Generalized Batch Normalization: Towards Accelerating Deep Neural Networks

Xiaoyong Yuan[*]
University of Florida
chbrian@ufl.edu

Zheng Feng[*]
University of Florida
fengzheng@ufl.edu

Matthew Norton
Naval Postgraduate School
mnorton@nps.edu

Xiaolin Li
University of Florida
andyli@ece.ufl.edu

Abstract

Utilizing recently introduced concepts from statistics and quantitative risk management, we present a general variant of Batch Normalization (BN) that offers accelerated convergence of Neural Network training compared to conventional BN. In general, we show that mean and standard deviation are not always the most appropriate choice for the centering and scaling procedure within the BN transformation, particularly if ReLU follows the normalization step. We present a Generalized Batch Normalization (GBN) transformation, which can utilize a variety of alternative *deviation measures* for scaling and *statistics* for centering, choices which naturally arise from the theory of *generalized deviation measures* and risk theory in general. When used in conjunction with the ReLU non-linearity, the underlying risk theory suggests natural, arguably optimal choices for the deviation measure and statistic. Utilizing the suggested deviation measure and statistic, we show experimentally that training is accelerated more so than with conventional BN, often with improved error rate as well. Overall, we propose a more flexible BN transformation supported by a complimentary theoretical framework that can potentially guide design choices.

1 Introduction

Training a deep neural network has traditionally been a difficult task. Issues such as the vanishing and exploding gradient, see e.g., (Pascanu, Mikolov, and Bengio 2013), make the use of gradient based optimization techniques difficult from the perspective of stability and fast convergence. However, new, seemingly simple tools have emerged to help practitioners overcome common pitfalls of neural network training. Two prominent examples are the use of Batch Normalization (BN) and Rectified Linear Units (ReLU).

Originally proposed by (Ioffe and Szegedy 2015), BN provides a simple transformation which incentivizes the homogenization of neural network layer outputs, so as to have the same scale and mean, eliminating what is referred to as internal covariate shift. Intuitively, this allows the 'signal' flowing through the neural network to maintain a consistent center and scale, potentially stabilizing gradients and the training procedure as a whole.

Consider a single layer of the network which first receives output from the previous layer h and then applies an affine transformation to get $\mathbf{x} = Wh + b$, followed by an element-wise non-linearity to produce output $g(\mathbf{x})$ which is fed to the next layer. Let $\mathbf{x} = [x_1, x_2, \cdots, x_n]^T$ denote its individual components so that we can write $g(\mathbf{x}) = [g(x_1), g(x_2), ..., g(x_n)]^T$.

The BN transformation is based upon the following transformation on each dimension j of the input,

$$\hat{x}_j \leftarrow \frac{x_j - E[x_j]}{\sqrt{E[(x_j - E[x_j])^2]}} \ ,$$

where $E[x_j]$ and $\sqrt{E[(x_j - E[x_j])^2]}$ are the mean and standard deviation of the random variable x_j, which are estimated during training with a batch of training examples.

In this paper, we begin by asking the question: Are mean and standard deviation the right choice for every network architecture? This simple question leads us to the main contribution of this paper, which is the observation that batch normalization can naturally be generalized and improved by considering the general transformation,

$$\hat{x}_j \leftarrow \frac{x_j - \mathcal{S}(x_j)}{\mathcal{D}(x_j)} \ ,$$

where $\mathcal{D}$ is some measure of *deviation*, not necessarily the standard deviation, and where $\mathcal{S}$ is a *statistic* which is not necessarily the mean. While arising from a specific set of axioms in risk theory, one can think of $\mathcal{D}$ as a general measure of the non-constancy of x and $\mathcal{S}$ as a type of 'center.'

We show that there exist many different choices for $\mathcal{D}$ and $\mathcal{S}$ besides standard deviation and mean, and that by formulating the batch normalization transformation with these alternatives one can accelerate neural network training compared to conventional BN and, in some settings, obtain improved predictive performance. Additionally, we show how the choice of $\mathcal{D}$ and $\mathcal{S}$ are driven not only by straightforward intuition, but also by recently developed theoretical tools from statistics and risk theory. Specifically, the theory of *generalized deviation measures* provides us with a wealth of choices for *deviation measure* $\mathcal{D}$, which includes standard deviation as a special case. In addition, for any choice of $\mathcal{D}$, there is a naturally corresponding *statistic* $\mathcal{S}$. Thus, choosing $\mathcal{D}$ implies natural choices for $\mathcal{S}$ and vice versa.

[*] Equal Contribution

Besides the simple observation that mean and standard deviation can be replaced by alternatives, our analysis is also driven by the observation that the appropriateness of the choice of $\mathcal{D}$ and $\mathcal{S}$ is directly tied to the choice of non-linearity which follows the normalization transformation. We focus our analysis on the ReLU non-linearity from (Glorot, Bordes, and Bengio 2011) and (Nair and Hinton 2010), which has played a significant role in stabilizing and accelerating neural network training (Krizhevsky, Sutskever, and Hinton 2012; Dahl, Sainath, and Hinton 2013). We show that mean and standard deviation are not natural choices for centering and scaling if ReLU follows the normalization transformation. Risk theory and simple intuition suggest more natural choices. In fact, we see that one of these choices, the superquantile deviation, allows explicit control over the level of sparsity of activation's; hypothesized to be an important property of ReLU (Glorot, Bordes, and Bengio 2011). While we focus on ReLU, this intuition can also be applied to any asymmetric non-linearity such as Leaky ReLU (Maas, Hannun, and Ng 2013), Exponential Linear Unit (Clevert, Unterthiner, and Hochreiter 2015), or any other arising from the ReLU family (e.g. (He et al. 2015)).

We demonstrate on MNIST, CIFAR-10, CIFAR-100, and SVHN datasets that the speed of convergence of stochastic gradient descent (SGD) can be increased by simply choosing a different $\mathcal{D}$ and $\mathcal{S}$ and that, in some settings, we obtain improved predictive performance. Our experimental analysis also serves to support the intuition that ReLU paired with $\mathcal{D} = \sqrt{E[(x - E[x])^2]}$ and $\mathcal{S} = E[x]$ is a mismatch and that asymmetric choices for $\mathcal{D}$ and $\mathcal{S}$ which are suggested by risk theory and intuition do, in fact, work better.

Although much further analysis is needed in this direction, we show that the use of ReLU's in tandem with BN can be tied directly to risk theory via a recently introduced concept called Buffered Probability of Exceedance (bPOE). Specifically, the use of normalization followed by a ReLU gives rise to what can be considered to be the tightest convex approximation to the $0 - 1$ loss. This is intriguing given the history of neural networks began with the concept of $0 - 1$ loss (indicator function) neural output which were then approximated with the sigmoid transformation as a differentiable surrogate (see e.g. (Rosenblatt 1958; McCulloch and Pitts 1943)).

2 Batch Normalization

The BN transformation is based upon the following transformation on each dimension j of the input,

$$\hat{x}_j \leftarrow \frac{x_j - \mu_j}{\sqrt{\sigma_j^2 + \epsilon}} ,$$

where σ_j and μ_j are the empirical standard deviation and mean of the random variable x_j, which are estimated during training with a batch of training examples. Throughout this paper, we will view $\mathbf{x}$ as a random vector which is observed empirically via the training batches. Thus, during training, $\mu_j = \frac{1}{|B|} \sum_{i=1}^{|B|} x_j^{(i)}$ with $|B|$ denoting the size of the training batch.

The BN procedure follows the actual normalization with the following linear transformation, where γ_j, β_j are parameters which will be tuned during training,

$$\gamma_j \hat{x}_j + \beta_j \ .$$

The BN procedure is then followed by the final non-linear transformation $g(\gamma_j \hat{x}_j + \beta_j)$. Why is this linear transformation needed? As noted by (Ioffe and Szegedy 2015), the BN transformation may not be appropriate or work well in conjunction with the non-linear transformation g that follows. Thus, the authors introduced a way to adjust the BN transformation if necessary. However, there is no guarantee that training will find the right linear transformation and be able to properly counteract a poor choice of scale and center. In some sense, this is why it is argued in (Mishkin and Matas 2015) that proper initialization is all that is needed. Assuming that the centering and scaling are not correct, which is to say that the trainable linear transformation is necessary to adjust the center and scale, then BN can be loosley viewed as a type of data dependent initialization strategy. In this sense, the additional linear transformation can be used within our proposed scheme in exactly the same way, but with more control over the initialization where one would hope to select a more appropriate data-dependent centering and scaling factor.

Cases where the standard BN may not work well in conjunction with the non-linearity g can be easily illustrated, particularly if g is the ReLU non-linearity. Consider a set of outputs $\{x_1, \cdots, x_N\}$ from a network layer which have mean zero, i.e., $\frac{1}{N} \sum_{i=1}^{N} x_i = 0$, with ordering $x_1 < \cdots < x_k < 0 < x_{k+1} < \cdots < x_N$. Assume that we are then going to divide by some normalization factor, such as standard deviation, and then feed these values into a ReLU non-linearity $\max\{0, x_i\}$. The ReLU non-linearity will map points $x_1, \cdots, x_k$ to zero. Considering this fact, does it make sense to first divide the whole set of N points by the standard deviation? Intuitively, it would make more sense to divide by the variance of only the set of points $\{x_{k+1}, \cdots, x_N\}$. The variation of the set of points $\{x_1, \cdots, x_k\}$ is irrelevant given the fact that a ReLU will follow, sending all of these points to zero. This consideration is particularly important if the conditional distributions $\{x_1, \cdots, x_k\}$ and $\{x_{k+1}, \cdots, x_N\}$ exhibit very different scales and variation. In this case, it may be more appropriate to use a one-sided measure of deviation $\mathcal{D}$ for the normalization step such as the Right Semi-Deviation (RSD) $\frac{1}{N} \sum_{i=1}^{N} \max\{0, x_i\}$. Furthermore, a similar argument can be applied to the centering operation. Assume, for instance, that the distribution $\{x_1, \cdots, x_N\}$ is heavy-tailed, with $\{x_1, \cdots, x_{N-1}\}$ having mean zero and variance 1, but with $x_N = 100$. The mean of all N points will be very large, and centering the data via mean subtraction will yield $x_N - \mu$ as the only term with value larger than zero. Thus, the application of the ReLU will leave only one sample as having non-zero value (and gradient), with much of the valuable learning signal lost because of poor choice of centering statistic $\mathcal{S}$.

This paper shows that there are other ways to perform batch normalization, potentially avoiding the need to adjust the normalization with the affine transformation (or at least reducing the amount by which it would need to be adjusted), offering accelerated convergence. Generalizations and variants of BN have been proposed before. For example, Klambauer et al. (2017) proposed a self-normalizing network layer, but is limited to standard feed-forward architectures. Ba, Kiros, and Hinton (2016) altered BN to work with recurrent neural networks. Mishkin and Matas (2015) argue that BN is simply another way to perform initialization, thus proposing initialization methods that produce similar effects. The idea of BN was altered to weight normalization by reparameterizing the weights (Salimans and Kingma 2016; Chunjie, Qiang, and others 2017). Our proposed approach, while relying on simple principles, is grounded in a broader theory and maintains all important flexibility of conventional BN.

3 Asymmetric Deviation Measures in Risk Theory

As alluded to in the introduction, it is easy to question the use of variance as the scale normalizing factor if it is followed by the ReLU transformation. This gives rise to the obvious question: What other options do we have that may be more appropriate? We find, in general, that risk theory provides us with an entire class of generalized deviation measures to choose from. In this section, we briefly introduce risk theory before discussing generalized deviation measures in Section 4 where we introduce the GBN transformation and show that generalized deviation measures provide us with an array of alternatives to mean and standard deviation.

Over the past 25 years, risk management theory has played a crucial role in the development of fundamental statistical concepts that not only measure risk (Artzner et al. 1999; Föllmer and Schied 2002; Szegö 2002), but have proven fundamental to statistical theory and optimization under uncertainty. A full review of risk theory is beyond the scope of this paper, but a simple example in the context of financial engineering can be used to illustrate. Consider an investment which will yield a loss of x, with x being a random monetary loss. Assume we knew the distribution of x, and we were to ask: How *risky* is this investment? How can we measure *risk* to compare it against other investments y? An obvious choice would be to look at the expected loss $E[x]$. However, this may be inappropriate, as investor objectives (or distribution of x) may be highly asymmetric. It may be more appropriate to measure risk with an asymmetric quantity. One example would be to use the quantile $q_\alpha(x) = \min\{z | P(x \leq z) \geq \alpha\}$, where $\alpha \in [0, 1]$ is a probability level. Its inverse, called Probability of Exceedance (POE), given by $P(x > z)$ where $z \in \mathbb{R}$ is some known threshold, may also be desirable if some threshold z is known and exceeding such a threshold is undesirable.

One of the primary drivers of risk theory, however, has been the need to quantify risk in such a way that optimization can take place (e.g. finding the portfolio with minimal risk). The quantile, also called the Value-at-Risk, and POE are numerically troublesome in this context. Specifically, these functions often prove to be non-convex and discontinuous, essentially reducing to sums of indicator ($0 - 1$ loss) functions. From this difficulty, more amenable alternatives have arisen.

Two popular alternatives that are relevant to our discussion are the superquantile and Buffered Probability of Exceedance (bPOE) (Rockafellar and Uryasev 2000; Rockafellar and Uryasev 2002; Acerbi and Tasche 2002; Mafusalov and Uryasev 2015). The superquantile is a measure of uncertainty similar to the quantile, but with superior mathematical properties. Formally, the superquantile, also called Conditional Value-at-Risk (CVaR) in the financial engineering literature, for a continuously distributed x is defined as

$$\bar{q}_\alpha(x) = E\left[x | x > q_\alpha(x)\right].$$

For general distributions, the superquantile can be defined by the following formula,

$$\bar{q}_\alpha(x) = \min_\gamma \gamma + \frac{E[x-\gamma]^+}{1-\alpha} , \tag{1}$$

where $[\cdot]^+ = \max\{\cdot, 0\}$.

Similar to $q_\alpha(x)$, the superquantile can be used to assess the tail of the distribution. The superquantile, though, is far easier to handle in optimization contexts. It also has the important property that it considers the magnitude of events within the tail. Therefore, in situations where a distribution may have a heavy tail, the superquantile accounts for magnitudes of low-probability large-loss tail events while the quantile does not account for this information.

bPOE is the inverse of the superquantile. In other words, bPOE calculates one minus the probability level at which the superquantile equals a specified threshold z. It is calculated by the formula

$$\bar{p}_z(x) = \min_{a \geq 0} E[a(x-z)+1]^+ = \min_{\gamma < z} \frac{E[x-\gamma]^+}{z-\gamma},$$

where $[\cdot]^+ = \max\{\cdot, 0\}$. In addition, we have the following formula which will be important for our case. Assuming that $\bar{p}_z(x) = 1 - \alpha$, we have that

$$\bar{p}_z(x) = \frac{E[x - q_\alpha(x)]^+}{\bar{q}_\alpha(x) - q_\alpha(x)} .$$

Roughly speaking, bPOE calculates the proportion of worst case outcomes which average to z.

As it relates to POE, bPOE can be viewed as an optimal convex approximation. More specifically, among law-invariant functions of x, $\bar{p}_z(x)$ is the minimal (tightest) quasi-convex upper bound of $P(x > z) = E[I(x > z)]$.

These ideas, though, have not been limited to risk management and finance. Machine Learning has also been impacted by this theory. For example, new support vector classifiers have been generated with superquantile and bPOE concepts (Takeda and Sugiyama 2008; Norton, Mafusalov, and Uryasev 2017; Gotoh and Uryasev 2017) and sequential decision problems are being formulated with risk in mind (Galichet, Sebag, and Teytaud 2013; Chow and Ghavamzadeh 2014).

4 Generalized Batch Normalization

In this paper, we define Generalized Batch Normalization (GBN) to be identical to conventional BN but with standard deviation replaced by a more general deviation measure $\mathcal{D}(x)$ and the mean replaced by a corresponding statistic $\mathcal{S}(x)$. In other words, we have the transformation,

$$\hat{x}_j \leftarrow \frac{x_j - \mathcal{S}(x_j)}{\mathcal{D}(x_j)} \ .$$

Here, each choice of $\mathcal{D}$ is naturally paired with some $\mathcal{S}$, which we discuss in the following section. In Section 5, we implement a suite of these new measures and test them on the MNIST, CIFAR-10, CIFAR-100, and SVHN datasets, showing that convergence can be accelerated, and sometime accuracy improved, by use of different deviation measures and statistics.

4.1 Generalized Deviation Measures and Statistics

In (Rockafellar, Uryasev, and Zabarankin 2006), the concept of a generalized deviation measure was introduced to broaden the statistical view of *deviation* beyond the single case of standard deviation, specifically for use in quantitative risk analysis. These deviation measures follow a very general set of axioms which we will not delve into here. However, some examples can be found in Table 1, and they can be understood intuitively as follows: Deviation measures quantify the *non-constancy* of a random variable. As seen in Table 1, standard deviation is only one of many possibilities, such as the asymmetric deviation measures RSD and SQD with $\alpha > 0$. These measures of deviation look at the variation only in the *right-tail* of the distribution of x. It's easy to see how this type of asymmetric measure would be of interest in finance, where it may be important to analyze the variation of only the largest losses within the *right-tail*.

The theory of generalized deviation measures is also complemented by the recently introduced theory of the Risk Quadrangle. Utilizing functional relationships that are beyond the scope of this paper, (Rockafellar and Uryasev 2013) shows that measures of deviation are intimately related to similar measures of risk, regret, and error. Furthermore, associated with any measure of deviation is a unique *statistic*. In short, however, without getting into too much detail, one can think of the statistic as a type of 'center.' In Table 1, we see how this intuition plays out, with the corresponding statistics listed in the right column. For SD, MAD, and RSD, we see that $\mathcal{S}(x)$ is simply the expectation. However, for SQD with $\alpha = .5$, we see that $\mathcal{S}(x) = q_{.5}(x)$ the median, certainly a different notion of the 'center.' Furthermore, we see that for RBD, the statistic is the center of the range. However, for SQD it is important to notice that we can achieve very different statistics by moving α, which gives us different quantiles.

4.2 Choosing $\mathcal{D}$ or $\mathcal{S}$: General Intuition

Now that we are given more options for deviation measures and statistics, we can begin to think about the benefits and drawbacks of each within the neural network architecture and the GBN transformation. Utilizing standard deviation seems like an intuitive choice. However, this depends heavily on the shape of the (empirical) distribution of x. If the distribution is relatively symmetric, then standard deviation will be indicative of the overall scale and the mean will be indicative of the 'center'. Similarly, this may hold true if the distribution does not have heavy tails or outliers on one side or the other. However, if the distribution of x has e.g. heavy tails, is highly asymmetric, has outliers, or is multimodal; then the mean may be a poor choice for the 'center' and the deviation of values to the right of the mean may be dramatically different than the deviation of values to the left of the mean. In this case, a quantile may be a more appropriate notion of the 'center.' Choosing, for example, the median instead of the mean assures that we are truly 'centering' the data, with half of the points on the 'left' and half on the 'right.'

Even if the distribution of x is not asymmetric or heavy tailed, the choice of center is particularly important if normalization is followed by the ReLU activation. Specifically, the choice of center controls the sparsity of activation's produced by the ReLU, since any elements left-of-center will be sent to zero. ReLU induced sparsity has been hypothesized as critical to its success (Glorot, Bordes, and Bengio 2011). In this case, the quantile is a natural choice for center that provides precise control over such sparsity. If the normalization centers w.r.t. the quantile at α, exactly $\alpha\%$ of activation's across the batch will have zero value.

Driving our intuition from the beginning was the idea that the non-linearity, deviation measure, and statistic should be chosen in tandem. As mentioned in Section 2, the pairing of ReLU with typical BN (i.e. standard deviation and mean normalization) does not seem appropriate given the fact that standard deviation is symmetric while ReLU is asymmetric. Thus, in light of Section 2, we find that asymmetric deviation measures are more appropriate such as RSD or SQD for any $\alpha > 0$. In Section 5, we see this intuition confirmed, with RSD and SQD outperforming SD in terms of convergence rate and, often times, test error. Although not explored in our experiments, this intuition applies to any asymmetric nonlinearity such as the Leaky ReLU (Maas, Hannun, and Ng 2013), Exponential Linear Unit (Clevert, Unterthiner, and Hochreiter 2015), or any other arising from the ReLU family (e.g. (He et al. 2015)).

4.3 An Optimal Choice

Beyond this simple intuition, we can utilize connections to risk theory to provide evidence that the ReLU should be used in tandem with an asymmetric deviation measure. Specifically, we show that the use of SQD and RSD followed by ReLU is approximately equivalent to a probabilistic transformation which mimics an optimal quasiconvex approximation to the $0-1$ (indicator) loss function.

Intuitively, ReLU's should be paired with an asymmetric measure of deviation, with candidates including RSD and SQD. However, a natural choice arises when looking at the similarities between bPOE and the combination of the GBN transformation and ReLU non-linearity. Consider a GBN transformation followed by a ReLU non-linearity.

	Deviation Measure $\mathcal{D}(x)$	Statistic $\mathcal{S}(x)$
Standard Deviation (SD)	$\sqrt{E[(x-E[x])^2]}$	$E[x]$
Mean Absolute Deviation (MAD)	$E[\lvert x-E[x]\rvert]$	$E[x]$
Right-Semi-Deviation (RSD)	$E[x-E[x]]^+$	$E[x]$
Superquantile Deviations (SQD) for $\alpha \in (0,1)$	$\bar{q}_\alpha(x-E[x])$	$q_\alpha(x)$
Range-Based Deviation (RBD)	$\sup x - \inf x$	$\frac{1}{2}(\sup x + \inf x)$
Worst-Case Deviation (WCD)	$\sup x - E[x]$	$\sup x$

Table 1: Examples of deviation measures and their corresponding statistics.$[x]^+ = \max\{0, x\}$

Now, for the GBN transformation let us choose SQD deviation measure $\mathcal{D}(x_j) = \bar{q}_\alpha(x_j - \mu_j)$ where α is chosen so that $q_\alpha(x_j) = \mu_j$, meaning that we are choosing the probability level on which the mean sits. This gives us the following transformation, where the superscript denotes the i^{th} sample from a batch:

$$\hat{x}_j^{(i)} \leftarrow \left[\frac{x_j^{(i)} - q_\alpha(x_j)}{\bar{q}_\alpha(x_j) - q_\alpha(x_j)}\right]^+ .$$

This can be re-written as,

$$\hat{x}_j^{(i)} \leftarrow \left[\frac{x_j^{(i)} - \mu_j}{E[x_j - \mu_j | x_j > \mu_j]}\right]^+ .$$

One will immediately notice that this is almost identical to a conventional BN transformation followed by ReLU with the only difference being that we are dividing by a one-sided semi-deviation rather than the two-sided standard deviation. One will notice, however, the following connection to bPOE:

$$\bar{p}_z(x_j) = E\left[\frac{x_j - q_\alpha(x_j)}{\bar{q}_\alpha(x_j) - q_\alpha(x_j)}\right]^+$$

for threshold $z = \bar{q}_\alpha(x_j)$. Thus, we see that the combination of GBN and ReLU yields a transformation based upon bPOE. If also divided by sample size N, each individual sample $x_j^{(i)}$ will yield output $\frac{1}{N}\left[\frac{x_j^{(i)} - q_\alpha(x_j)}{\bar{q}_\alpha(x_j) - q_\alpha(x_j)}\right]^+ \in (0,1)$ with the sum,

$$\frac{1}{N}\left[\frac{x_j^{(i)} - q_\alpha(x_j)}{\bar{q}_\alpha(x_j) - q_\alpha(x_j)}\right]^+ = \bar{\bar{p}}_z(x_j) ,$$

where $\bar{\bar{p}}_z(x_j)$ simply denotes the empirical bPOE calculated from a sample. This means that the overall output distribution will consist of values in the range $[0,1]$ with non-zero items being those that are in the bPOE-tail of the empirical distribution of x_j.

Thus, by combining GBN and ReLU we are effectively performing a probabilistic transformation, with the transformation mimicking the optimal quasiconvex approximation to the $0-1$ loss.

5 Experimental Evaluation

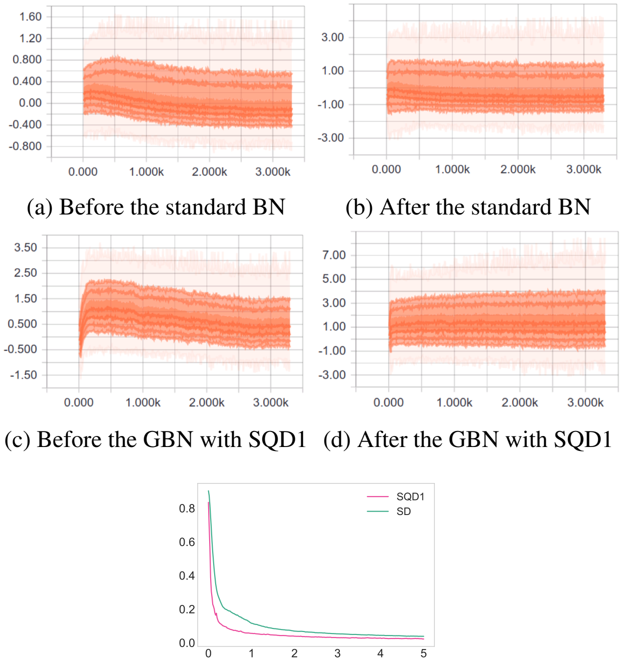
Overall, the first goal of our experiments is to demonstrate the obvious: All other things being equal, different normalization methods (i.e. different choices for deviation measure and statistic) lead to different network properties. We then explore the specifics of these changes. First, we show that convergence rate and stability of NN training via SGD can often be improved by utilizing alternative deviation measures. Improvement is measured relative to conventional BN, which uses mean and standard deviation as its statistic and deviation measure. Overall, we find that SQD, MAD, and RSD often lead to increased convergence rates and, sometimes, increased stability in terms of smoothly decreasing test error during SGD. Second, we see that these alternative choices often lead to testing error that is nearly as good as, or better, than that achieved by standard BN.

For all experiments, GBN is implemented in exactly the same manner as standard BN, only with mean and variance replaced by generalized $\mathcal{S}$ and $\mathcal{D}$ within the batch normalization transformation. This includes appropriate inclusion of the chosen deviation measure and statistic within the gradient calculation as well as the batch-based estimation of $\mathcal{D}(x_j)$ and $\mathcal{S}(x_j)$ during training and population-based estimation for inference. This also includes the additional linear transformation which typically follows the normalization step, before application of non-linearity. See (Ioffe and Szegedy 2015) for specifics.

We performed experiments on MNIST, CIFAR-10, CIFAR-100, and SVHN datasets. We compared the performance of GBN transformations with 7 different deviation measures and statistics, including the conventional mean and standard deviation. As indicated in Table 1, we utilized standard SD along with MAD, RSD, RBD, and SQD with $\alpha = .25, .5,$ and $.75$ which we denote by SQD1, SQD2, and SQD3 respectively. We omit WCD since centering w.r.t. $\sup x$ is obviously a poor choice when paired with ReLU. Subtracting $\sup x$ would make all points less than or equal to zero and the ReLU would send them all then to zero, producing an untrainable network without activations.

5.1 MNIST

GBN transformation over time To illustrate the effect that an asymmetric deviation measure and statistic have on the distribution of network activations when paired with ReLU, we observe the predictive error rate and the distribution over one feature before and after the GBN transformation. We conduct classification on MNIST (LeCun et al. 1998) with neural network architecture LeNet with the input size of 28x28 and two convolutional layers with kernel size 5, and number of filters 20 and 50 respectively. The batch normalization is added after each of the convolutional layers and then followed by a ReLU non-linearity. The comparison



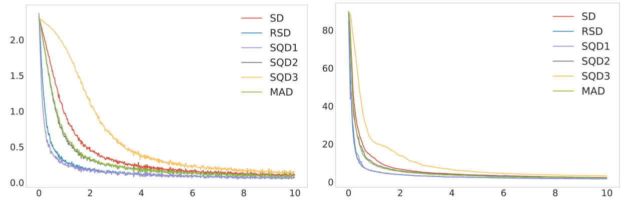
(a) Before the standard BN (b) After the standard BN

(c) Before the GBN with SQD1 (d) After the GBN with SQD1

(e) Error rate on held out test set

Figure 1: (a, b) The distribution evolution of standard BN on a selected feature along with the iteration steps. (c, d) The same figure over the GBD with SQD1 deviation measure. (e) The error rate of two settings on the held out test set. GBN with SQD1 help the network converges faster and also achieves better error rate that standard BN.

is performed on standard BN and GBN with deviation measure SQD1, which has statistic equal to the $\alpha = .25$ quantile. We choose to observe one feature pixel of the second convolutional layer's feature map. Figure 1(a,b) shows this feature's distribution density before and after standard BN. Figure 1(c,d) shows the same feature's distribution density before and after applying GBN with SQD1. All the distributions before batch normalization exhibit significant change in terms of mean and variance. Both of the two normalization approaches removed the covariate shift effect and output a stabilized distribution over time. And after GBN with the deviation measure of SQD1, most of the values appear larger than 0 compared to the symmetric distribution of standard BN having the mean of 0. As one would expect, centering w.r.t. the $\alpha = .25$ quantile forces $\alpha\%$ of the activations to be less than zero before applying the non-linearity. In Figure 1(e), this consistent asymmetric distribution of the GBN's output helps it achieve faster convergence rate and better error rate compared to the standard BN.



(a) Training Loss (b) Test Error Rate(%)

Figure 2: Performance comparison of the MNIST classification with LeNet. X-axis: Number of epochs; Y-axis: Training loss/Test error rate.

GBN performance on MNIST To compare the performance of various deviation measures and statistics on MNIST, we use the same experimental setting of neural network above with vanilla SGD as the optimizer, with learning rate equal to .01, and batch size equal to 1000. Figure 2 shows the error rate of 6 different choices for deviation measure and statistic. All settings are evaluated on the training loss and test error rate. We see that GBN with SQD1, RSD, SQD2, and MAD all perform better than standard BN in terms of converge rate and test error rate. And GBN with deviation measures of SQD1 and RSD converge remarkably faster than others.

5.2 CIFAR-10, CIFAR-100, and SVHN

We compare the performance and convergence rate on the CIFAR-10, CIFAR-100, and Street View House Numbers (SVHN) datasets. The CIFAR-10 and CIFAR-100 dataset consist of 60,000 tiny color images (32x32) with 10 and 100 classes respectively for image recognition task (Krizhevsky and Hinton 2009). The SVHN Dataset consists of Google Street View images with 10 house digit classes (Netzer et al. 2011).

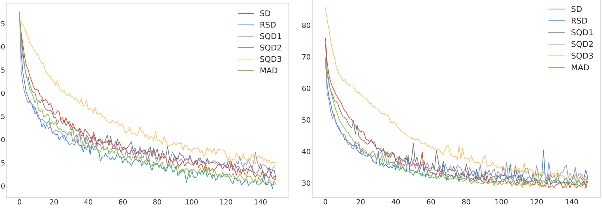
We trained LeNet networks on the CIFAR-10 (200 epochs) and the SVHN (200 epochs) datasets. The setting is set similar to that used with MNIST dataset: SGD with learning rate 0.1 and 0.01, batch size 1024. Figure 3 and Figure 4 illustrate the performance comparison of six different choices of deviation measure on the CIFAR-10 and SVHN datasets respectively.

We also train a ResNet architecture with 20 layers (exactly the same architecture and settings used in (He et al. 2015)) for 200 epochs on the CIFAR-10 and CIFAR-100 dataset. We trained the ResNet with and without data augmentation (i.e., random crop and random horizontal flip). For the CIFAR-10 dataset, we observe that with data augmentation, the proposed methods achieve the similar performance as standard BN. However, if we do not augment data (less symmetric distribution), both RSD and MAD perform better than standard BN (Figure 5). For the CIFAR-100 dataset, even with data augmentation, RSD and MAN outperform standard BN (Figure 6).

Most methods (MAD, SQD1, SQD2, and RSD) converge faster than standard SD. The error rates of these methods are similar, but with slight improvement achieved by the proposed alternative deviation measures. Table 2 contains more detailed results, including the best error rate for a few extra settings. However, faster converging alternatives achieve similar error rates. Here, we see that not only can faster convergence be achieved with alternative deviation measures, but increased accuracy as well.

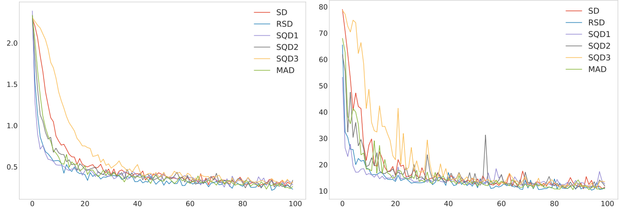
Dataset	Architecture	Learning Rate	Batch Size	SD	MAD	RSD	RBD	SQD1	SQD2	SQD3
CIFAR-10	LeNet	0.1	256	30.73	29.70	30.04	30.38	33.29	29.53	**29.07**
CIFAR-10	LeNet	0.1	1024	29.31	28.58	28.85	31.05	33.33	28.63	**28.03**
CIFAR-10	LeNet	0.1	2048	28.96	29.32	**27.51**	36.28	39.94	28.79	29.18
CIFAR-10	LeNet	0.01	256	29.69	29.03	29.90	33.32	29.99	27.96	**27.36**
CIFAR-10	LeNet	0.01	1024	**28.29**	28.37	29.44	47.51	30.62	29.33	30.97
CIFAR-10	LeNet	0.01	2048	31.64	**30.53**	30.81	57.08	32.45	31.35	37.26
CIFAR-10	ResNet20	0.1	1024	29.67	29.33	30.55	**20.49**	23.64	24.86	24.99
CIFAR-10	ResNet20	0.01	1024	34.57	33.59	35.52	47.17	32.20	**32.02**	47.89
SVHN	LeNet	0.1	1024	10.68	10.17	10.91	12.17	80.41	10.60	**10.08**
SVHN	LeNet	0.1	2048	10.46	10.18	10.57	13.58	22.03	10.68	**9.75**
SVHN	LeNet	0.01	1024	10.83	10.46	**10.30**	20.62	11.62	10.87	11.29
SVHN	LeNet	0.01	2048	12.60	11.74	**11.22**	43.04	11.99	11.83	12.80

Table 2: Performance Comparison of Test Error Rate (%) on the CIFAR-10 and SVHN datasets.



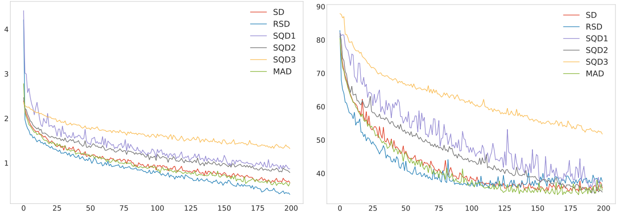
(a) Training Loss (b) Test Error Rate

Figure 3: Performance comparison on the CIFAR-10 dataset using LeNet. X-axis: Number of epochs; Y-axis: Training loss/Test error rate (%).



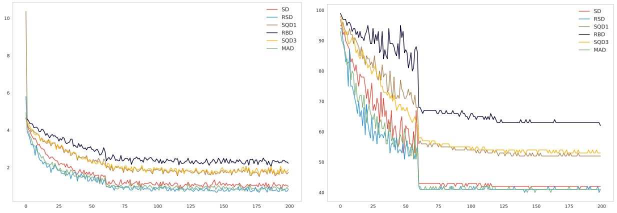
(a) Training Loss (b) Test Error Rate

Figure 4: Performance comparison on the SVHN dataset. X-axis: Number of epochs; Y-axis: Training loss/Test error rate (%).



(a) Training Loss (b) Test Error Rate

Figure 5: Performance comparison on the CIFAR-10 dataset using ResNet20. X-axis: Number of epochs; Y-axis: Training loss/Test error rate (%).



(a) Training Loss (b) Test Error Rate

Figure 6: Performance comparison on the CIFAR-100 dataset using ResNet20. X-axis: Number of epochs; Y-axis: Training loss/Test error rate (%).

5.3 Discussion

When choosing $\mathcal{S}$ and $\mathcal{D}$, it is important to consider their estimation properties. For example, it is well-known that empirical estimates of the mean are more stable, and converge more quickly to the true mean, than empirical estimates of the superquantile. This also applies to SD and one-sided deviation measures like RSD. Clearly, since only one side of the distribution is involved, more samples will be needed for accurate, low variance estimation of asymmetric (one-sided) deviation measures or statistics. Compared with small batch size, we observe that training with large batch size improves the convergence rate. However, this small-batch degradation is not a new consideration and has been observed with standard BN. (Ioffe 2017) discusses this issue and shows that there do exist techniques to help alleviate this affect for BN. Although we leave this discussion to future work, it would seem straightforward to apply the same techniques to GBN in general.

6 Conclusion

In this paper, we have proposed a generalized variant of batch normalization which can be used to improve the convergence rate and, often, the error rate compared to vanilla batch normalization. As a generalization, we show that there are many other natural choices for the scaling and centering factors which we pose as general deviation measures and statistics. We also show that conventional normalization is not optimal if followed by the ReLU non-linearity and we provide alternatives that are justified both intuitively and theoretically, showing also that these new methods increase convergence speed experimentally.

7 Acknowledgement

The authors would like to thank anonymous reviewers whose suggestions help improve the quality of the paper. This research was supported in part by National Science Foundation (CNS-1624782, CNS-1747783), National Institutes of Health (R01-GM110240), and Industrial Members of NSF Center for Big Learning (CBL).

References

Acerbi, C., and Tasche, D. 2002. On the coherence of expected shortfall. *Journal of Banking & Finance* 26(7):1487–1503.

Artzner, P.; Delbaen, F.; Eber, J.-M.; and Heath, D. 1999. Coherent measures of risk. *Mathematical finance* 9(3):203–228.

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

Chow, Y., and Ghavamzadeh, M. 2014. Algorithms for cvar optimization in mdps. In *NIPS*, 3509–3517.

Chunjie, L.; Qiang, Y.; et al. 2017. Cosine normalization: Using cosine similarity instead of dot product in neural networks. *arXiv preprint arXiv:1702.05870*.

Clevert, D.-A.; Unterthiner, T.; and Hochreiter, S. 2015. Fast and accurate deep network learning by exponential linear units (elus). *arXiv preprint arXiv:1511.07289*.

Dahl, G. E.; Sainath, T. N.; and Hinton, G. E. 2013. Improving deep neural networks for lvcsr using rectified linear units and dropout. In *ICASSP*, 8609–8613. IEEE.

Föllmer, H., and Schied, A. 2002. Convex measures of risk and trading constraints. *Finance and stochastics* 6(4):429–447.

Galichet, N.; Sebag, M.; and Teytaud, O. 2013. Exploration vs exploitation vs safety: Risk-aware multi-armed bandits. In *Asian Conference on Machine Learning*, 245–260.

Glorot, X.; Bordes, A.; and Bengio, Y. 2011. Deep sparse rectifier neural networks. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, 315–323.

Gotoh, J.-y., and Uryasev, S. 2017. Support vector machines based on convex risk functions and general norms. *Annals of Operations Research* 249(1-2):301–328.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2015. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *ICCV*, 1026–1034.

Ioffe, S., and Szegedy, C. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, 448–456.

Ioffe, S. 2017. Batch renormalization: Towards reducing minibatch dependence in batch-normalized models. In *NIPS*, 1942–1950.

Klambauer, G.; Unterthiner, T.; Mayr, A.; and Hochreiter, S. 2017. Self-normalizing neural networks. In *NIPS*, 972–981.

Krizhevsky, A., and Hinton, G. 2009. Learning multiple layers of features from tiny images. *Technical report, University of Toronto*.

Krizhevsky, A.; Sutskever, I.; and Hinton, G. E. 2012. Imagenet classification with deep convolutional neural networks. In *NIPS*, 1097–1105.

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.

Maas, A. L.; Hannun, A. Y.; and Ng, A. Y. 2013. Rectifier nonlinearities improve neural network acoustic models. In *Proc. icml*, volume 30, 3.

Mafusalov, A., and Uryasev, S. 2015. Buffered probability of exceedance: Mathematical properties and optimization algorithms. *Research Report 2014-1, ISE Dept., University of Florida*.

McCulloch, W. S., and Pitts, W. 1943. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics* 5(4):115–133.

Mishkin, D., and Matas, J. 2015. All you need is a good init. *arXiv preprint arXiv:1511.06422*.

Nair, V., and Hinton, G. E. 2010. Rectified linear units improve restricted boltzmann machines. In *ICML*, 807–814.

Netzer, Y.; Wang, T.; Coates, A.; Bissacco, A.; Wu, B.; and Ng, A. Y. 2011. Reading digits in natural images with unsupervised feature learning. In *NIPS workshop on deep learning and unsupervised feature learning*, volume 2011, 5.

Norton, M.; Mafusalov, A.; and Uryasev, S. 2017. Soft margin support vector classification as buffered probability minimization. *JMLR* 18(1):2285–2327.

Pascanu, R.; Mikolov, T.; and Bengio, Y. 2013. On the difficulty of training recurrent neural networks. In *ICML*.

Rockafellar, R., and Uryasev, S. 2000. Optimization of conditional value-at-risk. *The Journal of Risk, Vol. 2, No. 3, 2000, 21-41*.

Rockafellar, R. T., and Uryasev, S. 2002. Conditional value-at-risk for general loss distributions. *Journal of banking & finance* 26(7):1443–1471.

Rockafellar, R. T., and Uryasev, S. 2013. The fundamental risk quadrangle in risk management, optimization and statistical estimation. *Surveys in Operations Research and Management Science* 18(1):33–53.

Rockafellar, R. T.; Uryasev, S.; and Zabarankin, M. 2006. Generalized deviations in risk analysis. *Finance and Stochastics* 10(1):51–74.

Rosenblatt, F. 1958. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological review* 65(6):386.

Salimans, T., and Kingma, D. P. 2016. Weight normalization: A simple reparameterization to accelerate training of deep neural networks. In *NIPS*, 901–909.

Szegö, G. 2002. Measures of risk. *Journal of Banking & Finance* 26(7):1253–1272.

Takeda, A., and Sugiyama, M. 2008. ν-support vector machine as conditional value-at-risk minimization. In *ICML*, 1056–1063. ACM.