| \leq 1$ since the second derivative of the nonlinear activation $\delta(u)$ with respect to u is less than 1.

Next, according to Lemma 1, we have

$$\begin{aligned}\|\nabla^2_{\hat{x'}_{i_k}^{[l]}}\hat{f}_{i_k}\|_2 =& \|(I \otimes (w_k^{[l]})^\top)\nabla^2_{\hat{y'}_{i_k}^{[l]}}\hat{f}_{i_k}(I \otimes w_k^{[l]})\|_2\\ \leq& \|\nabla^2_{\hat{y'}_{i_k}^{[l]}}\hat{f}_{i_k}\|\|(I \otimes (w_k^{[l]})^\top)(I \otimes w_k^{[l]})\|_2\\ =& \|I \otimes ((w_k^{[l]})^\top w_k^{[l]})\|_2\|\nabla^2_{\hat{y'}_{i_k}^{[l]}}\hat{f}_{i_k}\|_2\\ =& \|(w_k^{[l]})^\top w_k^{[l]}\|_2\|\nabla^2_{\hat{y'}_{i_k}^{[l]}}\hat{f}_{i_k}\|\\ =& \|w_k^{[l]}\|_2^2\|\nabla^2_{\hat{y'}_{i_k}^{[l]}}\hat{f}_{i_k}\|,\end{aligned} \tag{48}$$

where the third equality (ii) results from $(A \otimes B)(C \otimes D) = AC \otimes BD$, and the fourth equality is following the fact that $\|A \otimes B\|_2 = \|A\|_2\|B\|_2$.

2. *The equality of Eq.(37) can be basically satisfied in practice. Because we commonly employ Xavier Initialization for the network, so that at the beginning, the parameter $w_0^{[l]}$ at the l-th layer make sure $\mathbb{E}(\|\nabla_{x_{i_0}^{[l]}}f_{i_k}\|_2^2) = \mathbb{E}(\|\nabla_{y_{i_0}^{[l]}}f_{i_k}\|_2^2)$ satisfied for both network with mean removal and the network without normalization. According to the latest neural tangent kernel(NTK) theory, overparameterized neural networks make their parameters hardly vary from their initial value in optimizing process. Furthermore, it is reasonable for $\|\delta'(\hat{y}_{i_k}^{[l]}) \odot \nabla_{\hat{x}_k^{[l+1]}}\hat{f}_{i_k}\| \simeq \|\delta'(\tilde{y}_{i_k}^{[l]}) \odot \nabla_{\tilde{x}_k^{[l+1]}}\tilde{f}_{i_k}\|$. Because if the nonlinear activation is Sigmoid or Tanh, it commonly works in the linear range, so that $\hat{\theta}_k^{[l]}$ and $\tilde{\theta}_k^{[l]}$ are both close to 1, and if nonlinear activation is a ReLU-type nonlinearity, the positive inputs will be passed almost without distortion and the negative inputs will be set to zero, and the inputs $\{\hat{x}_{i_k}^{[L]}\}_{i_k\in\mathcal{U}}$ and $\{\tilde{x}_{i_k}^{[L]}\}_{i_k\in\mathcal{U}}$ are commonly symmetrical around zero. Hence, loosely speaking, $\mathbb{E}[\|\nabla_{\hat{x}_k^{[l]}}\hat{f}_{i_k}\|_2] \leq \mathbb{E}[\|\nabla_{\tilde{x}_k^{[l]}}\tilde{f}_{i_k}\|]$ can hold without the upper bound symbol.*

It is known $\hat{x'}^{[l]}_{i_k} = \hat{x}^{[l]}_{i_k} - \frac{1}{|\mathcal{B}|}\sum_{i_k \in \mathcal{B}} \hat{x}^{[l]}_{i_k} + \beta^{[l]}_k$, and the definitions of $\hat{X}^{[l]}_{\mathcal{B}_k}$ and $\hat{X'}^{[l]}_{\mathcal{B}_k}$ are shown in Eq.(38) and Eq.(39), and then we have $\left(\hat{x'}^{[l]}_{\mathcal{B}_k}\right)_j = \left(\hat{x}^{[l]}_{\mathcal{B}_k}\right)_j \left(I - \frac{1}{|\mathcal{B}|}\mathbf{1}\mathbf{1}^\top\right) + (\beta^{[l]}_k)_j$. We set $V = I - \frac{1}{|\mathcal{B}|}\mathbf{1}\mathbf{1}^\top$, and we further obtain

$$\begin{aligned}\left\|\left(\nabla^2_{\hat{x}^{[l]}_{\mathcal{B}_k}} \hat{f}_{\mathcal{B}_k}\right)_j\right\| &= \left\|(V \otimes I)\left(\nabla^2_{\hat{x'}^{[l]}_{\mathcal{B}_k}} \hat{f}_{\mathcal{B}_k}\right)_j (V^\top \otimes I)\right\| \\ &\leq \|(V \otimes I)\|^2 \left\|\left(\nabla^2_{\hat{x'}^{[l]}_{\mathcal{B}_k}} \hat{f}_{\mathcal{B}_k}\right)_j\right\| \\ &= \|V\|^2 \left\|\left(\nabla^2_{\hat{x'}^{[l]}_{\mathcal{B}_k}} \hat{f}_{\mathcal{B}_k}\right)_j\right\| \\ &= \left\|\left(\nabla^2_{\hat{x'}^{[l]}_{\mathcal{B}_k}} \hat{f}_{\mathcal{B}_k}\right)_j\right\|^2,\end{aligned} \tag{49}$$

where the first equality is due to Lemma 1; the second inequality is due to the fact that $(A \otimes B)^\top = A^\top \otimes B^\top$ for any A and B and $V^\top = V$; the third equality results from the fact that $\|A \otimes B\|_2 = \|A\|_2\|B\|_2$ for any A and B; the last inequality is owing to Lemma 2.

Then, we obtain

$$\begin{aligned}\mathbb{E}\left[\left\|\nabla^2_{\hat{x}^{[l]}_{i_k}} \hat{f}_{i_k}\right\|^2\right] &= \frac{1}{|\mathcal{B}|}\mathbb{E}\left[\left\|\nabla^2_{\hat{x}^{[l]}_{\mathcal{B}_k}} \hat{f}_{i_k}\right\|^2\right] \\ &= \frac{1}{|\mathcal{B}|}\mathbb{E}\left[\sum_{j=1}^{d}\left\|\left(\nabla^2_{\hat{x}^{[l]}_{\mathcal{B}_k}} \hat{f}_{\mathcal{B}_k}\right)_j\right\|^2\right] \\ &\leq \frac{1}{|\mathcal{B}|}\mathbb{E}\left[\sum_{j=1}^{d}\left\|\left(\nabla^2_{\hat{x'}^{[l]}_{\mathcal{B}_k}} \hat{f}_{\mathcal{B}_k}\right)_j\right\|^2\right] \\ &= \frac{1}{|\mathcal{B}|}\mathbb{E}\left[\left\|\nabla_{\hat{x'}^{[l]}_{\mathcal{B}_k}} \hat{f}^2_{\mathcal{B}_k}\right\|^2\right] \\ &= \mathbb{E}\left[\left\|\nabla^2_{\hat{x'}^{[l]}_{i_k}} \hat{f}_{i_k}\right\|^2\right]\end{aligned} \tag{50}$$

Combining Eq.(47-50) and the fact $\hat{x}^{[l+1]}_{i_k} = \mathbb{I}(\mathrm{SC})\hat{x}^{[l+1]}_{i_k} + \delta(\hat{y}^{[l]}_{i_k})$ where $\mathbb{I}(\mathrm{SC})$ indicates whether there is a shortcut, we have

$$\begin{aligned}\|\nabla^2_{\hat{x}^{[l]}_k} \hat{F}_k\|^2 \leq& \mathbb{E}[\|\nabla^2_{\hat{x}^{[l]}_{i_k}} \hat{f}_{i_k}\|^2] \\ \leq& \left(\mathbb{I}(\mathrm{SC}) + \|w^{[l]}_k\|^4_2\right)\mathbb{E}[\|\nabla^2_{\hat{x}^{[l+1]}_{i_k}} \hat{f}_{i_k}\|^2]\end{aligned} \tag{51}$$

Similar to Eq.(47-48)and Eq.(51), we can obtain the following inequality for the l-th layer in the network without normalization,

$$\begin{aligned}&\|\nabla^2_{\tilde{x}^{[l]}_k} \tilde{F}_k\|^2 \leq \mathbb{E}[\|\nabla^2_{\tilde{x}^{[l]}_{i_k}} \hat{f}_{i_k}\|^2] \\ =&\mathbb{E}\left[\mathbb{I}(\mathrm{SC})\nabla^2_{\tilde{y'}^{[l]}_{i_k}} \tilde{f}_{i_k} + \|(I \otimes (w^{[l]}_k)^\top)(\delta''(\hat{y}^{[l]}_{i_k}) \odot \nabla^2_{\tilde{y'}^{[l]}_{i_k}} \tilde{f}_{i_k})(I \otimes w^{[l]}_k)\|^2\right] \\ \leq&(\mathbb{I}(\mathrm{SC}) + \|w^{[l]}_k\|^4_2)\mathbb{E}[\|\nabla^2_{\tilde{x}^{[l+1]}_{i_k}} \hat{f}_{i_k}\|^2]\end{aligned} \tag{52}$$

Combining Eq.(46), Eq.(51) and Eq.(52), we have

$$\sup(\|\nabla^2_{\hat{x}^{[l]}_k} \hat{F}_k\|) \leq \sup(\|\nabla^2_{\hat{x}^{[l]}_k} \tilde{F}_k\|), \tag{53}$$

where this completes the inductive step. We know that l_2-norm of the second-order derivative of $\hat{F}_k$ with respect to $\hat{x}_{i_k}$ for the network with mean removal and $\tilde{F}$ with respect to $\tilde{x}_k$ for the network without normalization are equal, *i.e.*, $\|\nabla^2_{\hat{x}^{[L]}_k} \hat{F}_k\|_2 = \|\nabla^2_{\tilde{x}^{[L]}_k} \tilde{F}_k\|_2$. Since both the base case and the induction step have been proved as true, by mathematical induction the statement $\sup(\|\nabla^2_{\hat{x}^{[s]}_k} \hat{F}_k\|) \leq \sup(\|\nabla^2_{\hat{x}^{[s]}_k} \tilde{F}_k\|)$ holds for any layer s $(1 \leq s \leq L)$. □

Appendix E
Proof of Theorem 2

Proof. (1). If Assumption 1 holds, the following inequality can be obtained with Lagrange Mean Theorem:

$$L = \|\nabla^2_w F\|_2, \tag{54}$$

where L is the gradient Lipschitz constant and $\nabla^2_w F$ is the second-order derivative of F.

It is known $\hat{F} = \frac{1}{|\mathcal{U}|}\sum_{i \in \mathcal{U}} \hat{f}_i$ where $\mathcal{U}$ is the universal sample set. At the l-th block, $\hat{y}^{[l]}_{i_k} = w^{[l]}_k \hat{x}^{[l]}_{i_k}$, and then second-order derivative of F with respect to $w^{[l]}_k$ is

$$\begin{aligned}\nabla^2_{w^{[l]}_k} \hat{F}_k =& \frac{1}{|\mathcal{U}|}\sum_{i_k \in \mathcal{U}} \nabla^2_{w^{[l]}_k} \hat{f}_{i_k} \\ =& \frac{1}{|\mathcal{U}|}\sum_{i_k \in \mathcal{U}} \left((\hat{x}^{[l]}_{i_k} \otimes I)\nabla^2_{\hat{y}^{[l]}_{i_k}} \hat{f}_{i_k}((\hat{x}^{[l]}_{i_k})^\top \otimes I)\right),\end{aligned} \tag{55}$$

where the second equality holds by following Lemma 1.

Then, we obtain

$$\begin{aligned}&\|\nabla^2_{w^{[l]}_k} \hat{F}_k\|_2 = \left\|\frac{1}{|\mathcal{U}|}\sum_{i_k \in \mathcal{U}} \left((\hat{x}^{[l]}_{i_k} \otimes I)\nabla^2_{\hat{y}^{[l]}_{i_k}} \hat{f}_{i_k}((\hat{x}^{[l]}_{i_k})^\top \otimes I)\right)\right\|_2 \\ &\leq \frac{1}{|\mathcal{U}|}\sum_{i_k \in \mathcal{U}} \left\|(\hat{x}^{[l]}_{i_k} \otimes I)\nabla^2_{\hat{y}^{[l]}_{i_k}} \hat{f}_{i_k}((\hat{x}^{[l]}_{i_k})^\top \otimes I)\right\|_2 \\ &\overset{(i)}{=} \frac{1}{|\mathcal{U}|}\sum_{i_k \in \mathcal{U}} \left\|((\hat{x}^{[l]}_{i_k})^\top \otimes I)(\hat{x}^{[l]}_{i_k} \otimes I)\nabla^2_{\hat{y}^{[l]}_{i_k}} \hat{f}_{i_k}\right\|_2 \\ &\leq \frac{1}{|\mathcal{U}|}\sum_{i_k \in \mathcal{U}} \left(\left\|((\hat{x}^{[l]}_{i_k})^\top \otimes I)(\hat{x}^{[l]}_{i_k} \otimes I)\right\|_2 \left\|\nabla^2_{\hat{y}^{[l]}_{i_k}} \hat{f}_{i_k}\right\|_2\right) \\ &\overset{(ii)}{=} \frac{1}{|\mathcal{U}|}\sum_{i_k \in \mathcal{U}} \left(\left\|(\hat{x}^{[l]}_{i_k})^\top \hat{x}^{[l]}_{i_k}) \otimes I\right\|_2 \left\|\nabla^2_{\hat{y}^{[l]}_{i_k}} \hat{f}_{i_k}\right\|_2\right) \\ &\overset{(iii)}{=} \frac{1}{|\mathcal{U}|}\sum_{i_k \in \mathcal{U}} \left(\left\|(\hat{x}^{[l]}_{i_k})^\top \hat{x}^{[l]}_{i_k}\right\|_2 \left\|\nabla^2_{\hat{y}^{[l]}_{i_k}} \hat{f}_{i_k}\right\|_2\right) \\ &= \frac{1}{|\mathcal{U}|}\sum_{i_k \in \mathcal{U}} \left(\left\|\hat{x}^{[l]}_{i_k}\right\|^2_2 \left\|\nabla^2_{\hat{y}^{[l]}_{i_k}} \hat{f}_{i_k}\right\|_2\right) \\ &\overset{(iv)}{\leq} \left(\frac{1}{|\mathcal{U}|}\sum_{i_k \in \mathcal{U}} \left\|\hat{x}^{[l]}_{i_k}\right\|^2_2\right) \left\|\nabla^2_{\hat{y}^{[l]}_k} \hat{F}_k\right\|_2 \\ &\overset{(v)}{=} \mathbb{E}\left[\left\|\hat{x}^{[l]}_{i_k}\right\|^2_2\right] \left\|\nabla^2_{\hat{y}^{[l]}_k} \hat{F}_k\right\|_2\end{aligned} \tag{56}$$

where the equality (i) holds due to that $\|ABC\|_2 = \|CAB\|_2$;the equality (ii) results from $(A \otimes B)(C \otimes D) = AC \otimes BD$; the equality (iii) is following the fact that $\|A \otimes B\|_2 = \|A\|_2\|B\|_2$; the inequality (iv) is due to

Cauchy-Schwarz inequality; the equality (v) is owing to $\mathbb{E}[\|\hat{x}_{i_k}^{[l]}\|_2^2] = \frac{1}{|\mathcal{U}|}\sum_{i_k\in\mathcal{U}}\left\|\hat{x}_{i_k}^{[l]}\right\|_2^2$.

Now we turn attention to the network with no normalization. Similar to Eq.(55-56), we can obtain

$$\|\nabla^2_{w_k^{[l]}}\tilde{F}_k\|_2 \le \mathbb{E}\left[\left\|\tilde{x}_{i_k}^{[l]}\right\|_2^2\right]\left\|\nabla^2_{\tilde{y}_k^{[l]}}\tilde{F}_k\right\|_2 \tag{57}$$

According to Lemma 3, we know

$$\sup(\mathbb{E}[\|\hat{x}_{i_k}^{[l]}\|_2^2]) \le \sup(\mathbb{E}[\|\tilde{x}_{i_k}^{[l]}\|_2^2]), \tag{58}$$

and

$$\sup(\|\nabla^2_{\hat{y}_k^{[l]}}\hat{F}_k\|_2) \le \sup(\|\nabla^2_{\tilde{y}_k^{[l]}}\tilde{F}_k\|). \tag{59}$$

Eq. (56-59) implies that

$$\sup(\|\nabla^2_{w_k^{[l]}}\hat{F}_k\|_2) \le \sup(\|\nabla^2_{w_k^{[l]}}\tilde{F}_k\|_2) \tag{60}$$

Therefore, we conclude that the upper bound of the gradient Lipschitz constant with respect to $\hat{w}_k^{[l]}$ at any l-th block for the network with mean removal is lower than that for the non-normalized network.

(2). We also first analyze the network with mean removal. At the l-th block, $\hat{y}_{i_k}^{[l]} = w_k^{[l]}\hat{x}_{i_k}^{[l]}$ where $w_k^{[l]} \in \mathcal{R}^{d_2\times d_1}$, $\hat{x}_{i_k}^{[l]} \in \mathcal{R}^{d_1\times 1}$ and $\hat{y}_{i_k}^{[l]} \in \mathcal{R}^{d_2\times 1}$, and then we have

$$\nabla_{w_k^{[l]}}\hat{f}_{i_k} = \nabla_{\hat{y}_{i_k}^{[l]}}\hat{f}_{i_k}(\hat{x}_{i_k}^{[l]})^\top. \tag{61}$$

Hence

$$\begin{aligned}\mathbb{E}[\|\nabla_{w_k^{[l]}}\hat{f}_{i_k}\|_2^2] =&\mathbb{E}[\|\nabla_{\hat{y}_{i_k}^{[l]}}\hat{f}_{i_k}(\hat{x}_{i_k}^{[l]})^\top\|_2^2]\\ \le&\mathbb{E}[\|\nabla_{\hat{y}_{i_k}^{[l]}}\|_2^2]\mathbb{E}[\|\hat{x}_{i_k}^{[l]}\|_2^2]\end{aligned} \tag{62}$$

Now we analyze the network with no normalization. Similar to Eq. (62), we can obtain

$$\mathbb{E}[\|\nabla_{w_k^{[l]}}\tilde{f}_{i_k}\|_2^2] \le \mathbb{E}[\|\nabla_{\tilde{y}_{i_k}^{[l]}}\tilde{f}_{i_k}\|_2^2]\mathbb{E}[\|\tilde{x}_{i_k}^{[l]}\|_2^2] \tag{63}$$

According to Lemma 3, we know

$$\sup(\mathbb{E}[\|\hat{x}_{i_k}^{[l]}\|_2^2]) \le \sup(\mathbb{E}[\|\tilde{x}_{i_k}^{[l]}\|_2^2]), \tag{64}$$

and

$$\sup(\mathbb{E}[\|\nabla_{\hat{y}_k^{[l]}}\hat{f}_{i_k}\|_2)]) \le \sup(\mathbb{E}[\|\nabla_{\tilde{y}_k^{[l]}}\tilde{f}_{i_k}\|_2]). \tag{65}$$

Eq. (62-65) implies that

$$\sup(\mathbb{E}[\|\nabla_{w_k^{[l]}}\hat{f}_{i_k}\|_2^2]) \le \sup(\mathbb{E}[\|\nabla_{w_k^{[l]}}\tilde{f}_{i_k}\|_2^2]) \tag{66}$$

Therefore, we conclude that at any l-th block the upper bound of the gradient squared expectation $\mathbb{E}[\|\nabla^{[l]}_{w_k}\hat{f}_{i_k}\|^2]$ for the network with mean removal is lower than the upper bound of the gradient squared expectation $\mathbb{E}[\|\nabla^{[l]}_{w_k}f_{i_k}\|^2]$ for the non-normalized network. □

Appendix F
Lemma 4

Lemma 4. *Given $f(u) = f(\frac{z}{\|z\|_2})$ where $z \in \mathcal{R}^{n\times 1}$ and $u \in \mathcal{R}^{n\times 1}$, we have*

$$\|\nabla f_z(z)\|_2 \le \frac{1}{\|z\|_2}\|\nabla f_u(u)\|_2 \tag{67}$$

and

$$\|\nabla^2 f_z(z)\|_2 \le \frac{1}{\|z\|_2^2}\|\nabla^2 f_u(u)\|_2 \tag{68}$$

Proof. Since $f(u) = f(\frac{z}{\|z\|_2})$, we have

$$\begin{aligned}\partial f =&(\nabla f_u(u))^\top\partial u\\ =&\mathrm{Tr}\left(\nabla f_u^\top(u)\partial(\frac{z}{\|z\|_2})\right)\\ =&\mathrm{Tr}\left(\nabla f_u^\top(u)\left(\frac{I}{\|z\|_2}-\frac{zz^\top}{\|z\|_2^3}\right)\partial z\right).\end{aligned} \tag{69}$$

Hence,

$$\nabla f_z(z) = \frac{\partial f}{\partial z} = \left(\frac{I}{\|z\|_2}-\frac{zz^\top}{\|z\|_2^3}\right)\nabla f_u(u). \tag{70}$$

Then, we obtain

$$\begin{aligned}\|\nabla f_z(z)\|_2 =&\left\|\left(\frac{I}{\|z\|_2}-\frac{zz^\top}{\|z\|_2^3}\right)\nabla f_u(u)\right\|_2\\ =&\frac{1}{\|z\|_2}\left\|\left(I-\frac{zz^\top}{\|z\|_2^2}\right)\nabla f_u(u)\right\|_2\\ \le&\frac{1}{\|z\|_2}\left\|I-\frac{zz^\top}{\|z\|_2^2}\right\|_2\|\nabla f_u(u)\|_2\\ =&\frac{1}{\|z\|_2}\|\nabla f_u(u)\|_2,\end{aligned} \tag{71}$$

where the inequality results from Lemma 2.

We further have

$$\begin{aligned}&\mathrm{vec}(\partial(\nabla f_z(z))) = \mathrm{vec}\left(\partial\left(\left(\frac{I}{\|z\|_2}-\frac{zz^\top}{\|z\|_2^3}\right)\nabla f_u(u)\right)\right)\\ =&\left(\frac{I}{\|z\|_2}-\frac{zz^\top}{\|z\|_2^3}\right)\mathrm{vec}\left(\partial(\nabla f_u(u))\right)\\ =&\left(\frac{I}{\|z\|_2}-\frac{zz^\top}{\|z\|_2^3}\right)\nabla^2 f_u(u)\mathrm{vec}\left(\partial\left(\frac{z}{\|z\|_2}\right)\right)\\ =&\left(\frac{I}{\|z\|_2}-\frac{zz^\top}{\|z\|_2^3}\right)\nabla^2 f_u(u)\mathrm{vec}\left(\left(\frac{I}{\|z\|_2}-\frac{zz^\top}{\|z\|_2^3}\right)\partial z\right)\\ =&\left(\frac{I}{\|z\|_2}-\frac{zz^\top}{\|z\|_2^3}\right)\nabla^2 f_u(u)\left(\frac{I}{\|z\|_2}-\frac{zz^\top}{\|z\|_2^3}\right)\mathrm{vec}(\partial z)\end{aligned} \tag{72}$$

Therefore,

$$\begin{aligned}\nabla^2 f_z(z) =&\frac{\mathrm{vec}(\partial(\nabla f_z(z)))}{\mathrm{vec}(\partial z)}\\ =&\left(\frac{I}{\|z\|_2}-\frac{zz^\top}{\|z\|_2^3}\right)\nabla^2 f_u(u)\left(\frac{I}{\|z\|_2}-\frac{zz^\top}{\|z\|_2^3}\right)\end{aligned} \tag{73}$$

Then, we obtain

$$
\begin{aligned}
\|\nabla^2 f_z(z)\|_2 =& \left\| \left(\frac{I}{\|z\|_2} - \frac{zz^\top}{\|z\|_2^3} \right) \nabla^2 f_u(u) \left(\frac{I}{\|z\|_2} - \frac{zz^\top}{\|z\|_2^3} \right) \right\|_2 \\
\leq& \left\| \frac{I}{\|z\|_2} - \frac{zz^\top}{\|z\|_2^3} \right\|_2^2 \|\nabla_u^2 f(u)\|_2 \\
=& \frac{1}{\|z\|_2^2} \|\nabla^2 f_u(u)\|_2.
\end{aligned}
\tag{74}
$$

where the inequality still results from Lemma 2. □

Appendix G Lemma 5

Lemma 5. *A standard network with no normalization and a standard network with normalization via l_2-Norm are respectively shown in Figure 1(a) and Figure 1(c). The inputs of the first block and the outputs of the last block for the both networks satisfy $\hat{x}_{i_k}^{[1]} = \tilde{x}_{i_k}^{[1]}$, $\|\nabla_{\hat{x}_{i_k}^{[L]}} \hat{f}_{i_k}\|_2 = \|\nabla_{\tilde{x}_{i_k}^{[L]}} \tilde{f}_{i_k}\|_2$ and $\|\nabla^2_{\hat{x}_{i_k}^{[L]}} \hat{f}_{i_k}\|_2 = \|\nabla^2_{\tilde{x}_{i_k}^{[L]}} \tilde{f}_{i_k}\|_2$. Suppose that at any l-th block with normalization via l_2-Norm $\tau_k^{[l]} = \frac{\gamma_k^{[l]}}{\rho \hat{\sigma}_{\mathcal{B}_k}^{[l]}} < 1$, we then have the following at any block:*

(1). The upper bound of $\mathbb{E}[\|\hat{x}_{i_k}^{[l]}\|_2^2]$ is less than the upper bound of $\mathbb{E}[\|\tilde{x}_{i_k}^{[l]}\|_2^2]$.

(2). The upper bound of $\mathbb{E}[\|\nabla_{\hat{x}_{i_k}^{[l]}} \hat{f}_{i_k}\|_2]$ is less than the upper bound of $\mathbb{E}[\|\nabla_{\tilde{x}_{i_k}^{[l]}} \tilde{f}_{i_k}\|_2]$.

(3). The upper bound of $\|\nabla^2_{\hat{x}_k^{[l]}} \hat{F}_k\|_2$ is less than the upper bound of $\|\nabla^2_{\tilde{x}_k^{[l]}} \tilde{F}_k\|_2$.

Proof. We use mathematical induction to prove the propositions.

(1) We first to comparative analysis for the forward paths for the network with mean removal and the network without normalization.

For the inductive step, we assume the following inequality holds, *i.e.*,

$$
\sup(\mathbb{E}[\|\hat{x}_{i_k}^{[l]}\|_2^2]) \leq \sup(\mathbb{E}[\|\tilde{x}_{i_k}^{[l]}\|_2^2]),
\tag{75}
$$

where $\mathcal{U}$ is the universal sample set.

We analyze the network with mean removal. It is known $\hat{x''}_{i_k}^{[l]} = \frac{\gamma_k^{[l]}}{\rho} \cdot \frac{\hat{x'}_{i_k}^{[l]}}{\hat{\sigma}_{\mathcal{B}_k}^{[l]}}$ and $\frac{\gamma_k^{[l]}}{\rho \hat{\sigma}_{\mathcal{B}_k}^{[l]}} \leq 1$, so we have

$$
\|\hat{x''}_{i_k}^{[l]}\|_2^2 \leq \|\hat{x}_{i_k}^{[l]}\|_2^2.
\tag{76}
$$

And then following Eq.(30-31) in proof of Lemma 3, we obtain

$$
\begin{aligned}
\mathbb{E}[\|\hat{x}_{i_k}^{[l+1]}\|_2^2] \leq& \left(\mathbb{I}(\mathrm{SC}) + \hat{\zeta}_{i_k}^{[l]} \|w_k^{[l]}\|_2^2 \right) \mathbb{E}[\|\hat{x}_{i_k}^{[l]}\|_2^2] \\
\leq& \left(\mathbb{I}(\mathrm{SC}) + \|w_k^{[l]}\|_2^2 \right) \mathbb{E}[\|\hat{x}_{i_k}^{[l]}\|_2^2].
\end{aligned}
\tag{77}
$$

For the l-th layer in the network without normalization, in Eq.(33) in the proof of Lemma 3, we also have

$$
\begin{aligned}
\mathbb{E}[\|\tilde{x}_{i_k}^{[l+1]}\|_2^2] \leq& \left(\mathbb{I}(\mathrm{SC}) + \tilde{\zeta}_{i_k}^{[l]} \|w_k^{[l]}\|_2^2 \right) \mathbb{E}[\|\tilde{x}_{i_k}^{[l]}\|_2^2]. \\
\leq& \left(\mathbb{I}(\mathrm{SC}) + \|w_k^{[l]}\|_2^2 \right) \mathbb{E}[\|\tilde{x}_{i_k}^{[l]}\|_2^2].
\end{aligned}
\tag{78}
$$

Combining Eq.(75), Eq.(77) and Eq.(78)

$$
\sup(\mathbb{E}[\|\hat{x}_{i_k}^{[l+1]}\|_2^2]) \leq \sup(\mathbb{E}[\|\tilde{x}_{i_k}^{[l+1]}\|_2^2]),
\tag{79}
$$

where this completes the inductive step. We know that the inputs of the network with mean removal and the non-normalized network are the the same, *i.e.*, $\hat{x}_{i_k}^{[1]} = \tilde{x}_{i_k}^{[1]}$. Since both the base case and the induction step have been proved as true, by mathematical induction the statement $\sup(\mathbb{E}[\|\hat{x}_{i_k}^{[s]}\|_2^2]) \leq \sup(\mathbb{E}[\|\tilde{x}_{i_k}^{[s]}\|_2^2])$ holds for any layer s $(1 \leq s \leq L)$.

(2) We first comparatively analyze the backward paths of the network with mean removal and the network without normalization.

For the inductive step, we assume that the upper bound of $\|\nabla_{\hat{x}_k^{[l+1]}} \hat{f}_{i_k}\|_2$ is less than the upper bound of $\|\nabla_{\hat{x}_k^{[l+1]}} \tilde{f}_{i_k}\|_2$ at the l-th layer holds, *i.e.*,

$$
\sup(\mathbb{E}[\|\nabla_{\hat{x}_k^{[l+1]}} \hat{f}_{i_k}\|_2]) \leq \sup(\mathbb{E}[\|\nabla_{\tilde{x}_k^{[l+1]}} \tilde{f}_{i_k}\|_2])
\tag{80}
$$

We first analyze the network with normalization via ℓ_2-Norm. Similar to Eq.(36-37) in the proof of Lemma 3, we can obtain

$$
\begin{aligned}
\|\nabla_{\hat{x''}_k^{[l]}} \hat{f}_{i_k}\| =& \|w_k^{[l]} (\delta'(\hat{y}_{i_k}^{[l]}) \odot \nabla_{\hat{x}_{i_k}^{[l+1]}} \hat{f}_{i_k})\| \\
\leq& \|w_k^{[l]}\| \|\nabla_{\hat{x}_{i_k}^{[l+1]}} \hat{f}_{i_k}\|,
\end{aligned}
\tag{81}
$$

where $\|\delta'(\hat{y}_{i_k}^{[l]})\| \leq 1$ since the derivative of the nonlinear activation $\delta(u)$ with respect to u is less than 1.

It is known $\hat{x''}_{i_k}^{[l]} = \frac{\gamma_k^{[l]}}{\rho} \cdot \frac{\hat{x}_{i_k}^{[l]}}{\hat{\sigma}_{\mathcal{B}_k}^{[l]}}$ where $\hat{\sigma}_{\mathcal{B}_k}^{[l]} = \sqrt{\frac{1}{|\mathcal{B}_k|} \sum_{i_k \in \mathcal{B}_k} (\hat{x}_{i_k}^{[l]})^2} = \frac{1}{\sqrt{|\mathcal{B}_k|}} \|\hat{x}_k^{[l]}\|_2$, and we define

$$
\hat{x}_{\mathcal{B}_k}^{[l]} = \left[\hat{x}_{1_k}^{[l]}, \hat{x}_{2_k}^{[l]}, ..., \hat{x}_{|\mathcal{B}|_k}^{[l]} \right]^\top
\tag{82}
$$

and

$$
\hat{x''}_{\mathcal{B}_k}^{[l]} = \left[\hat{x''}_{1_k}^{[l]}, \hat{x''}_{2_k}^{[l]}, ..., \hat{x''}_{|\mathcal{B}|_k}^{[l]} \right]^\top.
\tag{83}
$$

Then, we have

$$
(\hat{x''}_{\mathcal{B}_k}^{[l]})_j = \frac{\gamma_k^{[l]} \sqrt{|\mathcal{B}_k|}}{\rho} \frac{(\hat{x}_{\mathcal{B}_k}^{[l]})_j}{\|(\hat{x}_{\mathcal{B}_k}^{[l]})_j\|}
\tag{84}
$$

where $j \in [1, 2, ..., d]$ and d is the number of dimension of $\hat{x}_{1_k}^{[l]}$ and $\hat{x''}_{1_k}^{[l]}$.

Following Lemma 4, we obtain

$$
\begin{aligned}
\left\| \left(\nabla_{\hat{x}_{\mathcal{B}_k}^{[l]}} \hat{f}_{\mathcal{B}_k}^{[l]} \right)_j \right\|_2 \leq& \frac{\gamma_k^{[l]} \sqrt{|\mathcal{B}_k|}}{\rho \|\hat{x}_{\mathcal{B}_k}^{[l]}\|_2} \left\| \left(\nabla_{\hat{x''}_k^{[l]}} \hat{f}_{\mathcal{B}_k}^{[l]} \right)_j \right\|_2 \\
=& \frac{\gamma_k^{[l]}}{\rho \hat{\sigma}_{\mathcal{B}_k}^{[l]}} \left\| \left(\nabla_{\hat{x''}_k^{[l]}} \hat{f}_{\mathcal{B}_k}^{[l]} \right)_j \right\|_2
\end{aligned}
\tag{85}
$$

Since $\frac{\gamma_k^{[l]}}{\rho \hat{\sigma}_{\mathcal{B}_k}^{[l]}} \leq 1$, Eq. (85) can ge further rearranged as

$$
\left\| \left(\nabla_{\hat{x}_{\mathcal{B}_k}^{[l]}} \hat{f}_{\mathcal{B}_k}^{[l]} \right)_j \right\|_2 \leq \left\| \left(\nabla_{\hat{x''}_{\mathcal{B}_k}^{[l]}} \hat{f}_{\mathcal{B}_k}^{[l]} \right)_j \right\|_2
\tag{86}
$$

We further have

$$\begin{aligned}\mathbb{E}\left[\left\|\nabla_{\hat{x}^{[l]}_{i_k}}\hat{f}_{i_k}\right\|^2\right] &= \frac{1}{|\mathcal{B}|}\mathbb{E}\left[\left\|\nabla_{\hat{x}^{[l]}_{\mathcal{B}_k}}\hat{f}_{i_k}\right\|^2\right] \\ &= \frac{1}{|\mathcal{B}|}\mathbb{E}\left[\sum_{j=1}^{d}\left\|\left(\nabla_{\hat{x}^{[l]}_{\mathcal{B}_k}}\hat{f}_{\mathcal{B}_k}\right)_j\right\|^2\right] \\ &\leq \frac{1}{|\mathcal{B}|}\mathbb{E}\left[\sum_{j=1}^{d}\left\|\left(\nabla_{\hat{x''}^{[l]}_{\mathcal{B}_k}}\hat{f}_{\mathcal{B}_k}\right)_j\right\|^2\right] \\ &= \frac{1}{|\mathcal{B}|}\mathbb{E}\left[\left\|\nabla_{\hat{x''}^{[l]}_{\mathcal{B}_k}}\hat{f}_{\mathcal{B}_k}\right\|^2\right] \\ &= \mathbb{E}\left[\left\|\nabla_{\hat{x''}^{[l]}_{i_k}}\hat{f}_{i_k}\right\|^2\right]\end{aligned} \tag{87}$$

Combining Eq.(81-87) and the fact $\hat{x}^{[l+1]}_{i_k} = \mathbb{I}(\mathrm{SC})\hat{x}^{[l]}_{i_k} + \delta(\hat{y}^{[l]}_{i_k})$ where $\mathbb{I}(\mathrm{SC})$ indicates whether there is a shortcut, we have

$$\mathbb{E}\left[\|\nabla_{\hat{x}^{[l]}_{i_k}}\hat{f}_{i_k}\|^2\right] \leq (\mathbb{I}(\mathrm{SC}) + \|w^{[l]}_k\|^2)\mathbb{E}[\|\nabla_{\hat{x}^{[l+1]}_{i_k}}\hat{f}_{i_k}\|^2] \tag{88}$$

In terms of the l-th layer in the network without normalization, recalling Eq.(44), we obtain

$$\begin{aligned}\mathbb{E}[\|\nabla_{\tilde{x}^{[l]}_{i_k}}\tilde{f}_{i_k}\|^2] =& \mathbb{E}[\|\mathbb{I}(\mathrm{SC}) + (w^{[l]}_k)^\top(\delta'(\tilde{y}^{[l]}_{i_k}) \odot \nabla_{\tilde{x}^{[l+1]}_{i_k}}\tilde{f}_{i_k})\|^2] \\ \leq& (\mathbb{I}(\mathrm{SC}) + \|w^{[l]}_k\|_2)\mathbb{E}[\|\nabla_{\tilde{x}^{[l+1]}_{i_k}}\tilde{f}_{i_k}\|^2].\end{aligned} \tag{89}$$

Combining Eq.(80), Eq.(88) and Eq.(89), we have

$$\sup(\mathbb{E}[\|\nabla_{\hat{x}^{[l]}_{i_k}}\hat{f}_{i_k}\|_2]) \leq \sup(\mathbb{E}[\|\nabla_{\tilde{x}^{[l]}_{i_k}}\tilde{f}_{i_k}\|]). \tag{90}$$

It completes the inductive step. We know that l_2-norm of the derivative of $\hat{f}_{i_k}$ with respect to $\hat{x}_{i_k}$ for the network with mean removal and $\tilde{f}$ with respect to $\tilde{x}_k$ for the network without normalization are equal, *i.e.*, $\|\nabla_{\hat{x}^{[L]}_{i_k}}\hat{f}_k\|^2 = \|\nabla_{\tilde{x}^{[L]}_k}\tilde{f}_k\|^2$. Since both the base case and the induction step have been proved as true, by mathematical induction the statement $\sup(\mathbb{E}[\|\nabla_{\hat{x}^{[s]}_k}\hat{f}_{i_k}\|^2]) \leq \sup(\mathbb{E}[\|\nabla_{\tilde{x}^{[s]}_k}\tilde{f}_{i_k}\|^2])$ holds for any layer s $(1 \leq s \leq L)$.

(3) We first comparatively analyze the backward paths of the network with mean removal and the network without normalization.

For the inductive step, we assume that the upper bound of $\|\nabla^2_{\hat{x}^{[l+1]}_k}\hat{F}_{i_k}\|_2$ is less than the upper bound of $\|\nabla^2_{\tilde{x}^{[l+1]}_k}\tilde{F}_{i_k}\|$ at the l-th layer holds, *i.e.*,

$$\sup(\|\nabla^2_{\hat{x}^{[l+1]}_k}\hat{F}_k\|_2) \leq \sup(\|\nabla^2_{\tilde{x}^{[l+1]}_k}\tilde{F}_k\|) \tag{91}$$

We analyze the network with normalization via ℓ_2-Norm. Similar to Eq.(46-47) in the proof of Lemma 3, we can obtain

$$\begin{aligned}\|\nabla^2_{\hat{y}^{[l]}_k}\hat{f}_{i_k}\| \leq& \|w^{[l]}_k\|^2\|\delta''(\hat{y}^{[l]}_{i_k}) \odot \nabla^2_{\hat{y'}^{[l]}_{i_k}}\hat{f}_{i_k}\| \\ \leq& \|w^{[l]}_k\|_2^2\|\nabla^2_{\hat{y'}^{[l]}_{i_k}}\hat{f}_{i_k}\|,\end{aligned} \tag{92}$$

where$\|\delta''(\hat{y}^{[l]}_{i_k})\| \leq 1$ since the second derivative of the nonlinear activation $\delta(u)$ with respect to u is less than 1.

It is known $\hat{x''}^{[l]}_{i_k} = \frac{\gamma^{[l]}_k}{\rho} \cdot \frac{\hat{x}^{[l]}_{i_k}}{\hat{\sigma}^{[l]}_{\mathcal{B}_k}}$ where $\hat{\sigma}^{[l]}_{\mathcal{B}_k} = \sqrt{\frac{1}{|\mathcal{B}_k|}\sum_{i_k\in\mathcal{B}_k}(\hat{x}^{[l]}_{i_k})^2} = \frac{1}{\sqrt{|\mathcal{B}_k|}}\|\hat{x}^{[l]}_k\|_2$, and the definitions of $\hat{x}^{[l]}_{\mathcal{B}_k}$ and $\hat{x''}^{[l]}_{\mathcal{B}_k}$ are shown in Eq.(82) and Eq.(83). Following Lemma 3, we have

$$\begin{aligned}\left\|\left(\nabla^2_{\hat{x}^{[l]}_{\mathcal{B}_k}}\hat{f}^{[l]}_{\mathcal{B}_k}\right)_j\right\|_2 \leq& \left(\frac{\gamma^{[l]}_k\sqrt{|\mathcal{B}_k|}}{\rho\|\hat{x}^{[l]}_{\mathcal{B}_k}\|_2}\right)^2\left\|\left(\nabla^2_{\hat{x''}^{[l]}_k}\hat{f}^{[l]}_{\mathcal{B}_k}\right)_j\right\|_2 \\ =& \left(\frac{\gamma^{[l]}_k}{\rho\hat{\sigma}^{[l]}_{\mathcal{B}_k}}\right)^2\left\|\left(\nabla^2_{\hat{x''}^{[l]}_k}\hat{f}^{[l]}_{\mathcal{B}_k}\right)_j\right\|_2\end{aligned} \tag{93}$$

Since $\frac{\gamma^{[l]}_k}{\rho\hat{\sigma}^{[l]}_{\mathcal{B}_k}} \leq 1$, Eq. (93) can ge further rearranged as

$$\left\|\left(\nabla^2_{\hat{x}^{[l]}_{\mathcal{B}_k}}\hat{f}^{[l]}_{\mathcal{B}_k}\right)_j\right\|_2 \leq \left\|\left(\nabla^2_{\hat{x''}^{[l]}_{\mathcal{B}_k}}\hat{f}^{[l]}_{\mathcal{B}_k}\right)_j\right\|_2. \tag{94}$$

We further have

$$\begin{aligned}\mathbb{E}\left[\left\|\nabla^2_{\hat{x}^{[l]}_{i_k}}\hat{f}_{i_k}\right\|^2\right] &= \frac{1}{|\mathcal{B}|}\mathbb{E}\left[\left\|\nabla^2_{\hat{x}^{[l]}_{\mathcal{B}_k}}\hat{f}_{i_k}\right\|^2\right] \\ &= \frac{1}{|\mathcal{B}|}\mathbb{E}\left[\sum_{j=1}^{d}\left\|\left(\nabla^2_{\hat{x}^{[l]}_{\mathcal{B}_k}}\hat{f}_{\mathcal{B}_k}\right)_j\right\|^2\right] \\ &\leq \frac{1}{|\mathcal{B}|}\mathbb{E}\left[\sum_{j=1}^{d}\left\|\left(\nabla^2_{\hat{x''}^{[l]}_{\mathcal{B}_k}}\hat{f}_{\mathcal{B}_k}\right)_j\right\|^2\right] \\ &= \frac{1}{|\mathcal{B}|}\mathbb{E}\left[\left\|\nabla^2_{\hat{x''}^{[l]}_{\mathcal{B}_k}}\hat{f}_{\mathcal{B}_k}\right\|^2\right] \\ &= \mathbb{E}\left[\left\|\nabla^2_{\hat{x''}^{[l]}_{i_k}}\hat{f}_{i_k}\right\|^2\right]\end{aligned} \tag{95}$$

Combining Eq.(92-95) and the fact $\hat{x}^{[l+1]}_{i_k} = \mathbb{I}(\mathrm{SC})\hat{x}^{[l]}_{i_k} + \delta(\hat{y}^{[l]}_{i_k})$ where $\mathbb{I}(\mathrm{SC})$ indicates whether there is a shortcut, we have

$$\begin{aligned}\|\nabla^2_{\hat{x}^{[l]}_k}\hat{F}_{i_k}\|^2 \leq& \mathbb{E}[\|\nabla^2_{\hat{x}^{[l]}_{i_k}}\hat{f}_{i_k}\|^2] \\ \leq& (\mathbb{I}(\mathrm{SC}) + \|w^{[l]}_k\|^4)\mathbb{E}[\|\nabla^2_{\hat{x}^{[l+1]}_{i_k}}\hat{f}_{i_k}\|^2]\end{aligned} \tag{96}$$

In terms of the l-th layer in the network without normalization, recalling Eq.(52), we obtain

$$\begin{aligned}\|\nabla^2_{\tilde{x}^{[l]}_k}\tilde{F}_k\|^2 \leq& \mathbb{E}[\|\nabla^2_{\tilde{x}^{[l]}_{i_k}}\tilde{f}_{i_k}\|^2] \\ =& \mathbb{E}\Big[\mathbb{I}(\mathrm{SC})\nabla^2_{\tilde{y'}^{[l]}_{i_k}}\tilde{f}_{i_k} + \|(I \otimes (w^{[l]}_k)^\top) \\ & (\delta''(\tilde{y}^{[l]}_{i_k}) \odot \nabla^2_{\tilde{y'}^{[l]}_{i_k}}\tilde{f}_{i_k})(I \otimes w^{[l]}_k)\|^2\Big] \\ \leq& (\mathbb{I}(\mathrm{SC}) + \|w^{[l]}_k\|_2^4)\mathbb{E}[\|\nabla^2_{\tilde{x}^{[l+1]}_{i_k}}\tilde{f}_{i_k}\|^2]\end{aligned} \tag{97}$$

Combining Eq.(91), Eq.(96) and Eq.(97), we obtain

$$\sup(\|\nabla^2_{\hat{x}^{[l]}_k}\hat{F}_k\|_2) \leq \sup(\|\nabla^2_{\tilde{x}^{[l]}_k}\tilde{F}_k\|), \tag{98}$$

It completes the inductive step. We know that l_2-norm of the second-order derivative of $\hat{f}_{i_k}$ with respect to $\hat{x}_{i_k}$ for the network with mean removal and $\tilde{f}$ with respect to $\tilde{x}_{i_k}$ for the network without normalization are equal, *i.e.*, $\|\nabla^2_{\hat{x}^{[L]}_k}\hat{F}_k\|_2 \leq$

$\|\nabla^2_{\tilde{x}_k^{[L]}} \tilde{F}_k\|$. Since both the base case and the induction step have been proved as true, by mathematical induction the statement $\sup(\|\nabla^2_{\hat{x}_k^{[s]}} \hat{F}_k\|_2) \leq \sup(\|\nabla^2_{\tilde{x}_k^{[s]}} \tilde{F}_k\|)\|_2$ holds for any layer s ($1 \leq s \leq L$). □

Appendix H Proof of Theorem 3

Proof. (1) We analyze the network with normalization via l_2-Norm. Following Eq.(55-56), we can obtain, we obtain

$$\|\nabla^2_{w_k^{[l]}} \hat{F}_k\|_2 \leq \mathbb{E}\left[\left\|\hat{x}_{i_k}^{[l]}\right\|_2^2\right] \left\|\nabla^2_{\hat{y}_k^{[l]}} \hat{F}_k\right\|_2 \tag{99}$$

For the network with no normalization, from Eq.(57)we know

$$\|\nabla^2_{w_k^{[l]}} \tilde{F}_k\|_2 \leq \mathbb{E}\left[\left\|\tilde{x}_{i_k}^{[l]}\right\|_2^2\right] \left\|\nabla^2_{\tilde{y}_k^{[l]}} \tilde{F}_k\right\|_2 \tag{100}$$

According to Lemma 5, we know

$$\sup(\mathbb{E}[\|\hat{x}_{i_k}^{[l]}\|_2^2]) \leq \sup(\mathbb{E}[\|\tilde{x}_{i_k}^{[l]}\|_2^2]), \tag{101}$$

and

$$\sup(\|\nabla^2_{\hat{y}_k^{[l]}} \hat{F}_k\|_2) \leq \sup(\|\nabla^2_{\tilde{y}_k^{[l]}} \tilde{F}_k\|). \tag{102}$$

Eq. (99-102) implies that

$$\sup(\|\nabla^2_{w_k^{[l]}} \hat{F}_k\|_2) \leq \sup(\|\nabla^2_{w_k^{[l]}} \tilde{F}_k\|_2) \tag{103}$$

Therefore, we conclude that the upper bound of the gradient Lipschitz constant of $\hat{F}$ with respect to $w_k^{[l]}$ for the network with normalization via ℓ_2-norm is lower than the upper bound of the gradient Lipschitz constant of $\tilde{F}$ with respect to $w_k^{[l]}$ for the non-normalized network.

(2) We analyze the network with with normalization with l_2-Norm. Following Eq.(62), we obtain

$$\mathbb{E}[\|\nabla_{w_k^{[l]}} \tilde{f}_{i_k}\|_2^2] \leq \mathbb{E}[\|\nabla_{\tilde{y}_{i_k}^{[l]}} \tilde{f}_{i_k}\|_2^2]\mathbb{E}[\|\tilde{x}_{i_k}^{[l]}\|_2^2] \tag{104}$$

Now we analyze the network with no normalization. From Eq. (63), we know

$$\mathbb{E}[\|\nabla_{w_k^{[l]}} \hat{f}_{i_k}\|_2^2] \leq \mathbb{E}[\|\nabla_{\hat{y}_{i_k}^{[l]}} \tilde{f}_{i_k}\|_2^2]\mathbb{E}[\|\hat{x}_{i_k}^{[l]}\|_2^2] \tag{105}$$

According to Lemma 5, we know

$$\sup(\mathbb{E}[\|\hat{x}_{i_k}^{[l]}\|_2^2]) \leq \sup(\mathbb{E}[\|\tilde{x}_{i_k}^{[l]}\|_2^2]), \tag{106}$$

and

$$\sup(\mathbb{E}[\|\nabla_{\hat{y}_k^{[l]}} \hat{f}_{i_k}\|_2)] \leq \sup(\mathbb{E}[\|\nabla_{\tilde{y}_k^{[l]}} \tilde{f}_{i_k}\|_2]). \tag{107}$$

Eq. (104-107) implies that

$$\sup(\mathbb{E}[\|\nabla_{w_k^{[l]}} \hat{f}_{i_k}\|_2^2]) \leq \sup(\mathbb{E}[\|\nabla_{w_k^{[l]}} \tilde{f}_{i_k}\|_2^2]) \tag{108}$$

Therefore, we conclude that the upper bound of the gradient squared expectation $\mathbb{E}[\|\nabla^{[l]}_{w_k} \hat{f}_{i_k}\|^2]$ for any l-th block in the network with normalization with ℓ_2-norm is lower than $\mathbb{E}[\|\nabla^{[l]}_{w_k} \tilde{f}_{i_k}\|^2]$ for any l-th block in the non-normalized network. □

Appendix I Theoretical Analysis for Adaptive Moving Average Statistics

In the main manuscript, we demonstrate reducing the variance of batch statistics directly decrease the gradient variance and ultimately make training converged minima smaller. In this subsection, we justify the idea that substituting statistics over a single batch with adaptive moving average statistics reduces the variance and brings few side effects.

We can utilize the moving average statistics to reduce the variance because the moving average increases the number size of batch statistics by exploiting historical information. It is easy to know that the batch statistics in the l-th block ($\mu_{\mathcal{B}_k}^{[l]}$ and $\sigma_{\mathcal{B}_k}^{[l]}$) are the functions of the weights to be learned in the first to the $(l-1)$-th block ($w_k^{[1]}, w_k^{[2]}, ..., w_k^{[1-1]}$), and the weights at the historical iteration and the current iteration are different, which will inevitably results in bias. We take the moving average mean as an example, *i.e.*,

$$\begin{aligned}\mu_{\mathcal{A}_k}^{[l]} =& \eta\mu_{\mathcal{A}_{k-1}}^{[l]} + (1-\eta)\mu_{\mathcal{B}_k}^{[l]} \\ =& \sum_{s=1}^{k}(1-\eta)\eta^{k-s}\mu_{\mathcal{B}_s}^{[l]} \\ =& \underbrace{\sum_{s=1}^{k}(1-\eta)\eta^{k-s}\mu_{\mathcal{B}_{s\to k}}^{[l]}}_{\text{variance reduction}} + \underbrace{\sum_{s=1}^{k}(1-\eta)\eta^{k-s}(\mu_{\mathcal{B}_s}^{[l]} - \mu_{\mathcal{B}_{s\to k}}^{[l]})}_{\text{bias increase}},\end{aligned} \tag{109}$$

where $\mu_{\mathcal{B}_s}^{[j]} = g(w_s^{[1]}, w_s^{[2]}, ..., w_s^{[l-1]}; x_s^{[0]})$ and $\mu_{\mathcal{B}_{s\to k}}^{[j]} = g(w_k^{[1]}, w_k^{[2]}, ..., w_k^{[l-1]}; x_s^{[0]})$.

Therefore, we should carefully tune the parameter η; otherwise, the side effects resulting from increased bias will exceed the benefits stemming from variance reduction. Specifically, if the difference between $\mu_{\mathcal{B}_s}^{[l]}$ and $\mu_{\mathcal{B}_{s\to k}}^{[l]}$ is small, we can set η large to enjoy greater variance reduction, and vice versa. Thus, η is better to adaptively track the difference. A network is commonly β-Lipschitz smooth, *i.e.*,

$$\left\|\mu_{\mathcal{B}_s}^{[l]} - \mu_{\mathcal{B}_{s\to k}}^{[l]}\right\| \leq \beta \sum_{p=1}^{l-1} \left\|w_s^{[p]} - w_k^{[p]}\right\|. \tag{110}$$

According to the SGD update rule in Eq. (1) in the main manuscript, we know that

$$w_s^{[p]} - w_k^{[p]} = \sum_{r=s}^{k} \alpha_r \nabla_{w_r^{[p]}} f_{\mathcal{B}_r}, \tag{111}$$

where α_r is the learning rate and $f_{\mathcal{B}_r}$ is the gradient with respect to $w_r^{[p]}$ at the r-th iteration.

In summary, loosely speaking, the learning rate α is proportional to the differences between $\mu_{\mathcal{B}_s}^{[l]}$ and $\mu_{\mathcal{B}_{s\to k}}^{[l]}$. Thus, if we set η to be a decreasing function of the learning rate α, the moving average statistics will adaptively strike a wise balance between variance reduction and bias increase.

Hanyang Peng is currently an assistant professor in Pengcheng Laboratory, Shenzhen, China. He received a B.S. degree in measurement and control technology from the Northeast University of China, Shenyang, China, in 2008, an M.E. degree in detection technology and automatic equipment from the Tianjin University of China, Tianjin, China, in 2010, and a Ph.D. degree in pattern recognition and intelligence systems from the Institute of Automation, Chinese Academy of Sciences, Beijing, China, in 2017. He currently works His current research interests include optimization in deep learning and distributed learning.

Yue Yu is a researcher at Pengcheng Lab, an associate professor in the College of Computer at National University of Defense Technology, and the technical committee member of OpenI community. His research findings have been published on TSE, TOSEM, CHI, CSCW, ICSE, FSE, ACL etc. His current research interests include software engineering and artificial Intelligence.

Shiqi Yu is currently an Associate Professor in the Department of Computer Science and Engineering, Southern University of Science and Technology, China. He received his B.E. degree in computer science and engineering from the Chu Kochen Honors College, Zhejiang University in 2002, and Ph.D. degree in pattern recognition and intelligent systems from the Institute of Automation, Chinese Academy of Sciences in 2007. He worked as an Assistant Professor and an Associate Professor in Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences from 2007 to 2010, and as an Associate Professor in Shenzhen University from 2010 to 2019. His research interests include gait recognition, face detection and computer vision.