BN-invariant Sharpness Regularizes the Training Model to Better Generalization

Mingyang Yi[1,2] , **Huishuai Zhang**[3] , **Wei Chen**[3] , **Zhi-Ming Ma**[1,2] and **Tie-Yan Liu**[3]
[1]University of Chinese Academy of Sciences
[2]Academy of Mathematics and Systems Science
[3] Microsoft Research
yimingyang17@mails.ucas.edu.cn, mazm@amt.ac.cn, {huzhang, wche, tie-yan.liu}@microsoft.com

Abstract

It is arguably believed that flatter minima can generalize better. However, it has been pointed out that the usual definitions of sharpness, which consider either the maxima or the integral of loss over a δ ball of parameters around minima, cannot give consistent measurement for scale invariant neural networks, e.g., networks with batch normalization layer. In this paper, we first propose a measure of sharpness, BN-Sharpness, which gives consistent value for equivalent networks under BN. It achieves the property of scale invariance by connecting the integral diameter with the scale of parameter. Then we present a computation-efficient way to calculate the BN-sharpness approximately i.e., one dimensional integral along the ”sharpest” direction. Furthermore, we use the BN-sharpness to regularize the training and design an algorithm to minimize the new regularized objective. Our algorithm achieves considerably better performance than vanilla SGD over various experiment settings.

1 Introduction

With the support of big data, deep learning techniques has achieved a huge success across domains. In recent years, it becomes a common sense that deep neural networks have perfect training performance. Specifically, [Zhang *et al.*, 2016] empirically shows that the popular neural networks (NN) can always reach zero training error at the end of training process. Moreover, [Du *et al.*, 2018] and [Allen-Zhu *et al.*, 2018] prove that, for the neural networks with sufficient width, gradient descent converges to the global minima of the objective, under some conditions on the data generation. Thus how to find minima taht generalize well seems to be more critical in the study of deep neural network.

It’s arguably believed that minima locate in a flat valley generalize better. [Neyshabur *et al.*, 2017] employs the PAC Bayes theory [McAllesterl, 1998] and proves a generalization bound give by ”expected sharpness”. The conclusion gives us a new angle to discuss generalization error of a neural network. This viewpoint is also empirically confirmed by [Keskar *et al.*, 2017] and [Li *et al.*, 2018]. However, for neural network with scale invariant property, the definitions of sharpness become problematic.

One important case is deep neural network with batch normalization (BN) [Ioffe and Szegedy, 2015]. BN is an important component which normalize the distribution of input to each neuron in the network by mean and standard deviation of input computed over a mini-batch of training data. Batch normalization will bring a scale invariant property to neural network which will make the definition of sharpness for neural network problematic (two parameter points with same generalization error but different sharpness). Since sharpness is related to generalization, a natural question is can we develop an appropriate definition of sharpness for neural network with batch normalization? Meanwhile, can we leverage the appropriate measurement of sharpness to develop an algorithm which helps us find minima generalize better?

In this paper, we answer the two above problems positively. First, we show that the original definition of sharpness for neural network with batch normalization is ambiguous. Specifically, the original sharpness (δ-sharpness) consider the maxima of loss over a δ ball of parameters around minima with radius δ unrelated to parameter θ. However, due to the scale invariant property of neural network with batch normalization, the radius δ should scale with parameter norm θ. On the other hand, the new measurement of sharpness should be computational efficient since we want to leverage it to help us find minima generalize better. Based on the two above points, we propose a ”BN-sharpness” to measure the sharpness for BN neural network. In specific, BN-Sharpness is a one dimensional integral of the loss’s difference value along the ”sharpest” direction in a small neighborhood while the diameter of this neighborhood is related with parameter scale. It’s an intuitively thought that a minimum locate in a flat valley when the loss values in the valley are closely with each other along the steepest changing direction.

In order to find minima locate in flat valley which generalize better, we penalize the original optimization objective with BN-Sharpness. Due to the one dimensional integral structure of BN-Sharpness, we can easily acquire the ”sharpest direction” by optimization on manifold [Boumal *et al.*, 2017] as well as the gradient of BN-Sharpness.

It’s empirically showed that SGD with large batch size tend to produce ”sharp” minima which generalize worse [Keskar *et al.*, 2017]. We test our algorithm on CIFAR dataset

[Krizhevsky *et al.*, 2012], it has preferable results under large batch size compared with baselines (SGD, Entropy SGD).

1.1 Related Work

Sharpness is a property connected with the loss surface. [Li *et al.*, 2018] first visualize the loss landscape of NN to describe that "sharp" minima indeed generalize worse. Such phenomenon was also empirically confirmed by [Keskar *et al.*, 2017] and [Wu *et al.*, 2018]. [Neyshabur *et al.*, 2017] also gives the conclusion a theoretical interpretation.

In fact, the definition of sharpness is proved to be ambiguous for ReLU neural network [Dinh *et al.*, 2017]. Suffering from the definition of sharpness and computational complexity, some local structured algorithms aimed to produce "flat" minima avoid exactly definition of sharpness. Entropy SGD proposed by [Chaudhari *et al.*, 2017] is motivated by the local geometry of the energy landscape, it alters loss as accumulated "energy value" instead of single point value. [Wen *et al.*, 2018] argues that choosing integral of a small cubic around iterate as loss function can produce a smoother loss function which will lead to a "flat" minimum. [Wang *et al.*, 2018] sets an expected loss function to produce "flatter" minima under PAC Bayes theory. However, all of these algorithms will face a high dimensional integral.

There are lots of sharpness based algorithms to find flat minima [Chaudhari *et al.*, 2017; Wang *et al.*, 2018; Wen *et al.*, 2018]. Since it's hard to compute the gradient of the original sharpness, they set optimization objective as integral of the loss function in a ball to force iterates walking into a flat valley instead of optimizing original loss function regularized by sharpness. This kind of optimization objective involves a high dimensional integral which is actually hard to be computed in practice.

For the topic of neural network with batch normalization, [Cho and Lee, 2018] and [Huang *et al.*, 2017] give Riemannian approach to optimize neural network with batch normalization. [Yuan *et al.*, 2018] propose a general variant of batch normalization to speed up training process. However, none of these works consider combining the sharpness with batch normalization to reduce generalization error.

1.2 Contribution

(1) We show that the generally accepted definition of sharpness: δ-sharpness [Keskar *et al.*, 2017] is ill-posed for NN with BN. We propose a scale invariant and computational efficient "BN-Sharpness" to measure sharpness for NN with BN.
(2) We present a computational efficient algorithm based on BN-Sharpness to enhance generalization ability of iterates which has preferable experimental results.

2 Background

2.1 Definitions of Sharpness

There are lots of definitions to describe sharpness [Keskar *et al.*, 2017; Wu *et al.*, 2018]. We focus on a widely discussed definition: δ-sharpness this paper.

Definition 2.1 (**δ-L^p sharpness**) *Let $B_2(\theta,\delta)$ be an Euclidean ball centered on a minimum θ with radius δ. Then, for a non-negative valued loss function $L(\cdot)$ and non-negative number p, the δ-L^p sharpness will be defined as*

$$S^p_{\delta\text{-}sharpness}(\theta)=\frac{\left(\int_{\theta'\in B_2(\theta,\delta)}|L(\theta')-L(\theta)|^p\,d\theta'\right)^{\frac{1}{p}}}{1+L(\theta)}. \quad (1)$$

In fact, denominator of equation (1) will close to 1 when $L(\theta)$ is small. Thus, we ignore it during discussion. We can actually prove that δ-sharpness $S_{\delta\text{-sharpness}}$ used in [Keskar *et al.*, 2017] is actually δ-L^∞ sharpness

2.2 Batch Normalization

We briefly revisit the batch normalization and its properties. A network with BN transforms the input value to a neuron from $z={}'Tx$ to

$$BN(z)=\gamma\frac{z-E(z)}{\sqrt{Var(z)}}+\beta=\gamma\frac{\theta^T(x-E(x))}{\sqrt{\theta^TV_x\theta}}+\beta. \quad (2)$$

We can easily verify $BN(az)=BN(z)$ for any $a>0$. Then, for a partially batch normalized neural network (Some of input values to neuron are batch normalized) with N parameter vector $\theta=(\theta_1,\cdots,\theta_{N_1},\theta_{N_1+1},\cdots,\theta_N)^T$, where θ_i $(i=1\cdots N_1)$ is parameter vector connected with batch normalized neuron and $(\theta_{N_1+1},\cdots,\theta_N)$ connect neuron without batch normalization. We can actually view the affine parameters γ,β and bias parameters of each neuron as un-batch normalized parameters. We integrate them into parameter vector θ during our latter discussion. In the latter discussion, our analysis is applied to partially batch normalized neural network if we don't give an additional illustration. Now we give the definition of scale transformation for partially batch normalized neural network.

Definition 2.2 *$T_{\vec{a}}(\cdot)$ denote scale transformation for a partially batch normalized neural network with*

$$T_{\vec{a}}(\theta)=(a_1\theta_1,\cdots a_{N_1}\theta_{N_1},\theta_{N_1+1},\cdots,\theta_N)^T, a_i>0. \quad (3)$$

We see the loss function $L(\theta)$ satisfy $L(\theta)=L(T_{\vec{a}}(\theta))$ for any $T_{\vec{a}}(\cdot)$ [1]. We call this the property of scale invariant of BN. In fact, the ambiguous of sharpness is brought by such property.

3 BN-Sharpness

In this section, we give a δ-BN sharpness to measure sharpness. A well defined geometrical measurement linked with generalization error for neural network with batch normalization should be scale invariant. We start with theorem to explain why the original measurement of sharpness is inappropriate.

Theorem 3.1 *Given a partially batch normalized network, δ-sharpness is not scale invariant.*

[1]For the situation of Pre-ResNet which has input $BN((\vec{1}+\theta)^Tz)$ in skip connection layer. It's not a scale invariant layer, we treat these θ as un-batch normalized parameters.

Proof 3.1 For $S_{\delta\text{-sharpness}}(\cdot)$, We consider a partially batch normalized neural network, without of loss generality we assume the network has batch normalized neuron in one layer. Then we choose $\vec{a}$ as $(1, \cdots, a_0, \cdots, a_0, 1, \cdots, 1)^T$ where a_0 locate in the same coordinate with batch normalized parameter. For any $\delta > 0$, we can choose

$$0 < a_0 \leq \frac{\delta}{\sqrt{N} \max_{1 \leq i \leq N} \|\theta_i\|} \tag{4}$$

with $L(T_{\vec{a}}(\theta)) = L(\theta)$. Because a parameter θ with one layer zero weight matrix locate in $B_2(T_{\vec{a}}(\theta), \delta)$ which result in $\max_{\theta' \in B_2(\theta,\delta)} (L(\theta') - L(\theta))$ is at least as high as constant function. Then, the value of $S_{\delta\text{-sharpness}}(T_{\vec{a}}(\theta))$ is relatively large. ■

The above result reveals that the original definition of sharpness is not well defined for BN-network. Actually, we can achieve a minimum with small generalization error but infinite sharpness if we rescale the minimizer adequately close to zero for δ-sharpness. Hence, using δ-sharpness as a measurement of generalization is meaningless. We see the ill-posed issue of δ-sharpness suffers from the scale invariant ability of batch normalization network. Based on Theorem 3.1, we aim to derive a *scale invariant* sharpness.

Due to these, we present a scale invariant meanwhile computational efficient measurement: ”BN-Sharpness”, which is instructive for us to leverage it to find ”flat” minima. Now, we give the exact definition of ”BN-Sharpness”.

Definition 3.1 (δ-L^p BN-Sharpness) *Given a positive number δ, and a parameter point θ, and a partially batch normalized network, the δ-L^p BN-Sharpness is defined as*

$$\|L(\cdot)\|_p^{\delta,\theta} = \sup_{v \in \phi(\theta)} \frac{1}{\delta^{\frac{1}{p}}} \left(\int_{-\delta}^{\delta} \left| \frac{L(\theta + tv) - L(\theta)}{\delta} \right|^p dt \right)^{\frac{1}{p}}, \tag{5}$$

where $\phi(\theta)$ is a set composed by v with $\|v_i\| = \|\theta_i\|, \sqrt{\sum_{j=N_1+1}^{N} \|v_j\|^2} = 1; i = 1, \cdots, N_1$.

We notice $v \in \phi(\theta)$ has same dimension with θ. And it's decided by the parameter of a partially batch normalized network. Since $v \in \phi(\theta)$ has identically l_2 norm which is $\sqrt{\sum_{i=1}^{N_1} \|\theta_i\|^2 + 1}$, we use $\|\phi(\theta)\|$ to denote it. For simplicity, we use BN-Sharpness to substitute δ-L^p BN-Sharpness.

As we have discussed in Section 2, the L^p norm of function $L(\cdot) - L(\theta)$ defined in $B_2(\theta, \delta)$ can be a measurement of sharpness (δ-sharpness is L^∞ norm). The crucial problem of such L^p-sharpness is losing sight of parameter scale when choosing region diameter δ. On the other hand, it's a high dimensional integral ($p < \infty$) which is hard to be computed in practice let alone combining them to reduce sharpness. The two shortages are also hold when $p = \infty$.

We see BN-Sharpness consider the scale of batch normalized parameter. Meanwhile BN-Sharpness is an one dimensional integral along the sharpest direction v of $L(\theta)$ for each parameter component θ_i. It will be computational efficient especially for the gradient which we will discuss it more specifically in Section 5. By Taylor's expansion, for each parameter component θ_i we have

$$\begin{aligned} L(\theta + tv) - L(\theta) &\approx \sum_{i=1}^{N} t \nabla_\theta L(\theta)_i^T v_i + o(t\|v\|) \\ &\leq t \sum_{i=1}^{N} \|\nabla_\theta L(\theta)_i\| \|v_i\| + o(t\|\phi(\theta)\|). \end{aligned} \tag{6}$$

Here $\nabla L(\theta)_i$ represent gradient of $L(\cdot)$ to component parameter vector θ_i. The equation (6) gives an approximate calculating of the ”sharpest” direction when δ is small. Precisely calculation of BN-Sharpness requires optimization on Oblique manifold which we will discuss in Section 4.

Now, we prove that BN-Sharpness is scale invariant. Hence, it's appropriate to measure sharpness for BN neural network.

Theorem 3.2 *The loss function $L(\theta)$ of a DNN with batch normalization satisfies $\|L(\cdot)\|_p^{\delta,\theta} = \|L(\cdot)\|_p^{\delta, T_{\vec{a}}(\theta)}$ for any $\vec{a}$ without negative element.*

Proof 3.2 For any $a \neq 0$, we notice that $L(\theta) = L(T_{\vec{a}}(\theta))$. Therefore for any v with $v \in \phi(\theta)$, we have

$$L(\theta + tv) - L(\theta) = L(T_{\vec{a}}(\theta) + tT_{\vec{a}}(v)) - L(T_{\vec{a}}(\theta)), \tag{7}$$

for $t \in [-\delta, \delta]$. It's easily to verify that

$$\begin{aligned} &\int_{-\delta}^{\delta} \left| \frac{L(\theta + tv) - L(\theta)}{\delta} \right|^p dt \\ &= \int_{-\delta}^{\delta} \left| \frac{L(T_{\vec{a}}(\theta) + tT_{\vec{a}}(v)) - L(T_{\vec{a}}(\theta))}{\delta} \right|^p dt \end{aligned} \tag{8}$$

Since

$$\begin{aligned} &\sup_{v \in \phi(\theta)} \frac{1}{\delta^{\frac{1}{p}}} \left(\int_{-\delta}^{\delta} \left| \frac{L(T_{\vec{a}}(\theta) + tT_{\vec{a}}(v)) - L(T_{\vec{a}}(\theta))}{\delta} \right|^p dt \right)^{\frac{1}{p}} \\ &= \sup_{v \in \phi(T_{\vec{a}}(\theta))} \frac{1}{\delta^{\frac{1}{p}}} \left(\int_{-\delta}^{\delta} \left| \frac{L(T_{\vec{a}}(\theta) + tv) - L(T_{\vec{a}}(\theta))}{\delta} \right|^p dt \right)^{\frac{1}{p}} \\ &= \|L(\cdot)\|_p^{\delta, T_{\vec{a}}(\theta)}, \end{aligned} \tag{9}$$

we get the conclusion. ■

Actually, we can derive a relationship between BN-Sharpness and generalization error. The result is based on PAC Bayesian theory [McAllesterl, 1998; 1999].

Theorem 3.3 *Given a ”prior” distribution P(for parameter θ) over the hypothesis that is independent of the training data, with probability at least $1 - \varepsilon$, we have*

$$\begin{aligned} |\mathbb{E}_u[L(\theta + u)] - \hat{L}(\theta)| &\leq \delta^{1+\frac{1}{p}} \|L(\cdot)\|_p^{\delta,\theta} \\ &+ 4\sqrt{\frac{1}{m}\left(KL(\theta + u||P) + \log\frac{2m}{\varepsilon}\right)}, \end{aligned} \tag{10}$$

where $L(\cdot)$ and $\hat{L}(\cdot)$ are respectively expected loss and training loss, m is the number of training data and θ is the parameter learned from training data. As long as u is an uniform distribution on any specific $v \in \delta \cdot \phi(\theta)$.

A straightforward connection between generalization error $|L(\theta) - \hat{L}(\theta)|$ and sharpness is nontrivial. We can only derive a perturbed generalization error $|\mathbb{E}_u[L(\theta+u)] - \hat{L}(\theta)|$, where u is the perturbation variable. A small perturbation variable u will make the perturbed generalization error close to the real generalization error which involves a small δ in equation (10). We claim that the result gives a quantitatively description of generalization error based on BN-Sharpness. It's a direct corollary according to the equation (6) in McAllester [McAllesterl, 2003] and the definition of BN-Sharpness.

4 Regularizing Training with BN-Sharpness

We already have an appropriate definition of BN-Sharpness. It has also been confirmed that smaller BN-Sharpness leads to better generalization. We naturally consider leveraging it to find a flat minimum generalizes better.

Intuitively, we consider using BN-Sharpness as a regularization term tend to produce flat minima for BN neural network. We substitute the optimization problem $\min_\theta L(\theta)$ with

$$\min_\theta L(\theta) + \lambda(\|L(\theta')\|_p^{\delta,\theta})^p. \tag{11}$$

An interesting phenomenon is such regularization term not only computational efficient and appropriate but also well-posed. More specifically, traditional regularization term such as l_1, l_2 would change the minimums set of loss function. But BN-Sharpness regularization term keeps minima of original loss function in the minimal point set of regularized loss function, while it force iterates move to a flat valley. The next theorem states that regularization term in (11) wouldn't remove minima of original loss function when δ is small.

Theorem 4.1 *The minima of problem $\min_\theta L(\theta)$ are also minima of optimization problem (11), when $\delta \to 0$ and $p \geq 1$.*

The optimization problem (11) is actually a multi-target programming with "accuracy target" and "flatness target". λ in (11) is actually a proportion between the two terms. Since we aim to find "flat" minima rather than "flat" point, an appropriate proportion between the two purpose is crucial.

Now, we give a computational efficient algorithm to solve the optimization problem (11). Here we simply explain our algorithm flow. We notice that the obstacle of optimization problem (11) is calculating the gradient of regularization term $(\|L(\theta')\|_p^{\delta,\theta})^p$ (BN-Sharpness). It involves two steps: First, calculating the "sharpest" direction which we need optimization on Oblique manifold; Second, computing the gradient of integral term in BN-Sharpness under the "sharpest" direction.

Now we give the complete flow of our algorithm: Algorithm 1. Then we will discuss more details about it.

In Algorithm 1, the first inner loop is calculating the "sharpest" direction which is the process of optimization on Oblique manifold. The second inner loop is general gradient descent to update parameter θ. In addition, we make an approximation to $\lambda\nabla_\theta \frac{1}{\delta}\int_{-\delta}^{\delta}\left(\frac{L(\theta+tv)-L(\theta)}{\delta}\right)^p dt$ and $\nabla_v \frac{1}{\delta}\int_{-\delta}^{\delta}\left(\frac{L(\theta+tv)-L(\theta)}{\delta}\right)^p dt$ in equation (15) and (16) to further reduce computational redundancy. We respectively denote them as $h_1(\theta, v, \delta, p, \lambda)$ and $h_2(\theta, v, \delta, p)$.

Algorithm 1 SGD with BN-Sharpness regularization

```
Input $\delta > 0$, even number $p$, initialize point $\theta^0 \in \mathbb{R}_n$, $v_0 \in \phi(\theta^0)$ set to be the gradient direction like equation (6), regularization coefficient $\lambda$, iterations $K_1, K_2$ and learning rate $\eta$.
Optimization with BN-Sharpness term
$k_2 = 0$
while $k_2 \leq K_2$ and $\nabla_\theta L(\theta^{k_2}) \neq 0$ do
    Iterate $K_1$ times to search the sharpest direction $v^{k_1}(\theta^{k_2})$ in point $\theta^{k_2}$
    $\theta^{k_2+1} = \theta^{k_2} - \eta(\nabla_\theta L(\theta^{k_2}) + h_1(\theta^{k_2}, v^{k_1}(\theta^{k_2}), \delta, p, \lambda))$,
    $k_2 = k_2 + 1, v_0 \in \phi(\theta_{k_2})$
end while
return $\theta_{k_2}$
```

4.1 Searching the Sharpest Direction v

The extra computation cost comparing to vanilla SGD is the procedure of searching the "sharpest" direction v which is the second inner loop in Algorithm 1. Inspiring by equation (6), we can also choose $v_k \in \phi(\theta_k)$ with v_k^i has same direction with $\nabla L(\theta_k)_i$ for each step to avoid the searching process (Iterate K_1 times in Algorithm 1).

However, there is a more accurate searching procedure which is optimization on Oblique manifold. Searching v in BN-Sharpness (5) is equivalent to solving optimization problem:

$$\arg\max_{v\in\phi(\theta)} \frac{1}{\delta^{\frac{1}{p}}}\left(\int_{-\delta}^{\delta}\left|\frac{L(\theta+tv)-L(\theta)}{\delta}\right|^p dt\right)^{\frac{1}{p}}. \tag{12}$$

This is a constrained optimization problem which can be converted to optimization on manifold. We briefly present optimization on manifold here. Detailed introduction can be referred to [Boumal *et al.*, 2017].

Optimization on manifold converts a constrained optimization problem into an un-constraint problem while iterates locate in a manifold satisfy the constraint. Specific definition of manifold $\mathcal{M}$ can be found in [Pierre-Antoine *et al.*, 2009]. Here we only consider matrix manifold i.e. a subspace of Euclidean space. Solving problem (12) needs gradient assent on manifold which produce iterates as

$$v_{k+1} = \mathrm{Retr}_{v_k}\left(\frac{1}{L}\mathrm{grad}f(v_k)\right). \tag{13}$$

Here $\mathrm{grad}f(x)$ is Riemannian gradient which is a map from tangent space $T_x\mathcal{M}$ of point x to manifold $\mathcal{M}$. The optimization problem (12) defined on the general Oblique manifold. We need the related formulations in Oblique manifold to process our algorithm.

Definition 4.1 (Oblique manifold) *Oblique manifold is a subset of Euclidean space satisfy* $\mathrm{St}(n,p) = \{X \in \mathbb{R}^{n\times p} : \mathrm{ddiag}(X^TX) = I_p\}$*, where* $\mathrm{ddiag}(\cdot)$ *is diagonal matrix of a matrix.*

Apparently, for a given θ in equation (12), v lives in a special Oblique product manifold $\|\theta\|_1\mathrm{St}(n_1,1) \times \cdots \times \|\theta\|_{N_1}\mathrm{St}(n_{N_1},1) \times \mathrm{St}(\sum_{j=N_1+1}^{N} n_j, 1)$ where n_i is the dimension of parameter θ_i.

As we discussed, optimization on manifold requires Retr_x and $\mathrm{grad}f(x)$. The two items on single Oblique manifold $\|\theta\|\mathrm{St}(n,1)$ respectively are

$$\mathrm{Retr}_x^{\|\theta\|}(\eta)=\frac{x+\eta}{\|x+\eta\|}\|\theta\|, P_x^{\|\theta\|}(\eta)=\eta-\frac{x}{\|\theta\|^2}\mathrm{ddiag}(x^T\eta),$$
$$\mathrm{grad}f(x)=P_x^{\|\theta\|}\left(\nabla f(x)\right)=\nabla f(x)-\frac{x}{\|\theta\|^2}\mathrm{ddiag}(x^T\nabla f(x)) \tag{14}$$

where $P_x^{\|\theta\|}(\cdot)$ is projection matrix of the tangent space at x. These results can be derived from the general formulas in [Pierre-Antoine *et al.*, 2009; Pierre-Antoine and Gallivan, 2006]. Then we can generalize it to achieve the update rule in product manifold we used. The procedure is gradually update v_i^k according to the manifold it located [Cho and Lee, 2018].

In addition, we note that our algorithm involves $\nabla_\theta\frac{1}{\delta}\int_{-\delta}^{\delta}\left(\frac{L(\theta+tv)-L(\theta)}{\delta}\right)^p dt$ (The gradient descent for parameter θ) and $\nabla_v\frac{1}{\delta}\int_{-\delta}^{\delta}\left(\frac{L(\theta+tv)-L(\theta)}{\delta}\right)^p dt$ (Searching the sharpest direction). Precise calculation of the two terms will bring some extra computational redundancy. Because $L(\theta)$ usually be a neural network, but calculating the integral term requires sampling value of $L(\theta+tv)$ and $\nabla L(\theta+tv)$. Each sampling corresponds to a forward and backward process of neural network. The next proposition gives an approximation to the gradient of BN-Sharpness. It reduces the times of froward and backward process to two. In the next proposition, we aim to a specific v and suppose that v is a constant vector.

Proposition 4.1 *Given $\delta>0$, for any v satisfy $\|v\|=\|\phi(\theta)\|$, we have*

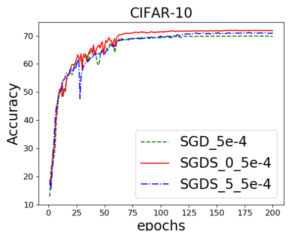
$$\Bigg\|\lambda\nabla_\theta\frac{1}{\delta}\left(\int_{-\delta}^{\delta}\left(\frac{L(\theta+tv)-L(\theta)}{\delta}\right)^p dt\right)-\frac{\lambda}{\delta}(\nabla_\theta L(\theta)^T v)^{p-1}$$
$$(\nabla_\theta L(\theta+\frac{p}{p+1}\delta v)+\nabla_\theta L(\theta-\frac{p}{p+1}\delta v)-2\nabla_\theta L(\theta))\Bigg\|<o(\delta\|v\|), \tag{15}$$

when p is an even number.

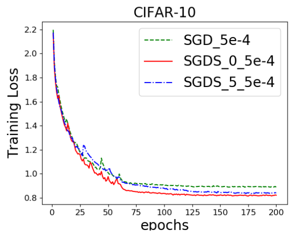
Proposition 4.1 indicates the gradient respect to parameter θ can be replaced by a computational efficient term. On the other hand, we can achieve a similar result presented as

$$\Bigg\|\nabla_v\frac{1}{\delta}\left(\int_{-\delta}^{\delta}\left(\frac{L(\theta+tv)-L(\theta)}{\delta}\right)^p dt\right)-\frac{p}{p+1}(\nabla_\theta L(\theta)^T v)^{p-1}$$
$$[\nabla L_v(\theta+\frac{p+1}{p+2}\delta v)+\nabla_v L(\theta-\frac{p+1}{p+2}\delta v)]\Bigg\|<o(\delta\|v\|). \tag{16}$$

for v. The equation gives an approximation for the gradient respect to vector v locate in a product Oblique manifold.



(a) Accuracy on CIFAR10



(b) Training loss on CIFAR10

Figure 1: Performance on LeNet

We will use the two approximation terms to substitute the original gradients of BN-Sharpness in our algorithm. Finally, we point out that the sharpest direction v_k of point θ in Algorithm 1 is produced as

$$v_i^k=\mathrm{Retr}_{v_i^{k-1}}^{\|\theta_i\|}\left(P_{v_i^{k-1}}^{\|\theta_i\|}(h_2^i(\theta,v^{k-1},\delta,p))\right), i=1,\cdots N_1$$
$$v_*^k=\mathrm{Retr}_{v_*^{k-1}}^{1}\left(P_{v_*^{k-1}}^{1}(h_2^i(\theta,v^{k-1},\delta,p))\right), \tag{17}$$

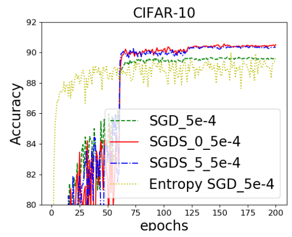
where $v_*=(v_{N_1+1}^T,\cdots,v_N^T)^T$.

5 Experiment

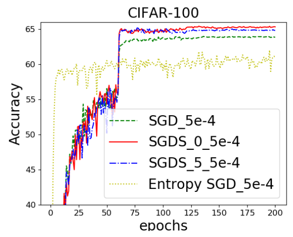
The previous work indicate that SGD with large batch size produces sharp minima while small batch size can avoid sharp minima itself [Keskar *et al.*, 2017]. Therefore, in order to finding flat minima under large batch size, we use Algorithm 1 to reach such target. We consider our algorithm should have a preferable result comparing to SGD under large batch size. We should highlight that the biases gradient $\nabla_\theta L\left(\theta+\frac{p}{p+1}\delta v\right)$ and $\nabla_\theta L\left(\theta-\frac{p}{p+1}\delta v\right)$ in equation (15) are calculated by different batch data. It's a way of reducing variance which is used in Entropy SGD [Chaudhari *et al.*, 2017].

First we test the algorithm with fully batch normalized LeNet [LeCun *et al.*, 1998] to test the performance for CIFAR10 [Krizhevsky *et al.*, 2012]. Update rule is SGD with momentum by setting learning rate as 0.2 and decay it by a factor 0.1 respectively in epoch 60, 120, 160 and momentum parameter as 0.9. We use 10000 batch size, and 5e-4 weight decay ratio for all the three experiments.

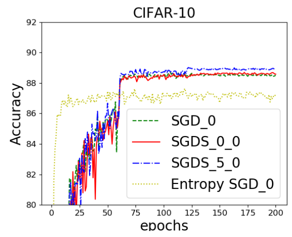
For experiments with regularization term, we clip the gradient of BN-Sharpness by norm with factor 0.1. We use "SGDS-number-decay" to represent the algorithm regularized by BN-Sharpness iterate "number" times of searching sharpest direction and using a weight decay ratio as "decay". For example, "SGDS-5-5e-4" means iterate 5 times of searching v in 3.1 (K_1=5 in Algorithm 1) and setting weight decay ratio as 5e-4. In addition, the initial point of iteration is set to be the gradient direction like in equation (6). For the experiments with regularized BN-Sharpness, we choose λ as 1e-4 which increase by a factor of 1.02 for each epoch. We set $\delta=0.001$, and the p is chosen as 2.



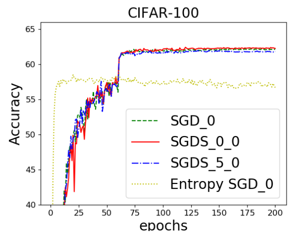
(a) Accuracy of 2048 batch size with weight decay on CIFAR10



(b) Accuracy of 2048 batch size with weight decay on CIFAR100



(c) Accuracy of 2048 batch size without weight decay on CIFAR10



(d) Accuracy of 2048 batch size without weight decay on CIFAR100

Figure 2: Performance on VGG11

Algorithm	SGD-5e-4	SGDS-0-5e-4	SGDS-5-5e-4	E-SGD-5e-4
CIFAR10	91.67	91.78	91.55	91.03
CIFAR100	69.33	69.07	68.53	67.37
Algorithm	SGD	SGDS-0-0	SGDS-5-0	E-SGD-0
CIFAR10	90.5	90.53	90.53	89.96
CIFAR100	65.5	65.45	64.82	63.45

Table 1: Performance of VGG with batch size as 128.

Algorithm	SGD-5e-4	SGDS-0-5e-4	SGDS-5-5e-4	E-SGD-5e-4
CIFAR10	89.6	90.51	90.35	89.67
CIFAR100	63.83	65.31	64.8	61.02
Algorithm	SGD	SGDS-0-0	SGDS-5-0	E-SGD-0
CIFAR10	88.5	88.58	88.9	87.13
CIFAR100	62.17	62.25	61.8	56.68

Table 2: Performance of VGG with batch size as 2048.

The accuracy result is referred to Figure 1a, where the accuracy results of ”SGD”, ”SGD with sharpness” and ”SGD with sharpness iteration=5” are respectively 69.87, 71.84, 70.95. We see that our algorithm has a significant promotion on such toy example. Now we test our algorithm on a deeper network VGG11 with bath normalization [Simonyan and Zisserman, 2014]. We respectively test vanilla SGD, SGD with BN-Sharpness regularization and Entropy-SGD (represented as E-SGD) on the network. The experiments can be divided into two groups, large batch size and small batch size. The batch size we used for the two groups of experiments are respectively 128 and 2048.

For SGDS, the δ in CIFAR10 is 5e-4 and in CIFAR100 is 1e-3, learning rate is 0.2 and decay by a factor 0.1 respectively in epoch 60, 120, 160. The learning rate for SGD is 0.1 and we decay it like SGDS. Other hyper-parameters we used follow the setting in the first experiment of LeNet.

For Entropy SGD we follow the same hyper parameters in [Chaudhari *et al.*, 2017] for experiments with 2048 batch size, except for the learning rate. We set learning rate like SGD experiments. But for experiment with 128 batch size, we turn off the drop out and use a 0.01 global learning rate. We also didn’t adjust the γ accord with iteration times. These adjustments will make Entropy-SGD perform better. From the results, we conclude that SGD with small batch size can avoid ”sharp” minima itself, and regularized by sharpness has little influence. Here we don’t pay a lot attention to searching the sharpest direction. This is motivated by reducing computation complexity as well as balancing regularized term and loss function.

We emphasize that dividing the sharpness target and accuracy target is important. It allows us to dynamically adjust ”flat” target which avoid iterates stack in a wide valley when accuracy in a low standard. Actually, if we use the same hyper parameters for Entropy SGD under batch size 128 and 2048. The training accuracy will end in a low level. Even though, Entropy SGD still perform general. We think it suffer from two aspects: First, sampling 20 points($L = 20$ in Entropy SGD) for a huge model is not enough, which make it unstable. It will present a high variance result; Second, iterates tend to stack in a inaccurate point, since the loss function may be inexact (See Supplement Material).

Training model with SGD for 600 epochs (adjust learning rate by iterations) can reach the results of SGDS. It’s a viewpoint for large batch size SGD: Training longer, generalize better [Hoffer *et al.*, 2017]. However, our method can reach such result under fewer iterations.

6 Conclusion

We first prove that tradition definitions of sharpness are insufficient to describe geometrical structure of neural network with batch normalization. Based on that, we propose a scale invariant BN-Sharpness to measure the sharpness of minima. Then, we give an algorithm based on BN-Sharpness. It has computational advantage comparing to existing sharpness based algorithms.

For all algorithms in order to find ”flat” minima (including our algorithm), the training purpose is finding a ”flatter” minima generalize better, rather than finding a ”flatter” point. Therefore, for such kind of algorithms, setting a large proportion to sharpness target will easily force iterates walk into a ”flat” local minimum that generalize poor. Therefore, balancing the sharpness target and loss target is a delicate but meaningful problem.

Finally, we don’t particularly discuss the combination of our algorithms with training tricks under large batch size such as [Hoffer *et al.*, 2017], [Choromanska *et al.*, 2015] and [De *et al.*, 2017]. In fact, combing them together it’s a very interesting topic.

References

[Allen-Zhu *et al.*, 2018] Zeyuan Allen-Zhu, Yuanzhi Li, and Zhao Song. A convergence theory for deep learning via over-parameterization. *arXiv preprint arXiv:*, 2018.

[Boumal *et al.*, 2017] Nicolas Boumal, Absil Pierre-Antoine, and Coralia Cartis. Global rates of convergence for nonconvex optimization on manifolds. *arxiv preprint arxiv:1605.08101*, 2017.

[Chaudhari *et al.*, 2017] Pratik Chaudhari, Anna Choromanska, Stefano Soatto, Yann LeCun, Carlo Baldassi, Christian Borgs, Jennifer Chayes, Levent Sagun, and Riccardo Zecchina. Entropy-sgd: Biasing gradient descent into wide valleys. *In ICLR'2017 arXiv:1611.01838*, 2017.

[Cho and Lee, 2018] Minhyung Cho and Jaehyung Lee. Riemannian approach to batch normalization. *arXiv preprint arXiv:1709.09603*, 2018.

[Choromanska *et al.*, 2015] Anna Choromanska, Mikael Henaff, Michael Mathieu, Gérard Ben Arous, and Yann LeCun. The loss surfaces of multilayer networks. *International Conference on Artificial Intelligence and Statistics(AISTATS 2015)*, 2015.

[De *et al.*, 2017] Soham De, Abhay Yadav, David Jacobs, and Tom Goldstein. Automated inference with adaptive batches. *International Conference on Artificial Intelligence and Statistics(AISTATS 2017)*, 2017.

[Dinh *et al.*, 2017] Laurent Dinh, Razvan Pascanu, Samy Bengio, and Yoshua Bengio. Sharp minima can generalize for deep nets. *In ICML'2017 arXiv:1703.04933*, 2017.

[Du *et al.*, 2018] Simon S. Du, Jason D. Lee, Haochuan Li, Liwei Wang, and Xiyu Zhai. Gradient descent finds global minima of deep neural networks. *arXiv preprint arXiv:1811.03804*, 2018.

[Hoffer *et al.*, 2017] Elad Hoffer, Itay Hubara, and Daniel Soudryn. Train longer, generalize better: closing the generalization gap in large batch training of neural networks. *Advances in Neural Information Processing Systems 30 (NIPS 2017)*, 2017.

[Huang *et al.*, 2017] Lei Huang, Xianglong Liu, Bo Lang, Adams Wei Yu, Yongliang Wang, and Bo Li. Orthogonal weight normalization: Solution to optimization over multiple dependent stiefel manifolds in deep neural networks. *arXiv preprint arXiv:1709.06079*, 2017.

[Ioffe and Szegedy, 2015] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *In Proceedings of The 32nd International Conference on Machine Learning,ICML*, pages 448–456, 2015.

[Keskar *et al.*, 2017] Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. On large-batch training for deep learning: Generalization gap and sharp minima. *In ICLR'2017 arxiv: 1609.04836*, 2017.

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

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

[Li *et al.*, 2018] Hao Li, Zheng Xu, Gavin Taylor, Christoph Studer, and Tom Goldstein. Visualizing the loss landscape of neural nets. *arxiv preprint arXiv:1712.0991*, 2018.

[McAllesterl, 1998] David A. McAllesterl. Some pac-bayesian theorems. *In Proceedings of the eleventh annual conference on Computational learning theory*, pages 230–234, 1998.

[McAllesterl, 1999] David A. McAllesterl. Pac-bayesian model averaging. *In Proceedings of the eleventh annual conference on Computational learning theory*, pages 164–170, 1999.

[McAllesterl, 2003] David A. McAllesterl. Simplified pac-bayesian margin bounds. *Lecture notes in computer science*, pages 203–215, 2003.

[Neyshabur *et al.*, 2017] Behnam Neyshabur, Srinadh Bhojanapalli, David McAllester, and Nathan Srebro. Exploring generalization in deep learning. *arXiv preprint arXiv:1706.08947*, 2017.

[Pierre-Antoine and Gallivan, 2006] Absil Pierre-Antoine and Kyle Gallivan. Joint diagonalization on the oblique manifold for independent component analysis. *ICASSP*, 2006.

[Pierre-Antoine *et al.*, 2009] Absil Pierre-Antoine, Robert. Mahony, and Rodolphe Sepulchre. *Optimization algorithms on matrix manifolds*. Princeton Press, 2009.

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

[Wang *et al.*, 2018] Huan Wang, Nitish Shirish Keskar, Caiming Xiong, and Richard Socher. Identifying generalization properties in neural network. *arXiv preprint arXiv:1809.07402*, 2018.

[Wen *et al.*, 2018] Wei Wen, Yandan Wang, Feng Yan, Cong Xu, Chunpeng Wu, Yiran Chen, and Hai Lii. Smoothout: Smoothing out sharp minima to improve generalization in deep learning. *arXiv preprint arXiv:1805.07898*, 2018.

[Wu *et al.*, 2018] Lei Wu, Chao Ma, and Weinan E. How sgd selects the global minima in over-parameterized learning: A dynamical stability perspective. *Advances in Neural Information Processing Systems 31 (NeurlPS 2018)*, 2018.

[Yuan *et al.*, 2018] Xiaoyong Yuan, Zheng Feng, Matthew Norton, and Xiaolin Lii. Generalized batch normalization: Towards accelerating deep neural networks. *arXiv preprint arXiv:1812.03271*, 2018.

[Zhang *et al.*, 2016] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. *In ICLR'2017 arXiv:1611.03530*, 2016.