Batch Normalization Is Blind to the First and Second Derivatives of the Loss

Zhanpeng Zhou[1*], Wen Shen[1*], Huixin Chen[1*], Ling Tang[1], Yuefeng Chen[2], Quanshi Zhang[1†]

[1]Shanghai Jiao Tong University
[2]Alibaba Group
{zzp1012, wen_shen, lural_chen, tling, zqs1022}@sjtu.edu.cn
yuefeng.chenyf@alibaba-inc.com

Abstract

We prove that when we do the Taylor series expansion of the loss function, the BN operation will block the influence of the first-order term and most influence of the second-order term of the loss. We also find that such a problem is caused by the standardization phase of the BN operation. We believe that proving the blocking of certain loss terms provides an analytic perspective for potential detects of a deep model with BN operations, although the blocking problem is not fully equivalent to significant damages in all tasks on benchmark datasets. Experiments show that the BN operation significantly affects feature representations in specific tasks.

Introduction

Explaining the representation capacity of a deep neural network (DNN) has received increasing attention in recent years. Since people have grasped much experimental experience in applying DNNs, the next challenge has been gradually shifted to a verifiable and analytic explanation for technical flaws of current heuristic network designs, or analytic insights into how DNNs learn features. For example, Tian et al. (2021) provided conceptual insights into how non-contrastive self-supervised learning methods avoided representational collapse. Deng et al. (2022) proved that it was usually difficult for a DNN to encode interactions between an intermediate number of input variables. Some studies proved that the BN operation was not compatible with both the dropout (Li et al. 2019) and the weight decay (Van Laarhoven 2017; Li, Chen, and Yang 2020).

Therefore, in this paper, we aim to analyze the distinctive behavior of information processing for any DNNs with batch-normalization (BN) layers. Specifically, this paper discovers and theoretically proves that in a regression task, for a DNN with stacked BN layers, **each BN operation blocks the back-propagation of the first and second derivatives of the loss function *w.r.t.* features**, when we do the Taylor series expansion of the loss function *w.r.t.* the output of the BN. That is, for a DNN with an *arbitrary* loss function (*e.g.*, MSE loss, logistic loss, etc.), network layers before the BN layer cannot learn from all first derivatives and some second derivatives of the loss function.

Note that for some certain tasks on benchmark datasets, the BN's blocking of certain derivatives may not lead to a significant damage of the network learning. However, clean and analytic insights into BN's potential risks may provide a new perspective for further understanding of DNNs.

In this paper, we consider using the Taylor series expansion to represent the loss[1] $\text{Loss}(\mathbf{y}^{(i)})$, where $\mathbf{y}^{(i)} \in \mathbb{R}^D$ denotes the output feature of the standardization phase (subtracting the mean and dividing the standard deviation) of the BN operation, given the i-th input sample. We find that no matter whether or not all samples share the same analytic formula of the loss, we always can use the same form of Taylor series expansion to represent the loss, *i.e.*, $\text{Loss}(\mathbf{y}^{(i)};\tilde{\mathbf{y}}) = \text{Loss}(\tilde{\mathbf{y}}) + (\mathbf{y}^{(i)}-\tilde{\mathbf{y}})^\top \widehat{\mathbf{g}} + \frac{1}{2!}(\mathbf{y}^{(i)}-\tilde{\mathbf{y}})^\top \widehat{\mathbf{H}}(\mathbf{y}^{(i)}-\tilde{\mathbf{y}}) + \widehat{\Delta}^{(i)}$, where $\widehat{\mathbf{g}}$ and $\widehat{\mathbf{H}}$ represent the gradient and the Hessian matrix of $\text{Loss}(\mathbf{y}^{(i)};\tilde{\mathbf{y}})$, or alternatively, the substitute gradient and Hessian matrix[2] to alleviate errors caused by discontinuous and unpredictable gating states in ReLU layers and influences of different analytic formulas of the loss. $\tilde{\mathbf{y}}$ is an arbitrary vector close to $\mathbf{y}^{(i)}$. When $\mathbf{y}^{(i)}$ is not close to the fixed point $\tilde{\mathbf{y}}$, the residual term $\widehat{\Delta}^{(i)}$ will be relatively large. Nevertheless, the blocking of the first-order and the second-order terms still has certain effects on learning.

Therefore, we have discovered and proved the following conclusions, as shown in Theorem 2 and Corollary 1.
1. *The gradient $\widehat{\boldsymbol{g}}$ at the point $\tilde{\mathbf{y}}$ cannot affect the learning of network parameters before the BN layer.*
2. *Diagonal elements in the Hessian matrix $\widehat{\boldsymbol{H}}$ at the point $\tilde{\mathbf{y}}$ cannot affect the learning of parameters before the BN layer.*
3. *For off-diagonal elements in $\widehat{\boldsymbol{H}}$ at the point $\tilde{\mathbf{y}}$, their impacts on learning parameters before BN are significantly reduced.*
4. *It is the derivatives of the mean and the standard deviation*

*These authors contributed equally.
†Quanshi Zhang is the corresponding author. He is with the Department of Computer Science and Engineering, the John Hopcroft Center, at the Shanghai Jiao Tong University, China.

[1]$\text{Loss}(\mathbf{y})$ is a simplification of $\text{Loss}(h(\mathbf{y}))$, where $h(\cdot): \mathbb{R}^D \to \mathbb{R}$ denotes the function of network layers between the BN operation and the loss function. We omit $h(\cdot)$ to simplify the description.
[2]Subsection "Case 2: A More General Case" introduces how to compute substitute $\widehat{\mathbf{g}}$ and $\widehat{\mathbf{H}}$ to approximate input samples with different loss formulas.

in the standardization phase that eliminate the influence of certain loss elements.

Although the affine transformation phase in the BN allows the DNN to learn from the gradient $\widehat{\mathbf{g}}$ and mitigate the blocking problem, to some extent, experiments have shown that the blocking problem still significantly affects the learning of specific DNNs. Furthermore, according to our theory, if we replace the BN with the layer normalization (LN) (Ba, Kiros, and Hinton 2016), then the learning process would no longer suffer from the blocking problem, although it is hard to say whether LN is an ideal substitute of BN. Nevertheless, an analytic understanding of the BN's blocking problem may provide new distinctive guidance in future development in deep learning theory.

Related Work

Some studies have discussed negative effects of normalization methods on representation learning in some applications. In experiments, the BN usually hurts the image generation quality of GAN (Goodfellow et al. 2014) and classification accuracy (Xie et al. 2020; Galloway et al. 2019; Wightman, Touvron, and Jegou 2021). The BN was not compatible with both the dropout (Li et al. 2019) and the weight decay (Van Laarhoven 2017; Li, Chen, and Yang 2020). Ablation experiments showed that the affine transformation in the LN increased the risk of over-fitting (Xu et al. 2019). Some studies have found the BN's problem when samples in a mini-batch were not independent in experiments (Ioffe 2017; Niu et al. 2023). To this end, our research precisely explains the gradient components blocked by the BN operation. We believe that intuitive findings induced from the analytic explanation is of considerable values for future studies.

Revisiting the BN opertaion. According to (Ioffe and Szegedy 2015), given n samples in a mini-batch, let $\mathbf{X} = [\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \dots, \mathbf{x}^{(n)}] \in \mathbb{R}^{D\times n}$ denote features of these samples in an intermediate layer before a BN operation, where the i-th column $\mathbf{x}^{(i)} \in \mathbb{R}^D$ corresponds to the i-th sample. The BN operation $\mathbf{Z} = [\mathbf{z}^{(1)}, \mathbf{z}^{(2)}, \dots, \mathbf{z}^{(n)}] = \mathit{BN}(\mathbf{X}) \in \mathbb{R}^{D\times n}$ contains the following two phases.

$$\mathbf{Y} = \mathit{diag}(\boldsymbol{\sigma} \circ \boldsymbol{\sigma} + \varepsilon \mathbf{1}_D)^{-\frac{1}{2}}(\mathbf{X} - \boldsymbol{\mu}\mathbf{1}_n^\top) \quad \text{(standardization)} \quad (1)$$

$$\mathbf{Z} = \mathit{diag}(\boldsymbol{\gamma})\mathbf{Y} + \boldsymbol{\beta}\mathbf{1}_n^\top \quad \text{(affine transformation)} \quad (2)$$

where $\mathbf{Y} = [\mathbf{y}^{(1)}, \dots, \mathbf{y}^{(n)}] \in \mathbb{R}^{D\times n}$; $\boldsymbol{\gamma}, \boldsymbol{\beta} \in \mathbb{R}^D$ are used to scale and shift features $\mathbf{Y}$; $\boldsymbol{\mu} = \frac{1}{n}\mathbf{X}\mathbf{1}_n \in \mathbb{R}^D$; $\boldsymbol{\sigma} = [\sqrt{\boldsymbol{\Sigma}_{1,1}}, \dots, \sqrt{\boldsymbol{\Sigma}_{D,D}}]^\top \in \mathbb{R}^D$ represents a vector of the standard deviations corresponding to diagonal elements in the covariance matrix $\boldsymbol{\Sigma} = \frac{1}{n}(\mathbf{X} - \boldsymbol{\mu}\mathbf{1}_n^\top)(\mathbf{X} - \boldsymbol{\mu}\mathbf{1}_n^\top)^\top \in \mathbb{R}^{D\times D}$; $\mathbf{1}_n \in \mathbb{R}^n$ is an all-ones vector; $\circ$ denotes the element-wise product; ε is a tiny positive constant to avoid dividing zero. We ignore the ε term to simplify further proofs. $\mathit{diag}(\cdot)$ transforms a vector to a diagonal matrix.

Blocking of Gradient Components

In this section, we aim to prove that any BN layer in a DNN will block the back-propagation of the first and the second derivatives of the loss function. Let the DNN be trained with any *arbirary* loss function. The training loss on the i-th sample can be represented as a function of the standardized feature $\mathbf{y}^{(i)}$. We use the Taylor series expansion at a fixed point $\tilde{\mathbf{y}} \in \mathbb{R}^D$ (which is an arbitrary vector close to $\mathbf{y}^{(i)}$) to decompose the loss function *w.r.t.* $\mathbf{y}^{(i)}$.

$$\begin{aligned}\mathrm{Loss}(\mathbf{y}^{(i)};\tilde{\mathbf{y}}) = \mathrm{Loss}(\tilde{\mathbf{y}}) + (\mathbf{y}^{(i)} - \tilde{\mathbf{y}})^\top \mathbf{g} \\ + \frac{1}{2!}(\mathbf{y}^{(i)} - \tilde{\mathbf{y}})^\top \mathbf{H}(\mathbf{y}^{(i)} - \tilde{\mathbf{y}}) + \Delta^{(i)},\end{aligned} \quad (3)$$

where $\mathbf{g} \in \mathbb{R}^D$ and $\mathbf{H} \in \mathbb{R}^{D\times D}$ denote the gradient and the Hessian matrix of $\mathrm{Loss}(\mathbf{y}^{(i)})$, respectively; $\Delta^{(i)}$ is a residual term that sums up all terms of higher orders.

In this paper, we consider the following two cases to discuss the BN's effects over all samples in a mini-batch.

Case 1: All Samples in a Mini-Batch Share the Same Analytic Formula of the Loss Function

We first make an attempt to analyze the BN's trend of ignoring specific loss terms in a simple setting that all samples in a mini-batch share the same analytic formula of the loss function, as Equation (3) shows. In fact, many applications belong to this case. For example, when training some invertible generative models (Dinh, Krueger, and Bengio 2014; Dinh, Sohl-Dickstein, and Bengio 2017; Kingma and Dhariwal 2018), all training samples share the same analytic formula of the loss function, $\mathrm{Loss}(\mathbf{y}^{(i)}) = -\log p(\mathit{input}^{(i)})$, which measures the log-likelihood of sample $\mathit{input}^{(i)}$. Similarly, when training the generator of a GAN, all samples in a mini-batch share the same analytic loss function formula, $\mathrm{Loss}(\mathbf{y}^{(i)}) = \log(1 - D(G(\mathit{input}^{(i)})))$, where $D(\cdot)$ and $G(\cdot)$ denote the discriminator and the generator, respectively.

More crucially, we will discuss how to generalize our theorems derived here to more general cases in Subsection "Case2: A More General Case." In general cases, findings in all theorems are still valid, although the problem of blocking loss terms is alleviated.

Let us first introduce the decomposition of loss terms.

Theorem 1 (Proof in Appendix C.) *If we ignore the tiny constant ε (which avoids dividing zero) in Equation (1) by setting $\varepsilon = 0$, then the overall loss of all samples in a mini-batch $\mathit{Loss}^{batch} = \sum_{i=1}^n \mathit{Loss}(\mathbf{y}^{(i)};\tilde{\mathbf{y}})$ can be decomposed as*

$$\mathit{Loss}^{batch} = C + \mathit{Loss}^{grad}(\boldsymbol{g}) + \mathit{Loss}^{Hessian}(\boldsymbol{H}) + \sum\nolimits_i \Delta^{(i)} \quad (4)$$

$$\mathit{Loss}^{Hessian}(\boldsymbol{H}) = \mathit{Loss}^{diag}(\boldsymbol{H}^{diag}) + \mathit{Loss}^{off}(\boldsymbol{H}^{off}) \quad (5)$$

$$\mathit{Loss}^{off}(\boldsymbol{H}^{off}) = L_d^{linear} + L_d^{non} + L_{others} \quad (6)$$

Then, let us introduce the formulation and physical meaning of each above compositional term. Please see Appendix C for detailed discussions of the above terms.

(1) $\mathrm{Loss}^{\mathrm{grad}}(\mathbf{g})$ and $\mathrm{Loss}^{\mathrm{Hessian}}(\mathbf{H})$ denote the first-order term and the second-order term in the Taylor series expansion of $\mathrm{Loss}^{\mathrm{batch}}$ in Equation (3). $\mathrm{Loss}^{\mathrm{grad}}(\mathbf{g}) \triangleq \sum_{i=1}^n (\mathbf{y}^{(i)} - \tilde{\mathbf{y}})^\top \mathbf{g}$ and $\mathrm{Loss}^{\mathrm{Hessian}}(\mathbf{H}) \triangleq \sum_{i=1}^n \frac{1}{2!}(\mathbf{y}^{(i)} - \tilde{\mathbf{y}})^\top \mathbf{H}(\mathbf{y}^{(i)} - \tilde{\mathbf{y}})$.

(2) $C \triangleq n\mathrm{Loss}(\tilde{\mathbf{y}})$ represents the constant w.r.t. $\mathbf{X}$ in the Taylor series expansion in Equation (3).

(3) The second-order term $\mathrm{Loss}^{\mathrm{Hessian}}(\mathbf{H})$ can be further decomposed into two terms, $\mathrm{Loss}^{\mathrm{diag}}(\mathbf{H}^{\mathrm{diag}})$ and $\mathrm{Loss}^{\mathrm{off}}(\mathbf{H}^{\mathrm{off}})$.

(3.1) The term of $\mathrm{Loss}^{\mathrm{diag}}(\mathbf{H}^{\mathrm{diag}}) = \sum_{i=1}^n \frac{1}{2!}(\mathbf{y}^{(i)} - \tilde{\mathbf{y}})^\top \mathbf{H}^{\mathrm{diag}}(\mathbf{y}^{(i)} - \tilde{\mathbf{y}})$ quantifies the effects made by diagonal elements in the Hessian matrix, where $\mathbf{H}^{\mathrm{diag}}$ denotes the matrix only containing diagonal elements in $\mathbf{H}$.

(3.2) The term of $\text{Loss}^{\text{off}}(\mathbf{H}^{\text{off}}) = \sum_{i=1}^{n} \frac{1}{2!}(\mathbf{y}^{(i)} - \tilde{\mathbf{y}})^\top \mathbf{H}^{\text{off}}(\mathbf{y}^{(i)} - \tilde{\mathbf{y}})$ quantifies the effects made by off-diagonal elements in the Hessian matrix, where $\mathbf{H}^{\text{off}} = \mathbf{H} - \mathbf{H}^{\text{diag}}$ denotes the matrix exclusively containing off-diagonal elements in $\mathbf{H}$. Furthermore, $\text{Loss}^{\text{off}}(\mathbf{H}^{\text{off}})$ can be decomposed into three terms, *i.e.*, L_d^{linear}, L_d^{non} and L_{others}.

(3.2.1) The term of $L_d^{\text{linear}} = \mathbf{H}^{\text{off}}_{d,:}\mathbf{Y}^{\text{linear}}\mathbf{y}_d$ measures the effects corresponding to feature components within $\mathbf{Y}$ that are linearly correlated with the d-th row of $\mathbf{Y}$, *i.e.*, $\mathbf{y}_d = [y_{d,1}, y_{d,2}, \ldots, y_{d,n}]^\top \in \mathbb{R}^n$. Here $\mathbf{Y}^{\text{linear}} = [\mathbf{o}_d^\top \mathbf{y}_1, \mathbf{o}_d^\top \mathbf{y}_2, \cdots, \mathbf{o}_d^\top \mathbf{y}_D]^\top \mathbf{o}_d^\top$ is decomposed from $\mathbf{Y}$. $\mathbf{o}_d = \mathbf{y}_d/\|\mathbf{y}_d\|$ denotes the unit vector in the direction of $\mathbf{y}_d$. $\mathbf{H}^{\text{off}}_{d,:} \in \mathbb{R}^D$ denotes the d-th row of $\mathbf{H}^{\text{off}}$.

(3.2.2) The term of $L_d^{\text{non}} = \mathbf{H}^{\text{off}}_{d,:}(\mathbf{Y}^{\text{non}} - \tilde{\mathbf{y}}\mathbf{1}_n^\top)\mathbf{y}_d$ measures non-correlated effects. $\mathbf{Y}^{\text{non}} = \mathbf{Y} - \mathbf{Y}^{\text{linear}}$ is disentangled from $\mathbf{Y}$, when we remove components linearly-correlated to $\mathbf{y}_d$, *i.e.*, $\mathbf{Y}^{\text{linear}}$, from $\mathbf{Y}$.

Results in Table 6 verify that the L_d^{linear} term has stronger gradients *w.r.t.* $\mathbf{y}_d$, *i.e.*, $\partial L_d^{\text{linear}}/\partial \mathbf{y}_d$, than the L_d^{non} term.

(3.2.3) The term of $L_{\text{others}} = \text{Loss}^{\text{off}}(\mathbf{H}^{\text{off}}) - L_d^{\text{linear}} - L_d^{\text{non}}$ corresponds to all other loss terms of $\text{Loss}^{\text{off}}(\mathbf{H}^{\text{off}})$, when we remove all terms (L_d^{linear} and L_d^{non}) depending on $\mathbf{y}_d$.

Theorem 2 (**Blocking of gradients** proved in Appendix D.1.)

$$\frac{\partial Loss^{grad}(\boldsymbol{g})}{\partial \boldsymbol{X}} = \boldsymbol{0}, \qquad \frac{\partial C}{\partial \boldsymbol{X}} = \boldsymbol{0}, \\ \frac{\partial Loss^{diag}(\boldsymbol{H}^{diag})}{\partial \boldsymbol{X}} = \boldsymbol{0}, \qquad \forall d,\ \frac{\partial L_d^{linear}}{\partial \boldsymbol{x}_d} = \boldsymbol{0}. \tag{7}$$

Theorem 2 shows that $\text{Loss}^{\text{grad}}(\mathbf{g})$ and $\text{Loss}^{\text{diag}}(\mathbf{H}^{\text{diag}})$ have no gradients on input features $\mathbf{X}$. This can simply lead to a corollary that *all network weights* $\boldsymbol{W}$ *before the BN layer will not be trained by terms of* $Loss^{grad}(\boldsymbol{g})$ *and* $Loss^{diag}(\boldsymbol{H}^{diag})$*, as follows.* Please see Appendix D.3 for more discussions.

$$\frac{\partial \text{Loss}^{\text{grad}}(\mathbf{g})}{\partial \mathbf{W}} = \frac{\partial \text{Loss}^{\text{diag}}(\mathbf{H}^{\text{diag}})}{\partial \mathbf{W}} = \mathbf{0} \tag{8}$$

Besides, because of $\frac{\partial L_d^{\text{linear}}}{\partial \mathbf{x}_d} = \mathbf{0}$, the impacts of off-diagonal elements in $\widehat{\mathbf{H}}$, *i.e.*, $\text{Loss}^{\text{off}}(\mathbf{H}^{\text{off}})$, on weights $\mathbf{W}$ before the BN layer are reduced. In Appendix D.4, we have further proved that only the batch centering operation in BN can already block the propagation of gradient $\frac{\partial \text{Loss}^{\text{grad}}(\mathbf{g})}{\partial \mathbf{X}}$.

Corollary 1 (Proof in Appendix D.2.) *Based on Theorem 2, in the training phase of a neural network, we have*

$$\frac{\partial^2 Loss^{batch}}{\partial \boldsymbol{X}\partial \boldsymbol{g}} = \boldsymbol{0}, \quad \frac{\partial^2 Loss^{batch}}{\partial \boldsymbol{X}\partial \boldsymbol{H}^{diag}} = \boldsymbol{0}, \tag{9}$$

and $\forall d$, $\frac{\partial^2 Loss^{batch}}{\partial \boldsymbol{x}_d \partial \boldsymbol{H}^{off}_{d,:}} = \frac{\partial^2 L_d^{linear}}{\partial \boldsymbol{x}_d \partial \boldsymbol{H}^{off}_{d,:}} + \frac{\partial^2 L_d^{non}}{\partial \boldsymbol{x}_d \partial \boldsymbol{H}^{off}_{d,:}}$*, where*

$$\frac{\partial^2 L_d^{linear}}{\partial \boldsymbol{x}_d \partial \boldsymbol{H}^{off}_{d,:}} = \frac{\partial^2}{\partial \boldsymbol{x}_d \partial \boldsymbol{H}^{off}_{d,:}}(A[\boldsymbol{o}_d^\top \boldsymbol{y}_1, \boldsymbol{o}_d^\top \boldsymbol{y}_2, \cdots, \boldsymbol{o}_d^\top \boldsymbol{y}_D]^\top) = \boldsymbol{0}, \tag{10}$$

s.t. $A = \|\boldsymbol{y}_d\| \cdot \boldsymbol{H}^{off}_{d,:}$*. In contrast, in the testing phase,* $\frac{\partial^2 Loss^{batch}}{\partial \boldsymbol{X}\partial \boldsymbol{g}} \neq \boldsymbol{0}$, $\frac{\partial^2 Loss^{batch}}{\partial \boldsymbol{X}\partial \boldsymbol{H}^{diag}} \neq \boldsymbol{0}$*, and* $\frac{\partial^2 L_d^{linear}}{\partial \boldsymbol{x}_d \partial \boldsymbol{H}^{off}_{d,:}} \neq \boldsymbol{0}$*.*

(Conclusion 1) Blocking of gradient components in Case 1. Given a DNN, $\frac{\partial^2 \text{Loss}^{\text{batch}}}{\partial \mathbf{X}\partial \mathbf{g}^\top} = \mathbf{0}$, $\frac{\partial^2 \text{Loss}^{\text{batch}}}{\partial \mathbf{X}\partial \mathbf{H}^{\text{diag}}} = \mathbf{0}$, and $\forall d$, $\frac{\partial^2 L_d^{\text{linear}}}{\partial \mathbf{x}_d \partial \mathbf{H}^{\text{off}}_{d,:}} = \mathbf{0}$ in Corollary 1 show that the BN operation will block the back-propagation of the following three types of influence, *i.e.*, (1) the influence of the first derivatives in $\mathbf{g}$, (2) the influence of the diagonal elements in the Hessian matrix $\mathbf{H}$, and (3) a considerable ratio of the influence of off-diagonal elements in the Hessian matrix $\mathbf{H}$.

(Conclusion 2) Reason for blocking. Corollary 1 tells us that the BN's blocking of the above influence only exists in the training phase, but it does not exist in the testing phase. In Appendix H, we have further proved that it is the derivatives of $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$ in the standardization phase that eliminate the influence of $\text{Loss}^{\text{grad}}(\mathbf{g})$, $\text{Loss}^{\text{diag}}(\mathbf{H}^{\text{diag}})$, and L_d^{linear}.

Furthermore, although $\boldsymbol{\gamma}$ in the affine transformation phase can **alleviate** the blocking of $\text{Loss}^{\text{grad}}(\mathbf{g})$ by encoding the gradient $\mathbf{g}$ of the first-order term of the loss, $\text{Loss}^{\text{diag}}(\mathbf{H}^{\text{diag}})$ and L_d^{linear} still cannot influence parameters in all layers before the BN operation. Beyond this, an analytic explanation for detailed information-processing behaviors of a BN layer may shed new lights into future theoretical research.

Case 2: A More General Case

Then, let us extend our theory to a more general case that considers following two issues. First, different samples in a mini-batch may have different analytic formulas of loss functions. Second, the gradient $\mathbf{g}$ and the Hessian matrix $\mathbf{H}$ are unstable, because the change of gating states over different samples in gating layers (such as the ReLU layer) is discontinuous and unpredictable.

In this subsection, we show that conclusions in Case 1 can still be adapted to the general case, although the ratio of blocked gradients to all gradients is reduced in the general case. In certain tasks, the block of gradients from specific loss terms may not lead to serious damages to DNNs' performance, but we believe that it is meaningful to provide analytic insights into the BN's distinctive behavior of information processing.

In general cases, the overall loss of a mini-batch can still be written in the form of the Taylor series expansion, just like in Equation (3), when we compute the average gradient $\widehat{\mathbf{g}} = \frac{1}{n}\sum_{i=1}^{n} \mathbf{g}^{(i)}$ and estimate an equivalent Hessian matrix $\widehat{\mathbf{H}} = \arg\min_{\widehat{\mathbf{H}}} \sum_{i=1}^{n} \|\mathbf{g}^{(i)} - \widehat{\mathbf{g}} - \widehat{\mathbf{H}}(\mathbf{y}^{(i)} - \tilde{\mathbf{y}})\|_2^2$[3]. Both $\widehat{\mathbf{g}}$ and $\widehat{\mathbf{H}}$ are computed for all samples in a mini-batch to approximately mimic the loss landscape over these samples. Later, we will examine the quality of using $\widehat{\mathbf{g}}$ and $\widehat{\mathbf{H}}$ to approximate $\text{Loss}(\mathbf{y}^{(i)}; \tilde{\mathbf{y}})$. Thus, the loss function for each i-th sample, $1 \leq i \leq n$, can be written as

$$\text{Loss}(\mathbf{y}^{(i)}; \tilde{\mathbf{y}}) = \text{Loss}(\tilde{\mathbf{y}}) + (\mathbf{y}^{(i)} - \tilde{\mathbf{y}})^\top \widehat{\mathbf{g}} \\ + \frac{1}{2!}(\mathbf{y}^{(i)} - \tilde{\mathbf{y}})^\top \widehat{\mathbf{H}}(\mathbf{y}^{(i)} - \tilde{\mathbf{y}}) + \widehat{\Delta}^{(i)}, \tag{11}$$

where $\widehat{\Delta}^{(i)}$ denotes the distinctive term of the i-th sample, which sums up all errors caused by $\widehat{\mathbf{g}}$ and $\widehat{\mathbf{H}}$, but $\widehat{\Delta}^{(i)}$ is not limited to terms of greater-than-two orders $\Delta^{(i)}$. Thus, the

[3]Please see Appendix E.2 for the computational details of $\widehat{\mathbf{H}}$.

	VGG-16	VGG-11	AlexNet
p^{error}	0.248 ± 0.015	0.240 ± 0.014	0.226 ± 0.010

Table 1: The proportion of loss values that cannot be approximated using $\widehat{\mathbf{g}}$ and $\widehat{\mathbf{H}}$.

	Δgrad_1	Δgrad_2	Δgrad_3
w/ BN[4]	**1.79e-7±2.1e-7**	**2.85e-7±3.9e-8**	**0.51±0.33**
w/o BN[6]	1.04±0.51	1.07±0.57	1.05±0.44

Table 2: Verifying that the BN blocked the propagation of the 1st and the 2nd derivatives in an MLP.

overall loss of all samples in a mini-batch can be re-written in the same form of Equation (4), as follows.

$$\text{Loss}^{\text{batch}} = C + \text{Loss}^{\text{grad}}(\widehat{\mathbf{g}}) + \text{Loss}^{\text{Hessian}}(\widehat{\mathbf{H}}) + \sum\nolimits_i \widehat{\Delta}^{(i)}. \quad (12)$$

In this way, both Case 1 and Case 2 can be represented using the same Taylor series expansion form. Thus, Theorems 1 and 2, and Corollary 1 derived based on the Taylor series expansion in Equation (4) are also valid for Case 2.

(Conclusion 3) Blocking of gradient components in Case 2. $\text{Loss}^{\text{grad}}(\widehat{\mathbf{g}})$ and $\text{Loss}^{\text{diag}}(\widehat{\mathbf{H}}^{\text{diag}})$ have no gradients on features before the BN operation, and $\text{Loss}^{\text{off}}(\widehat{\mathbf{H}}^{\text{off}})$ does not have strong effects on features before the BN operation. Please see Appendix E.1 for discussions.

• *Examining the quality of using $\widehat{\boldsymbol{g}}$ and $\widehat{\boldsymbol{H}}$ to approximately compute Loss$(\mathbf{y}^{(i)};\tilde{\mathbf{y}})$.* Equation (12) shows that losses $\text{Loss}(\mathbf{y}^{(i)};\tilde{\mathbf{y}})$ of all samples in a mini-batch can be approximated using the same analytic formula based on $\widehat{\mathbf{g}}$ and $\widehat{\mathbf{H}}$. We have conducted experiments to examine the quality of such an approximation. Specifically, we used the metric $p^{\text{error}} = \sum_{i=1}^{n}|\Delta^{(i)}|/(\sum_{i=1}^{n}(|\text{Loss}(\tilde{\mathbf{y}})|+|\text{Loss}^{\text{grad}}(\widehat{\mathbf{g}})|+|\text{Loss}^{\text{diag}}(\widehat{\mathbf{H}}^{\text{diag}})|+|\text{Loss}^{\text{off}}(\widehat{\mathbf{H}}^{\text{off}})| + |\Delta^{(i)}|))$ to measure the proportion of loss values that cannot be approximated using $\widehat{\mathbf{g}}$ and $\widehat{\mathbf{H}}$. To this end, we conducted experiments on AlexNet (Krizhevsky, Sutskever, and Hinton 2012), VGG-11/16 (Simonyan and Zisserman 2015), where each network was added a BN layer before the top FC layer. These DNNs were trained on the CIFAR-10 dataset (Krizhevsky, Hinton et al. 2009) for image classification. Note that we generally used cross entropy loss for image classification task. Table 1 reports the proportion p^{error} averaged over all mini-batches. Results show that $p^{\text{error}} < 0.25$, which means that less than 1/4 loss values could not be represented by $\widehat{\mathbf{g}}$ and $\widehat{\mathbf{H}}$.

• *Limitation & extended questions. How about when the feature $\mathbf{y}^{(i)}$ is not so close to the fixed point $\tilde{\mathbf{y}}$?* In other words, *do we need a very tiny residual term $\Delta^{(i)}$ in Equation (3) or $\widehat{\Delta}^{(i)}$ in Equation (12) in the Taylor series expansion?* As a common trick, we can set $\tilde{\mathbf{y}}$ to the mean value $\mathbb{E}_i[\mathbf{y}^{(i)}]$ to boost the fitness of the Taylor series expansion of losses on most samples, thereby reducing the residual term. In particular, we have proved in Appendix F that the third and higher-order derivatives in the residual term of the sigmoid function have small strengths when the classification is confident.

More crucially, even though we cannot let the unexplainable residual term $\Delta^{(i)}$ or $\widehat{\Delta}^{(i)}$ be fully ignorable in some tasks, an analytic explanation for the blocking of loss components of $\text{Loss}^{\text{grad}}(\widehat{\mathbf{g}})$ and $\text{Loss}^{\text{Hessian}}(\widehat{\mathbf{H}})$ is still of considerable values for future studies. In fact, Table 4 shows that **the $\text{Loss}^{\text{grad}}(\widehat{\mathbf{g}})$ and $\text{Loss}^{\text{diag}}(\widehat{\mathbf{H}}^{\text{diag}})$ terms have made 37.8%–79.5% influence *w.r.t.* the loss value in a general task of image classification.**

Note that though the BN operation blocks the propagation of many loss terms, **the non-negligible residual loss term ensures that the DNN can still learn from the data.** Nevertheless, experimental results in Figures 1 and 2 show that given an negligible residual term, the blocking problem may fully dominate the learning process, thereby damaging feature representations of the DNN. Therefore, to avoid the blocking problem, we recommend to ensure the diversity of samples in a mini-batch. For example, in federated learning (Konečnỳ, McMahan, and Ramage 2015), it is necessary to avoid each node being assigned to learn from samples of a single category.

Experiments

Experimental Verification of Theorem 2

• **1. Verifying the blocking of g and $\mathbf{H}^{\text{diag}}$.** We did experiments on a synthetic dataset and the CIFAR-10 dataset, respectively, to verify that $\text{Loss}^{\text{grad}}(\mathbf{g})$ and $\text{Loss}^{\text{diag}}(\mathbf{H}^{\text{diag}})$ had no gradients on features before the BN.

Experiment 1-1 based on the synthesized loss functions. In this experiment, we directly designed the loss function as a polynomial with derivatives of different orders, in order to evaluate the effects of the BN operation on derivatives of different orders. To this end, we synthesized a group of five loss functions with derivatives of different orders, $\forall 1 \le k \le 4$, $\text{Loss}_k(\mathbf{y}|\boldsymbol{\lambda}) = \sum_{i=1}^{n}\sum_{k'=k}^{4}\lambda_{k'}(y^{(i)})^{k'}$, by sampling parameters $\boldsymbol{\lambda} = [\lambda_1,\ldots,\lambda_4]^\top \sim N(\boldsymbol{\mu}=\mathbf{0}, \Sigma = I_{4\times4})$ from a Gaussian distribution. Thus, we constructed a dataset containing 1000 groups of loss functions by sampling $\boldsymbol{\lambda}$ for 1000 times. We used each group of four losses *w.r.t.* a specific $\boldsymbol{\lambda}$ to train four 5-layer MLPs. For each MLP, we put a BN operation on its top. Each layer in the MLP contained M =100 neurons, except for the last layer containing a single neuron. Similarly, we also trained five MLPs without BN layers as baseline models for comparison. The input of each MLP was a noisy vector sampled from a Gaussian distribution $N(\boldsymbol{\mu}=\mathbf{0}, \Sigma = I_{M\times M})$. Here, M =100.

We measured $\Delta\text{grad}_q = \mathbb{E}_{\boldsymbol{\lambda}}\big[\|\frac{\partial \text{Loss}_q(\mathbf{y}|\boldsymbol{\lambda})}{\partial \mathbf{x}} - \frac{\partial \text{Loss}_{q+1}(\mathbf{y}|\boldsymbol{\lambda})}{\partial \mathbf{x}}\|/\|\frac{\partial \text{Loss}_q(\mathbf{y}|\boldsymbol{\lambda})}{\partial \mathbf{x}}\|\big]$, $q = 1,2,3$, to examine whether the q-th order term of the loss could pass its influence through the BN operation. Table 2 shows that for MLPs with BN layers, $\Delta\text{grad}_1 \approx 0$, and $\Delta\text{grad}_2 \approx 0$ (see footnote [4]), which proved that the zeroth-order, the first-order, and the second-order terms[5] of the loss could not pass their influence through the

[4]The small deviation was caused by the accumulation of tiny systematic computational errors in a DNN, *e.g.*, 0.1 + 0.2 = 0.30000000000000004 in Python caused by the 32-bit floating-point operations.

[5]Here the Hessian matrix reduced to scalar second derivative.

Layer in VGG-11		$\Delta\text{grad}^{\text{first}}$	$\Delta\text{grad}^{\text{second,diag}}$	$\Delta\text{grad}^{\text{second,off}}$
The 3rd top FC	w/ BN[4]	**1.37e-8±2.72e-8**	**2.72e-8±5.41e-8**	**1019.49±116.45**
	w/o BN[6]	3.58±0.36	2.22±0.26	100.70±11.78
conv5-2	w/ BN[4]	**6.97e-16±1.18e-16**	**1.23e-9±2.47e-9**	**323.48±33.79**
	w/o BN[6]	1.40±0.14	1.64±0.19	34.41±3.88
conv5-1	w/ BN[4]	**6.88e-16±1.71e-16**	**1.35e-9±2.93e-9**	**387.46±300.19**
	w/o BN[6]	1.33±0.14	1.81±0.20	39.25±4.26

Table 3: Verifying that all elements in $\mathbf{g}$ and diagonal elements in $\mathbf{H}$ could not pass their influence through the BN layer.

	VGG-16	VGG-11	AlexNet
p^{blocked}	0.795 ± 0.397	0.662 ± 0.303	0.378 ± 0.200

Table 4: Measuring the proportion of the blocked two loss terms $\text{Loss}^{\text{grad}}(\widehat{\mathbf{g}})$ and $\text{Loss}^{\text{diag}}(\widehat{\mathbf{H}}^{\text{diag}})$.

BN operation. Besides, $\Delta\text{grad}_3 = 0.51 \pm 0.33$ indicated in the Taylor series expansion that the third-order term successfully passed its influence through the BN operation. In comparison, MLPs without BN layers had relatively large $\forall q, \Delta\text{grad}_q$ values[6].

Experiment 1-2 based on the CIFAR-10 dataset for image classification. We constructed three VGG-11 networks by adding a BN layer (in Equations (1) and (2)) after the third top FC, conv5-2, and conv5-1, respectively. Each network was trained on the CIFAR-10 dataset with a classification loss ($\text{Loss}^* = \sum_{i=1}^n \text{Loss}^{\text{cls}}(\mathbf{y}^{(i)})$). Then, we tested the blocking of the gradient $\mathbf{g}$ and diagonal elements in the Hessian matrix $\mathbf{H}$ of each network.

Specifically, in order to evaluate the exact effect of each term in the Taylor series expansion of the loss, we manually added noisy first derivatives and noisy second derivatives to construct three additional losses, including $\text{Loss}_2 = \sum_{i=1}^n (\text{Loss}^{\text{cls}}(\mathbf{y}^{(i)}) + \boldsymbol{\epsilon}^\top \mathbf{y}^{(i)})$, $\text{Loss}_3 = \sum_{i=1}^n (\text{Loss}^{\text{cls}}(\mathbf{y}^{(i)}) + (\mathbf{y}^{(i)} - \tilde{\mathbf{y}})^\top \textit{diag}(\boldsymbol{\epsilon})(\mathbf{y}^{(i)} - \tilde{\mathbf{y}}))$, and $\text{Loss}_4 = \sum_{i=1}^n (\text{Loss}^{\text{cls}}(\mathbf{y}^{(i)}) + (\mathbf{y}^{(i)} - \tilde{\mathbf{y}})^\top \mathbf{E}^{\text{off}} (\mathbf{y}^{(i)} - \tilde{\mathbf{y}}))$. We set $\tilde{\mathbf{y}} = \frac{1}{n}\sum_{i=1}^n \mathbf{y}^{(i)}$ in this experiment. Each element in $\boldsymbol{\epsilon} \in \mathbb{R}^D$ and $\mathbf{E}^{\text{off}} \in \mathbb{R}^{D\times D}$ was sampled from a Gaussian distribution $N(\mu = 0, \sigma^2 = 0.1^2)$. Diagonal elements in $\mathbf{E}^{\text{off}}$ were set to 0.

In this way, if $\frac{\partial \text{Loss}^*}{\partial \mathbf{X}} = \frac{\partial \text{Loss}_2}{\partial \mathbf{X}} = \frac{\partial \text{Loss}_3}{\partial \mathbf{X}}$, then it proved that the first derivatives in $\mathbf{g}$ and diagonal elements in $\mathbf{H}$ could not pass their influence through the BN operation. Therefore, we used metrics $\Delta\text{grad}^{\text{first}} = \|\frac{\partial \text{Loss}^*}{\partial \mathbf{X}} - \frac{\partial \text{Loss}_2}{\partial \mathbf{X}}\|_F / \|\frac{\partial \text{Loss}^*}{\partial \mathbf{X}}\|_F$, $\Delta\text{grad}^{\text{second,diag}} = \|\frac{\partial \text{Loss}^*}{\partial \mathbf{X}} - \frac{\partial \text{Loss}_3}{\partial \mathbf{X}}\|_F / \|\frac{\partial \text{Loss}^*}{\partial \mathbf{X}}\|_F$, and $\Delta\text{grad}^{\text{second,off}} = \|\frac{\partial \text{Loss}^*}{\partial \mathbf{X}} - \frac{\partial \text{Loss}_4}{\partial \mathbf{X}}\|_F / \|\frac{\partial \text{Loss}^*}{\partial \mathbf{X}}\|_F$ to measure the influence of the first derivatives in $\mathbf{g}$, diagonal elements in $\mathbf{H}$, and off-diagonal elements in $\mathbf{H}$ on the gradient $\frac{\partial \text{Loss}^*}{\partial \mathbf{X}}$. Given a DNN with BN layers, we trained another DNN of the same architecture but without BN layers as the baseline for comparison.

According to Table 3, for DNNs with BN layers, $\Delta\text{grad}^{\text{first}} \approx 0$ and $\Delta\text{grad}^{\text{second,diag}} \approx 0$ (see footnote [4]) proved that elements in $\mathbf{g}$ and diagonal elements in $\mathbf{H}$ could not pass their influence through the BN operation. Besides, the large value of $\Delta\text{grad}^{\text{second,off}}$ indicated that off-diagonal elements in $\mathbf{H}$ could pass their influence through the BN operation. In comparison, baseline DNNs without BN layers had larger $\Delta\text{grad}^{\text{first}}$, $\Delta\text{grad}^{\text{second,diag}}$, and $\Delta\text{grad}^{\text{second,off}}$ values. In addition, we have reported results when each element in $\boldsymbol{\epsilon}$ and $\mathbf{E}^{\text{off}}$ was sampled from Gaussian distributions with large μ and large σ^2 in Appendix I.5, which yielded similar conclusions.

• **2. Verifying that L_d^{linear} had no gradients on features before the BN layer.** To this end, we directly computed the norm of the gradient $\|\partial L_d^{\text{linear}} / \partial \mathbf{x}_d\|$, where $\mathbf{H}_{d,:}^{\text{off}}$ was computed by following Cohen et al. (2020). To comprehensively test the BN on different layers, we revised the AlexNet by adding five additional FC layers before the top FC layer, and revised the LeNet (LeCun et al. 1989) by adding seven additional FC layers before the top FC layer. Please see Appendix I.1 about the revised architectures. For each DNN, we added a BN layer before the 1st, 2nd, and 3rd top FC layers, respectively, to construct *AlexNet-1/2/3 and LeNet-1/2/3*. Table 5 reports the statistics of $\|\partial L_d^{\text{linear}} / \partial \mathbf{x}_d\|$ over different feature dimensions (d) and different mini-batches on the MNIST dataset (LeCun et al. 1998). Results show that $\|\partial L_d^{\text{linear}} / \partial \mathbf{x}_d\| \approx 0$ in the above six DNNs, which proved that L_d^{linear} had no gradients on features before the BN layer. In comparison, when we removed the BN layer from the DNN, we found that the metric[6] $\|\partial L_d^{\text{linear}} / \partial \mathbf{x}_d\|$ became much larger.

• **3. Examining that interactions between linearly-correlated feature components took a main part of the loss L_d^{off}.** In Theorem 1, $L_d^{\text{off}} = L_d^{\text{linear}} + L_d^{\text{non}}$. Note that we have proved that the L_d^{linear} term has no effects on $\mathbf{x}_d$, so we aimed to further show that the L_d^{linear} term had considerable gradients *w.r.t.* $\mathbf{y}_d$. If so, it means that the BN blocked significant influence of $\partial L_d^{\text{linear}} / \partial \mathbf{y}_d$ in back-propagation. To this end, we computed $\frac{\|\partial L_d^{\text{linear}} / \partial \mathbf{y}_d\|}{\|\partial L_d^{\text{off}} / \partial \mathbf{y}_d\|}$ to measure the relative significance of the compositional influence of $\partial L_d^{\text{linear}} / \partial \mathbf{y}_d$ to $\partial L_d^{\text{off}} / \partial \mathbf{y}_d$. Similarly, we computed $\frac{\|\partial L_d^{\text{non}} / \partial \mathbf{y}_d\|}{\|\partial L_d^{\text{off}} / \partial \mathbf{y}_d\|}$ measured the relative significance of $\partial L_d^{\text{non}} / \partial \mathbf{y}_d$ to $\partial L_d^{\text{off}} / \partial \mathbf{y}_d$. We conducted experiments on *AlexNet-1/2/3* and *LeNet-1/2/3*. Table 6 reports the statistics of $\frac{\|\partial L_d^{\text{linear}} / \partial \mathbf{y}_d\|}{\|\partial L_d^{\text{off}} / \partial \mathbf{y}_d\|}$ and $\frac{\|\partial L_d^{\text{non}} / \partial \mathbf{y}_d\|}{\|\partial L_d^{\text{off}} / \partial \mathbf{y}_d\|}$ over different feature dimensions and different mini-batches. $\partial L_d^{\text{linear}} / \partial \mathbf{y}_d$ had considerable impacts on $\partial L_d^{\text{off}} / \partial \mathbf{y}_d$, which demonstrated that the BN blocked significant influence of L_d^{off}.

[6]The metric is computed using $\mathbf{x}$ in the same layer, but the BN operation after $\mathbf{x}$ was removed.

	AlexNet-1	AlexNet-2	AlexNet-3	LeNet-1	LeNet-2	LeNet-3
w/ BN[4]	**7.1e-12±8.6e-12**	**2.9e-11±5.1e-11**	**3.9e-11±7.5e-11**	**1.0e-10±2.0e-9**	**5.2e-12±8.8e-12**	**1.5e-11±2.3e-11**
w/o BN[6]	4.8e-3±4.6e-3	3.6e-3±3.4e-3	6.6e-5±4.8e-5	7.0e-3±5.5e-3	2.7e-3±1.4e-3	1.8e-4±1.7e-4

Table 5: Verifying that L_d^{linear} passed almost zero[4] gradients $\|\partial L_d^{\text{linear}}/\partial \mathbf{x}_d\|$ through the BN layer.

	AlexNet-1	AlexNet-2	AlexNet-3	LeNet-1	LeNet-2	LeNet-3
$\frac{\|\partial L_d^{\text{linear}}/\partial \mathbf{y}_d\|}{\|\partial L_d/\partial \mathbf{y}_d\|}$	**0.84±0.19**	**0.73±0.25**	**0.85±0.22**	**0.67±0.29**	**0.68±0.29**	**0.77±0.27**
$\frac{\|\partial L_d^{\text{non}}/\partial \mathbf{y}_d\|}{\|\partial L_d/\partial \mathbf{y}_d\|}$	0.47±0.22	0.56±0.29	0.40±0.28	0.63±0.27	0.61±0.29	0.51±0.27

Table 6: Verifying that $\partial L_d^{\text{linear}}/\partial \mathbf{y}_d$ had greater impacts on the overall gradient $\partial L_d^{\text{off}}/\partial \mathbf{y}_d$ than $\partial L_d^{\text{non}}/\partial \mathbf{y}_d$.

Evaluating the Explanation in General Cases

In this experiment, we examined whether the blocked loss terms $\text{Loss}^{\text{grad}}(\widehat{\mathbf{g}})$ and $\text{Loss}^{\text{diag}}(\widehat{\mathbf{H}}^{\text{diag}})$ were non-ignorable w.r.t. the entire loss in **general** cases, so as to evaluate the effectiveness of our theory. To this end, we measured the proportion of $\text{Loss}^{\text{grad}}(\widehat{\mathbf{g}})$ and $\text{Loss}^{\text{diag}}(\widehat{\mathbf{H}}^{\text{diag}})$ in the overall loss in the general case. We conducted experiments on AlexNet, VGG-11/16, each network was added a BN layer before the top FC layer. These DNNs were trained on the CIFAR-10 dataset for image classification. We used the metric $p^{\text{blocked}} = (\sum_{i=1}^n |(\mathbf{y}^{(i)} - \tilde{\mathbf{y}})^\top \widehat{\mathbf{g}}| + \sum_{i=1}^n \frac{1}{2!}|(\mathbf{y}^{(i)} - \tilde{\mathbf{y}})^\top \widehat{\mathbf{H}}^{\text{diag}}(\mathbf{y}^{(i)} - \tilde{\mathbf{y}})|)/\sum_{i=1}^n \text{Loss}(\mathbf{y}^{(i)}; \tilde{\mathbf{y}})$ to measure the proportion of the blocked above two loss terms. In Table 4, p^{blocked} values for the VGG-16, VGG-11, and AlexNet were 79.5%, 66.2%, and 37.8%, respectively. It meant that the blocked loss signals were as significant as 79.5% of the loss value. This proved that the blocked loss terms $\text{Loss}^{\text{grad}}(\widehat{\mathbf{g}})$ and $\text{Loss}^{\text{diag}}(\widehat{\mathbf{H}}^{\text{diag}})$ were still made non-ignorable impacts on model training in general cases. In addition, we also reported the proportion of compositional gradients $\partial \text{Loss}^{\text{grad}}(\widehat{\mathbf{g}})/\partial \mathbf{y}^{(i)}$ and $\partial \text{Loss}^{\text{diag}}(\widehat{\mathbf{H}}^{\text{diag}})/\partial \mathbf{y}^{(i)}$ over the entire gradient $\partial \text{Loss}(\mathbf{y}^{(i)})/\partial \mathbf{y}^{(i)}$ in Appendix I.7. It showed that gradients of the blocked loss terms $\text{Loss}^{\text{grad}}(\widehat{\mathbf{g}})$ and $\text{Loss}^{\text{diag}}(\widehat{\mathbf{H}}^{\text{diag}})$ were non-ignorable *w.r.t.* gradients of the entire loss in *general* cases.

Analyzing the Features Learned by Neural Networks with BN Operations

In this subsection, we tried to use our theory to conceptually explain the difference of performance between DNNs with BN layers and DNNs without BN layers.

Experiment 1: We measured the BN's effects on feature representations of the invertible generative model RealNVP (Dinh, Sohl-Dickstein, and Bengio 2017). The vanilla RealNVP had BN operations, thus being termed *RealNVP-BN*. Besides, we constructed another RealNVP by replacing all BN layers with LN layers, namely *RealNVP-LN*, for comparison[7]. All RealNVPs were trained on the MNIST dataset.

We tested whether a RealNVP model could successfully distinguish real images and fake images. This was the key capacity for a generative model. Specifically, a well-trained RealNVP was supposed to predict high log-likelihood on real images I_1^{real} and I_2^{real}, and yield low log-likelihood on fake images. We generated fake images by linear interpolation, $I_\alpha^{\text{inter}} = \alpha I_1^{\text{real}} + (1-\alpha) I_2^{\text{real}}, \alpha \in (0,1)$. To sharpen the difference caused by the BN operation, we learned different groups of RealNVP-BN/LN, each being trained to generate a specific pair of categories, as Figure 1 shows. Then, for each RealNVP, we computed the average log-likelihood of interpolated images, *i.e.*, $\frac{1}{M}\mathbb{E}_{I_1^{\text{real}}, I_2^{\text{real}}}[\log p(I_\alpha^{\text{inter}})]$, where M denotes the number of pixels for normalization. We computed different log-likelihood values by applying different interpolation rates α. Figure 1 shows that RealNVP-LN usually assigned much higher log-likelihood with real images (*i.e.*, images at the points of $\alpha = 0$ and $\alpha = 1$) than interpolated images. In comparison, RealNVP-BN could not significantly distinguish real images and interpolated images. It may be because the average derivatives shared by all samples in a mini-batch usually reflected the most common signal in real images. Thus, the blocking of the first derivatives prevented the RealNVP from modeling common features of real digits. Besides, we have reported results on more RealNVP models with various revised architectures in Appendix I.3, and results of using more than two real images to generate a fake image, which also yielded similar conclusions.

Experiment 2: We measured the BN's effects on feature representations for classification by comparing four groups of DNNs trained on the CIFAR-10 dataset. The first group of DNNs did not contain any normalization operations, namely *DNN-ori*. The second group of DNNs were obtained by adding BN operations to the DNN-ori, termed *DNN-BN*. The third group of DNNs were obtained by replacing all BN operations in the DNN-BN with LN operations, namely *DNN-LN*. These three groups of DNNs were trained when each mini-batch only contained samples in a specific category. The fourth group of DNNs had the same architecture as the DNN-ori, but they were trained when all samples in a mini-batch had different labels, termed *DNN-ori-base*. We selected VGG-16, ResNet-34 (He et al. 2016), DenseNet-169 (Huang et al. 2017) for classification. For VGG-16, we constructed and learned all four groups of DNNs. Specifically, we added one single BN (or LN) layer before the second top FC layer of the VGG network. For ResNet-34 and

[7]Appendix I.3 shows how to invert features in RealNVP-LN.

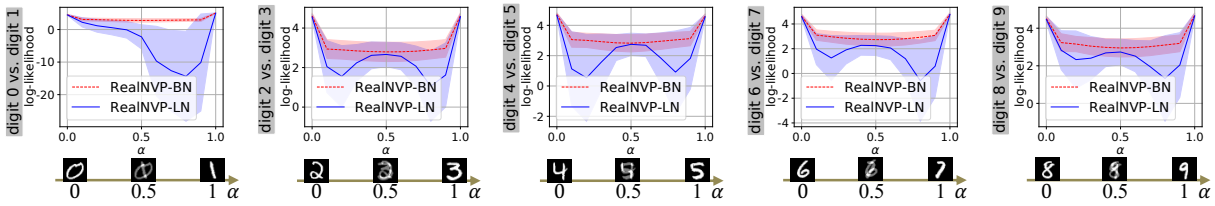

Figure 1: Log-likelihood of real images (at $\alpha = 0$ and $\alpha = 1$) and interpolated images generated by the RealNVP-BN and the RealNVP-LN. The shaded area represents the standard deviation.

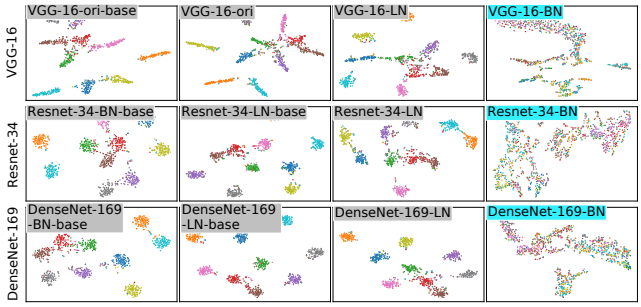

Figure 2: Comparing features of DNNs with and without BN layers. Points of different colors indicate samples of different categories. The DNN-BN usually learned much less clustered features than the DNN-LN, when these two DNNs were learned with the same settings.

DenseNet-169 that already contained BN operations, we directly selected them as DNN-BN networks, and there were no DNN-ori networks for such DNNs. In addition, we also learned two baselines, namely *DNN-BN-base* and *DNN-LN-base*, which were trained when all samples in a mini-batch had different labels.

In Figure 2, we used t-SNE (van der Maaten and Hinton 2008) to visualize the input feature of the top FC layer of each DNN, in order to compare the BN's effects on the feature representation. When losses of all samples in a mini-batch shared the same analytic formula, features of the DNN-BN on different samples were far less clustered than those of the DNN-ori and the DNN-LN. This indicates that the BN prevented the DNN from learning discriminative features of samples in each specific category. Results on more DNNs in Appendix I.4 also yielded similar conclusions.

In comparison, when losses of all samples in a mini-batch had different analytic formulas, the blocking problem did not significantly damage the feature representation of the DNN. It was because the remained terms, including both the residual term and effects of non-diagonal elements in $\mathbf{H}$, were already enough for classification.

The reason for the damage of feature representation might be that the first derivatives on the average sample $\tilde{\mathbf{y}}$ in a specific category usually contained information of common discriminative features shared by different samples in this category. Specifically, in Appendix I.8, we visualized features corresponding to the average first derivative $\mathbf{g}$ over different samples, so as to empirically justify the discrimination power of such features.

Experiment 3: We measured the BN's effects on the typical task of inverting feature gradients to the input in the scenario of privacy protection (Zhu, Liu, and Han 2019). We trained a decoder, *dec-BN*, to invert gradients received from the BN layer to reconstruct the input sample. We trained another decoder, *dec-base*, to invert gradients when we removed the BN layer. Experimental results in Appendix I.6 show that it took much more time to train the *dec-BN* than the *dec-base* to hack the private input. It was because the first derivatives $\mathbf{g}$ removed by the BN usually contained information of common features shared by most samples. Thus, removing $\mathbf{g}$ boosted the difficulty of training the *dec-BN*. On the other hand, the final fitting error of the *dec-BN* was lower than that of the *dec-base*. It was because the BN avoided $\mathbf{g}$ dominating the gradient signal, thereby letting the *dec-BN* pay more attention to more detailed difference between samples in a mini-batch. Please see Appendix I.6 for details.

Conclusions and Discussion

In this paper, we have discovered and theoretically proven the intrinsic blocking problem with the BN. Such a problem may bring in an uncommon yet non-ignorable risk in the learning of DNNs. Experiments have demonstrated that the BN's blocking prevents the DNN from learning discriminative features in specific applications.

Moreover, we have also proven that it is the standardization phase of the BN operation causes the above effects. However, it is difficult to obtain a simple conclusion that the standardization phase is harmful. In fact, Liu *et al.* (Liu et al. 2021) showed that the standardization phase in the BN could effectively alleviate the "self-enhancement" phenomenon, and avoided hurting the diversity of features in the DNN. Besides, Xu *et al.* (Xu et al. 2019) proved that the standardization phase in the LN (different from that in the BN) re-centered gradients of the loss *w.r.t.* features, and reduced the variance of these gradients. Xu *et al.* also found that the standardization phase improved the performance of DNNs in experiment. Nevertheless, we have proved that the standardization phase causes the blocking of the first and second derivatives of the loss function in specific applications. To this end, we use the LN operation to replace the BN operation in such applications, which avoids the blocking problem. We further introduces some limitations of applying our findings to real applications in Appendix A.

Acknowledgments

This work is partially supported by the National Key R&D Program of China (2021ZD0111602), the National Nature Science Foundation of China (62276165, 62206170), Shanghai Natural Science Foundation (21JC1403800, 21ZR1434600). This work is also partially supported by Alibaba Group through Alibaba Innovative Research Program.

References

Ba, J. L.; Kiros, J. R.; and Hinton, G. E. 2016. Layer normalization. *arXiv preprint arXiv:1607.06450*.

Cohen, J.; Kaur, S.; Li, Y.; Kolter, J. Z.; and Talwalkar, A. 2020. Gradient Descent on Neural Networks Typically Occurs at the Edge of Stability. In *International Conference on Learning Representations*.

Deng, H.; Ren, Q.; Chen, X.; Zhang, H.; Ren, J.; and Zhang, Q. 2022. Discovering and Explaining the Representation Bottleneck of DNNs. In *International Conference on Learning Representations*.

Dinh, L.; Krueger, D.; and Bengio, Y. 2014. Nice: Nonlinear independent components estimation. *arXiv preprint arXiv:1410.8516*.

Dinh, L.; Sohl-Dickstein, J.; and Bengio, S. 2017. Density estimation using Real NVP. In *International Conference on Learning Representations*.

Galloway, A.; Golubeva, A.; Tanay, T.; Moussa, M.; and Taylor, G. W. 2019. Batch normalization is a cause of adversarial vulnerability. *arXiv, abs/1905.02161*.

Goodfellow, I.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; and Bengio, Y. 2014. Generative Adversarial Nets. In Ghahramani, Z.; Welling, M.; Cortes, C.; Lawrence, N.; and Weinberger, K., eds., *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 770–778.

Huang, G.; Liu, Z.; van der Maaten, L.; and Weinberger, K. Q. 2017. Densely Connected Convolutional Networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Ioffe, S. 2017. Batch Renormalization: Towards Reducing Minibatch Dependence in Batch-Normalized Models. In Guyon, I.; Luxburg, U. V.; Bengio, S.; Wallach, H.; Fergus, R.; Vishwanathan, S.; and Garnett, R., eds., *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Ioffe, S.; and Szegedy, C. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, 448–456. PMLR.

Kingma, D. P.; and Dhariwal, P. 2018. Glow: Generative Flow with Invertible 1x1 Convolutions. In Bengio, S.; Wallach, H.; Larochelle, H.; Grauman, K.; Cesa-Bianchi, N.; and Garnett, R., eds., *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc.

Konečnỳ, J.; McMahan, B.; and Ramage, D. 2015. Federated optimization: Distributed optimization beyond the datacenter. *arXiv preprint arXiv:1511.03575*.

Krizhevsky, A.; Hinton, G.; et al. 2009. Learning multiple layers of features from tiny images.

Krizhevsky, A.; Sutskever, I.; and Hinton, G. E. 2012. ImageNet Classification with Deep Convolutional Neural Networks. In Pereira, F.; Burges, C.; Bottou, L.; and Weinberger, K., eds., *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc.

LeCun, Y.; Boser, B.; Denker, J. S.; Henderson, D.; Howard, R. E.; Hubbard, W.; and Jackel, L. D. 1989. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4): 541–551.

LeCun, Y.; Bottou, L.; Bengio, Y.; and Haffner, P. 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11): 2278–2324.

Li, X.; Chen, S.; Hu, X.; and Yang, J. 2019. Understanding the Disharmony Between Dropout and Batch Normalization by Variance Shift. In *Computer Vision and Pattern Recognition*.

Li, X.; Chen, S.; and Yang, J. 2020. Understanding the Disharmony between Weight Normalization Family and Weight Decay. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(04): 4715–4722.

Liu, D.; Wang, S.; Ren, J.; Wang, K.; Yin, S.; and Zhang, Q. 2021. Trap of Feature Diversity in the Learning of MLPs. *arXiv preprint arXiv:2112.00980*.

Niu, S.; Wu, J.; Zhang, Y.; Wen, Z.; Chen, Y.; Zhao, P.; and Tan, M. 2023. Towards Stable Test-time Adaptation in Dynamic Wild World. In *The Eleventh International Conference on Learning Representations*.

Simonyan, K.; and Zisserman, A. 2015. Very Deep Convolutional Networks for Large-Scale Image Recognition. In *ICLR*.

Tian, Y.; Chen, X.; and Ganguli, S. 2021. Understanding self-supervised learning dynamics without contrastive pairs. In *International Conference on Machine Learning*, 10268–10278. PMLR.

van der Maaten, L.; and Hinton, G. 2008. Visualizing data using t-SNE. *Journal of machine learning research*, 9(Nov): 2579–2605.

Van Laarhoven, T. 2017. L2 regularization versus batch and weight normalization. *arXiv preprint arXiv:1706.05350*.

Wightman, R.; Touvron, H.; and Jegou, H. 2021. ResNet strikes back: An improved training procedure in timm. In *NeurIPS 2021 Workshop on ImageNet: Past, Present, and Future*.

Xie, C.; Tan, M.; Gong, B.; Wang, J.; Yuille, A. L.; and Le, Q. V. 2020. Adversarial examples improve image recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 819–828.

Xu, J.; Sun, X.; Zhang, Z.; Zhao, G.; and Lin, J. 2019. Understanding and Improving Layer Normalization. In Wallach, H.; Larochelle, H.; Beygelzimer, A.; d'Alché-Buc, F.;

Fox, E.; and Garnett, R., eds., *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Zhu, L.; Liu, Z.; and Han, S. 2019. Deep Leakage from Gradients. In Wallach, H.; Larochelle, H.; Beygelzimer, A.; d'Alché-Buc, F.; Fox, E.; and Garnett, R., eds., *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.