# Normalized Gradient with Adaptive Stepsize Method for Deep Neural Network Training

Adams Wei Yu[1], Qihang Lin[2], Ruslan Salakhutdinov[1], and Jaime Carbonell[1]

[1]School of Computer Science, Carnegie Mellon University
[2]Tippie College of Business, The University of Iowa

July 18, 2017

## Abstract

In this paper, we propose a generic and simple algorithmic framework for first order optimization. The framework essentially contains two consecutive steps in each iteration: 1) computing and normalizing the mini-batch stochastic gradient; 2) selecting adaptive step size to update the decision variable (parameter) towards the negative of the normalized gradient. We show that the proposed approach, when customized to the popular adaptive stepsize methods, such as AdaGrad, can enjoy a sublinear convergence rate, if the objective is convex. We also conduct extensive empirical studies on various non-convex neural network optimization problems, including multi layer perceptron, convolution neural networks and recurrent neural networks. The results indicate the normalized gradient with adaptive step size can help accelerate the training of neural networks. In particular, significant speedup can be observed if the networks are deep or the dependencies are long.

## 1 Introduction

Continuous optimization is a core technique for training non-convex sophisticated machine learning models such as deep neural networks [Bengio, 2009]. Compared to convex optimization where a global optimal solution is expected, non-convex optimization usually aims to find a stationary point or a local optimal solution of an objective function by iterative algorithms. Among a large volume of optimization algorithms, first-order methods, which only use (stochastic) gradient information of objective functions to update solution, are most widely used due to its relatively low requirement on memory space and computation time.

In this paper, we are particularly interested in solving the deep neural network training problems with first order method. However, compared to other non-convex problems, deep neural network training has the following challenges:

1. Gradient computation is expensive. A full gradient is usually difficult to compute, especially when applied to machine learning problems over big data, because its calculation requires going through the whole dataset.

2. Gradient may be vanishing and/or exploding. The gradient may become very small over some plateaus regions while sharply becoming large on the cliffs of objective function. As the number

of layers in the neural network increases, this phenomenon of vanishing or exploding gradients becomes more severe so that the iterative solution will converge very slowly or diverge quickly.

These two challenges indicate that for each iteration, stochastic gradient might be the best practical first order information we can get. Once we obtain this information, we should develop certain mechanism to avoid its exploding or vanishing phenomenon and eventually ensure faster convergence.

In this paper we aim to solve those challenges from the two key ingredients of first-order optimization algorithm per iteration, i.e., an updating direction and a step size (a.k.a. learning rate). The updating direction indicates where the next iterated point should move to and the step size determines how far the move should end up with.

We propose to use stochastic *normalized gradient* which is constructed by dividing the (stochastic) gradient by its norm. Compared to the regular gradient, normalized gradient only provides an updating direction but does not incorporate the local steepness of the objective through its magnitude, which help to control the change of the solution through a well-designed step length. Intuitively, as it constrains the magnitude of the gradient to be 1, it should to some extent prevent the gradient vanishing or exploding phenomenon. In fact, as showed in [Hazan et al., 2015, Levy, 2016], normalized gradient descent (NGD) methods are more numerically stable and have better theoretical convergence properties than the regular gradient descent method in non-convex optimization.

Once the updating direction is determined, step size is the next important component in the design of first-order methods. While for convex problems there are some well studied strategies to find a stepsize ensuring convergence, for non-convex optimization, the choice of step size is more difficult and critical as it may either enlarge or reduce the impact of the aforementioned vanishing or exploding gradients. When a normalized gradient is used, the step size fully controls the progress of the solution so that the convergence is possible only when the step size is carefully designed.

Among different choices of step sizes, the adaptive *feature-dependent* step size has attracted lots of attention due to its great performance when applied to training deep neural networks [Duchi et al., 2011, Kingma and Ba, 2014]. Different from the standard step-size rule which multiplies a same number to each coordinate of gradient, the adaptive feature-dependent step-size rule multiplies different numbers to coordinates of gradient so that different parameters in the learning model can be updated in different paces. For example, the adaptive step size invented by [Duchi et al., 2011] is constructed by aggregating each coordinates of historical gradients. As discussed by [Duchi et al., 2011], this method can dynamically incorporate the frequency of features in the step size so that frequently occurring features will have a small step size while infrequent features have a long step size. The similar adaptive step size is proposed in [Kingma and Ba, 2014] but the historical gradients are integrated into feature-dependent step size by a different weighting scheme.

In this paper, we propose a generic framework using the mini-batch stochastic normalized gradient as the updating direction (like [Hazan et al., 2015, Levy, 2016]) and the step size is adaptive to each coordinate as in [Duchi et al., 2011, Kingma and Ba, 2014]. In this way, we hope the resulting methods can benefit from both techniques and generate a even better performance than the existing methods. Our framework starts with computing regular mini-batch stochastic gradient, which is immediately normalized. The normalized version is then plugged in the current popular adaptive step size methods, such as Adam [Kingma and Ba, 2014] and AdaGrad [Duchi et al., 2011]. The numerical results shows that normalized gradient always helps to improve the performance of the original methods especially when the network structure is deep, and the best algorithms in our comparison is Adam [Kingma and Ba, 2014] with normalized gradients. It seems to be the first

thorough empirical study on various types of neural networks with this idea. Besides, although we focus our empirical studies on deep learning where the objective is highly non-convex, we also provide a convergence proof under this framework when the problem is convex.

The rest of the paper is organized as follows. In Section 2, we briefly go through the previous work that are related to ours. In Section 3, we formalize the problem to solve and propose the generic algorithm framework. We then provide a concrete example of this type of algorithm and show its convergence property under the convex setting. In Section 4, we conduct comprehensive experimental studies to compare the performance of different algorithms on various neural network structures. Finally, we conclude the paper in Section 5.

## 2   Related Work

A pioneering work on normalized gradient descent (NGD) method was by Nesterov [Nesterov, 1984] where it was shown that NGD can find a $\epsilon$-optimal solution within $O(\frac{1}{\epsilon^2})$ iterations when the objective function is quasi-convex. Kiwiel [Kiwiel, 2001] and Hazan et al [Hazan et al., 2015] extended NGD for upper semi-continuous quasi-convex objective functions and local-quasi-convex objective functions, respectively, and achieved the same iteration complexity. Moreover, Hazan et al [Hazan et al., 2015] showed that NGD's iteration complexity can be reduced to $O(\frac{1}{\epsilon})$ if the objective function is local-quasi-convex and locally-smooth. A stochastic NGD algorithm is also proposed by Hazan et al [Hazan et al., 2015] which, if a mini-batch is used to construct the stochastic normalized gradient in each iteration, finds $\epsilon$-optimal solution with a high probability for locally-quasi-convex functions within $O(\frac{1}{\epsilon^2})$ iterations. Levy [Levy, 2016] proposed a Saddle-Normalized Gradient Descent (Saddle-NGD) method, which adds a zero-mean Gaussian random noise to the stochastic normalized gradient periodically. When applied to strict-saddle functions with some additional assumption, it is shown [Levy, 2016] that Saddle-NGD can evade the saddle points and find a local minimum point approximately with a high probability.

Analogous yet orthogonal to the gradient normalization ideas have been proposed for the deep neural network training. For example, batch normalization [Ioffe and Szegedy, 2015] is used to address the internal covariate shift phenomenon in the during deep learning training. It benefits from making normalization a part of the model architecture and performing the normalization for each training mini-batch. Weight normalization [Salimans and Kingma, 2016], on the other hand, aims at a reparameterization of the weight vectors that decouples the length of those weight vectors from their direction. Recently [Neyshabur et al., 2015] proposes to use path normalization, an approximate path-regularized steepest descent with respect to a path-wise regularizer related to max-norm regularization to achieve better convergence than vanilla SGD and AdaGrad. Perhaps the most related idea to ours is Gradient clipping. It is proposed in [Pascanu et al., 2013] to avoid the gradient explosion, by pulling the magnitude of a large gradient to a certain level. However, this method does not do anything when the magnitude of the gradient is small.

Adaptive step size has been studied for years in the optimization community. The most celebrated method is the line search scheme. However, while the exact line search is usually computational infeasible, the inexact line search also involves a lot of full gradient evaluation. Hence, they are not suitable for the deep learning setting. Recently, provably correct algorithms with adaptive step sizes in the convex scenario start to be applied to the non-convex neural network training, such as AdaGrad [Duchi et al., 2012], Adam [Kingma and Ba, 2014] and RMSProp. However they directly use the unnormalized gradient, which is different from our framework. To the best of our knowledge,

our work seems to be the first study on combining normalized gradient with adaptive stepsize for deep neural network training.

## 3 Algorithm Framework

In this section, we present our algorithm framework to solve the following general problem:

$$\min_{x \in \mathbb{R}^d} f(x) = \mathbb{E}(F(x, \xi)), \tag{1}$$

where $F(\cdot, \xi)$ is a loss function for each $\xi \sim \mathbb{P}$, with $\mathbb{P}$ being a probability distribution. Now our goal is distilled to minimize the objective function $f$ over $x$, where $x$ is usually called the parameter of the model in machine learning literature.

We propose the generic optimization framework in Algorithm 1. In iteration $t$, it firstly computes the (sub)gradient $F'(x_t, \xi_t)$ of $F$ at $x = x_t$ with a mini-batch data $\xi_t$, and then normalizes it to get a direction $g_t = \frac{F'(x_t, \xi_t)}{\|F'(x_t, \xi_t)\|_2}$. The next is to find an adaptive step size $\tau_t$ to move the current parameter $x_t$ towards the negative direction of $g_t$ or its transformation, i.e., $h_t(g_t)$. In fact, our framework can be customized to most of existing first order methods with adaptive step sizes, such as AdaGrad[Duchi et al., 2011], RMSProp and Adam[Kingma and Ba, 2014], by adopting the their step size rules respectively.

---

**Algorithm 1** Normalized Gradient with Adaptive Stepsize (NGAS)

---

1: Choose $x_1$.
2: **for** $t = 1, 2, ...,$ **do**
3:     Sample a mini-batch of data $\xi_t$ and compute the stochastic gradient $g_t = \frac{F'(x_t, \xi_t)}{\|F'(x_t, \xi_t)\|_2}$
4:     $x_{t+1} = x_t - \tau_t h_t(g_t)$, where $\tau_t$ is an adaptive step size.
5: **end for**

---

As a concrete example, we present the normalized gradient customization of AdaGrad in Algorithm 2. As a theoretical contribution of this paper, we further show the $O(\frac{1}{\sqrt{T}})$ convergence of this adaptation, if the objective function $f$ is convex over $x$. In the following, we denote $\|x\|_{H_t} := \sqrt{x^\top H_t x}$ as the the Mahalanobis norm associated to the positive definite diagonal matrix $H_t$. Its dual norm is defined as $\|x\|_{H_t^*} := \sqrt{x^\top H_t^{-1} x}$. The convergence of Normalized AdaGrad is presented in the following theorem.

---

**Algorithm 2** AdaGrad with Normalized Gradient (AdaGradNG)

---

1: Choose $x_1$, set $g_{1:0} = [\ ]$
2: **for** $t = 1, 2, ...,$ **do**
3:     Sample a mini-batch data $\xi_t$ and compute the stochastic gradient $g_t = \frac{F'(x_t, \xi_t)}{\|F'(x_t, \xi_t)\|_2}$
4:     Let $g_{1:t} = [g_{1:t-1}, g_t]$, $s_{t,i} = \|g_{1:t,i}\|_2$, and $H_t = \delta I + \text{diag}(s_t)$
5:     $x_{t+1} = x_t - \eta_t H_t^{-1} g_t$
6: **end for**

---

**Theorem 1** *Suppose $F$ is convex over $x$, $\|F'(x_t, \xi_t)\|_2 \leq M$ and $\|x_t - x^*\|_\infty \leq D_\infty$ for all $t$, with constants $M, D_\infty > 0$. Let $\eta_t = \eta\|F'(x_t, \xi_t)\|$. The iterating points of Normalized AdaGrad are guaranteed to have*

$$f(\bar{x}_T) - f(x^*) \quad \leq \quad \frac{\|x_1 - x^*\|_{H_1}^2}{2\eta T} + \frac{D_\infty^2}{2\eta\sqrt{T}} + \frac{\eta M^2 \sqrt{d}}{\sqrt{T}}.$$

*Proof:* According to the updating scheme of $x_{t+1}$, we have

$$
\begin{aligned}
\|x_{t+1} - x^*\|_{H_t}^2 &= (x_t - \eta_t H_t^{-1} g_t - x^*)^\top H_t (x_t - \eta_t H_t^{-1} g_t - x^*) \\
&\leq \|x_t - x^*\|_{H_t}^2 - 2\eta_t(x_t - x^*)^\top g_t + \eta_t^2 \|g_t\|_{H_t^*}.
\end{aligned}
$$

Because of the definition $\eta_t = \eta\|F'(x_t, \xi_t)\|$ and the definition of $g_t$, the inequality above implies

$$
\begin{aligned}
F(x_t, \xi_t) - F(x^*, \xi_t) &= \frac{\|x_t - x^*\|_{H_t}^2}{2\eta} - \frac{\|x_{t+1} - x^*\|_{H_t}^2}{2\eta} + \frac{\eta_t^2 \|g_t\|_{H_t^*}^2}{2\eta} \\
&\leq \frac{\|x_t - x^*\|_{H_t}^2}{2\eta} - \frac{\|x_{t+1} - x^*\|_{H_t}^2}{2\eta} + \frac{\eta M^2 \|g_t\|_{H_t^*}^2}{2},
\end{aligned}
$$

where the second inequality is because $\eta_t = \eta\|F'(x_t, \xi_t)\| \leq \eta M$. Taking expectation over $\xi_t$ for $t = 1, 2, \ldots$ and averaging the above inequality give

$$\mathbb{E}[f(\bar{x}_T) - f(x^*)] \quad \leq \quad \frac{1}{T}\sum_{t=1}^{T}\left[\frac{\mathbb{E}\|x_t - x^*\|_{H_t}^2}{2\eta} - \frac{\mathbb{E}\|x_{t+1} - x^*\|_{H_t}^2}{2\eta}\right] + \sum_{t=1}^{T}\frac{\eta M^2 \mathbb{E}\|g_t\|_{H_t^*}^2}{2T}. \quad (2)$$

According to the equation (24) in the proof of Lemma 4 in [Duchi et al., 2011], we have

$$\sum_{t=1}^{T}\|g_t\|_{H_t^*}^2 = \sum_{t=1}^{T}\sum_{i=1}^{d}\frac{g_{t,i}^2}{\delta + s_{t,i}} \leq \sum_{i=1}^{d} 2\|g_{1:T,i}\|_2 \leq 2\sqrt{T}d \quad (3)$$

where the second inequality is due to Cauchy-Schwarz inequality and the fact that $\|g_t\|_2 = 1$.

Following the analysis in the proof of Theorem 5 in [Duchi et al., 2011], we show that

$$
\begin{aligned}
\|x_{t+1} - x^*\|_{H_{t+1}}^2 - \|x_{t+1} - x^*\|_{H_t}^2 &= \langle x^* - x_{t+1}, \text{diag}(s_{t+1} - s_t)(x^* - x_{t+1}) \rangle \\
&\leq D_\infty^2 \|s_{t+1} - s_t\|_1 = D_\infty^2 \langle s_{t+1} - s_t, \mathbf{1} \rangle. \quad (4)
\end{aligned}
$$

After applying (3) and (4) to (2) and reorganizing terms, we have

$$
\begin{aligned}
\mathbb{E}[f(\bar{x}_T) - f(x^*)] &\leq \frac{\|x_1 - x^*\|_{H_1}^2}{2\eta T} + \frac{D_\infty^2 \mathbb{E}\langle s_T, \mathbf{1}\rangle}{2\eta T} + \frac{\eta M^2 \sqrt{d}}{\sqrt{T}} \\
&\leq \frac{\|x_1 - x^*\|_{H_1}^2}{2\eta T} + \frac{D_\infty^2 \sqrt{T}}{2\eta T} + \frac{\eta M^2 \sqrt{d}}{\sqrt{T}} \\
&= \frac{\|x_1 - x^*\|_{H_1}^2}{2\eta T} + \frac{D_\infty^2}{2\eta\sqrt{T}} + \frac{\eta M^2 \sqrt{d}}{\sqrt{T}},
\end{aligned}
$$

where the second inequality is also due to the equation (24) in the proof of Lemma 4 in [Duchi et al., 2011], which implies $\langle s_T, \mathbf{1}\rangle = \sum_{i=1}^{d}\|g_{1:T,i}\|_2 \leq \sqrt{T}d$. ∎

# 4 Numerical Experiments

**Basic Experiment Setup** In this section, we conduct comprehensive numerical experiments on different types of neural networks. The algorithms we are testing are SGD with Momentum (SGDM), AdaGrad [Duchi et al., 2013], Adam [Kingma and Ba, 2014] and their normalized gradient counterparts, which are denoted with suffix "NG"[1]. Note that SGDM is the algorithms without adaptive stepsize while Adam and AdaGrad are two popular algorithms used in deep neural network learning with adaptive stepsize. Therefore, the six algorithms under test have covered all the combinations that are with/without normalized gradient and/or adaptive step size.

Our experiments are on four diverse tasks, ranging from image classification to natural language processing. The neural network structures under investigation include multi layer perceptron, long-short term memory and convolution neural networks.

To exclude the potential effect that might be introduced by advanced techniques, in all the experiments, we only adopt the basic version of the neural networks, unless otherwise stated. The loss functions for classifications are cross entropy, while the one for language modeling is log perplexity. Since the computational time is proportional to the epochs, we only show the performance versus epochs. Those with running time are similar so we omit them for brevity. For all the algorithms, we use their default settings. More specifically, for Adam/AdamNG, the initial step size scale $\alpha = 0.001$, first order momentum $\beta_1 = 0.9$, second order momentum $\beta_2 = 0.999$, the parameter to avoid division of zero $\epsilon = 1e^{-8}$; for AdaGrad/AdaGradNG, the initial step size scale is 0.01; for SGDM/SGDMNG. the learning rate is 0.1 and momentum is 0.5. All the codes are written with the deep learning package pytorch[2].

**Result Summary** We want to answer the following questions:

1. Is an algorithm with normalized gradient faster than its unnormalized counterpart.

2. Is an algorithm with adaptive step size faster than the version without?

3. What is the fastest among all the algorithms under test?

In general, the answers to the first two questions above are both affirmative, as long as the neural networks that are either with long dependence (e.g., LSTMs unrolled 40 to 1000 steps) or deep architecture (e.g., MLP or CNN with over 18 layers). The answer to the third, from our empirical study is AdamNG. To give the readers a quick and clear idea of the answers, we extract the key information from figures in the LSTM experiment of Section 4.3 and visualize it in Figure 1. For details of the experiment please see Section 4.3. The left figure shows that the Adam with normalized gradient is faster than its original version. The middle figure shows that Adam and Adagrad (methods with adaptive step size) are both faster than SGDM. The right figure shows that the AdamNG has the fastest convergence compared to the normalized version of Adagrad and SGDM. In the following, we elaborate on the details of each individual experiments.

---

[1]In the experiment, the normalization is on each individual variable rather than on the concatenation of all of them.
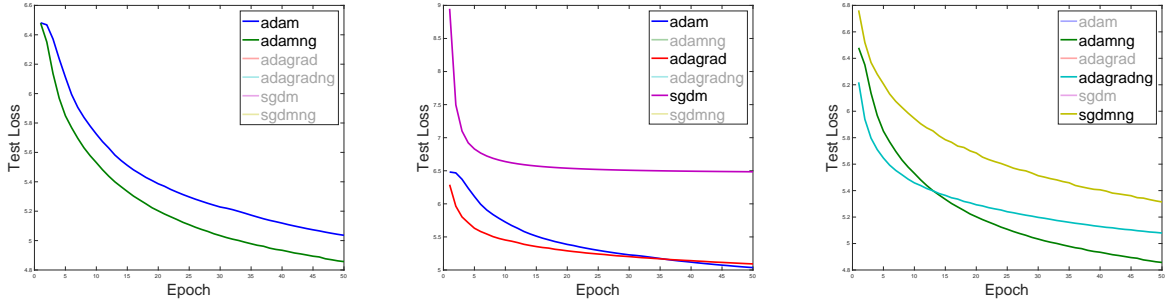
[2]http://pytorch.org/

Figure 1: The visualization of answers to three questions by the LSTM experiment. The BPTT length is 1000. Only testing objectives are shown as the phenomenon of training is similar.

## 4.1 MNIST Image Classification with Feedforward Neural Network

The first network structure we are going to test upon is the Multi Layer Perceptron (MLP). We will adopt the handwritten digit recognition data set MNIST[3]Lecun et al. [1998], in which, each data is an image of hand written digits from $\{1, 2, 3, 4, 5, 6, 7, 8, 9, 0\}$. There are 60k training and 10k testing examples and the task is to tell the right number contained in the test image. Our approach is applying MLP to learn an end-to-end classifier, where the input is the raw $28 \times 28$ images and the output is the label probability. The predicted label is the one with the largest probability. In each middle layer of the MLP, the hidden unit number are 100, and the first and last layer respectively contain 784 and 10 units. The activation functions between layers are all sigmoid and the batch size is 100 for all the algorithms.

We choose different numbers of layer from $\{6, 12, 18\}$. The results are shown in Figure 2. Each column of the figures corresponds to the training and testing objective curves of the MLP with a given layer number. From left to right, the layer numbers are respectively 6, 12 and 18. We can see that, when the network is as shallow as containing 6 layers, the normalized stochastic gradient descent can outperform its unnormalized counterpart, while the Adam and AdaGrad are on par with or even slightly better than their unnormalized versions. As the networks become deeper, the acceleration brought by the gradient normalization turns more significant. For example, starting from the second column, AdamNG outperforms Adam in terms of both training and testing convergence. In fact, when the network depth is 18, the AdamNG can still converge to a small objective value while Adam gets stuck from the very beginning. We can observe the similar trend in the comparison between AdaGrad (resp. SGDM) and AdaGradNG (resp. SGDMNG). On the other hand, the algorithms with adaptive step sizes can usually generate a stable learning curve. For example, we can see from the last two column that SGDNG causes significant fluctuation in both training and testing curves. Finally, under any setting, AdamNG is always the best algorithm in terms of convergence performance.

## 4.2 CIFAR-10 Image Classification with Deep Convolution Neural Network

Given the success of the proposed framework on training deep feedforward neural networks, we anticipate to see the similar phenomenon in other types of networks. In this section, we will test
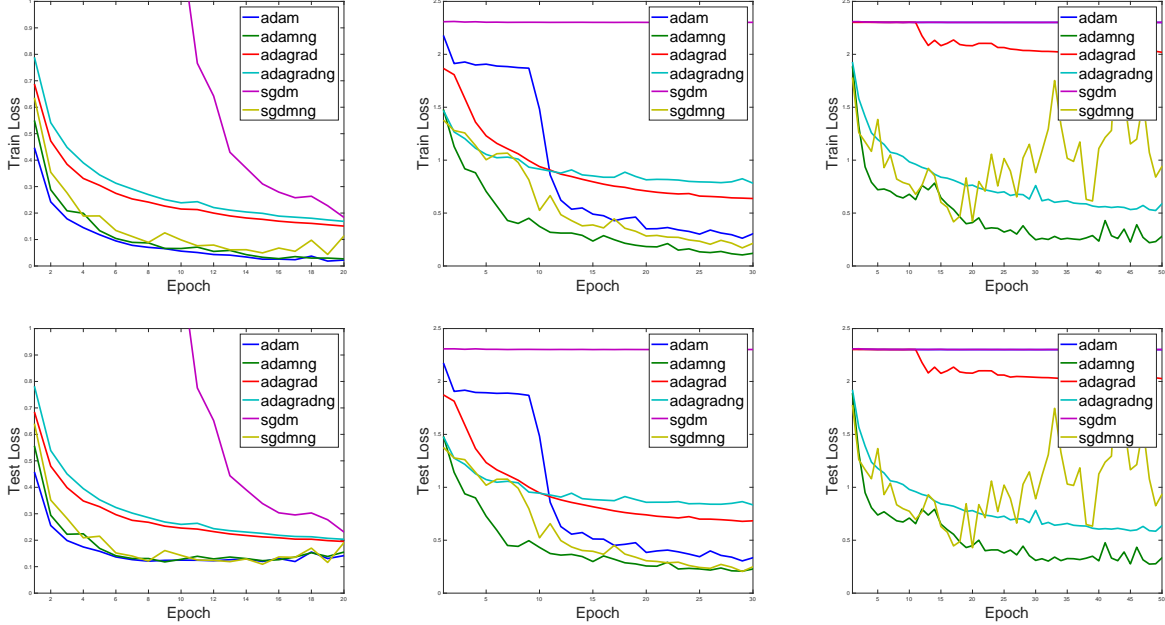
---

[3]http://yann.lecun.com/exdb/mnist/

7

Figure 2: The training and testing objective curves on MNIST dataset with multi layer perceptron. From left to right, the layer numbers are 6, 12 and 18 respectively. The first row is the training curve and the second is testing.

the algorithms on image classification task with CIFAR-10[4][Krizhevsky, 2009] data set, which will be approached by a deep Convolution Neural Network (CNN). The CIFAR-10 is a collection of 10 class of natural images consisting of 60k 32x32 color images, with 6k images per class. In this experiment, we adopt the network VGG, short for Visual Geometry Group, a deep convolution neural network to finish our task. It was proposed by [Simonyan and Zisserman, 2014] to attack the ImageNet Challenge 2014 and won the localization and classification tracks. We choose the very original version of the networks containing 11 (VGG11) or 19 (VGG19) layers. Interested readers might take to look at [Simonyan and Zisserman, 2014] to know more details of the network.

The results are summarized in Figure 3. The take-home messages are similar to those in the last section. In particular, when the depth of the network is relatively shallow (VGG11), the gradient normalization with adaptive stepsize strategy can hardly bring any benefit to the convergence. In fact, the vanilla SGD with momentum (without either normalized gradient or adaptive stepsize) achieves the fastest convergence in both training and testing. Nevertheless, when the depth increases to 19, the power of the proposed strategy becomes clear. More specifically, while both SGDM and Adam get stuck at the beginning of training in VGG19, all the gradient normalization versions can push the objective to go down. Among those converging methods, AdamNG achieves the best testing objective with the fastest speed. Therefore, we can safely draw the conclusion that the NGAS framework is also suitable to train deep convolution neural networks.

---

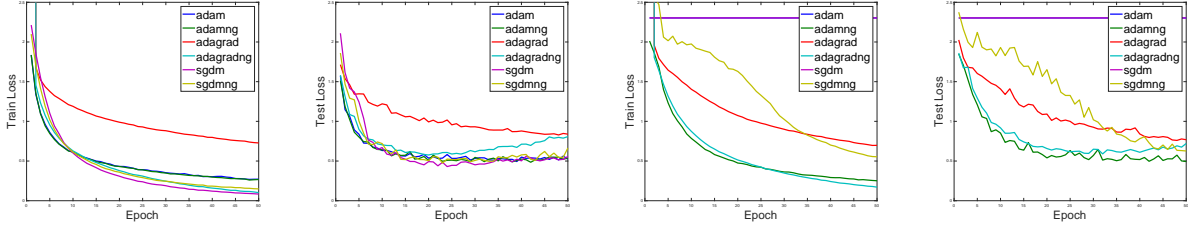[4]https://www.cs.toronto.edu/~kriz/cifar.html

Figure 3: The training and testing objective curves on CIFAR-10 dataset with VGG convolution neural networks. From left to right, the figures are respectively VGG11-Training, VGG11-Testing, VGG19-Training and VGG19-Testing.

## 4.3 Language Modeling with Recurrent Neural Network

Now that the NGAS framework is powerful in training MLP and CNN for the image classification tasks, we conjecture that this idea can also be applied to learning the Recurrent Neural Networks (RNN). Our confidence is based on the fact that an unrolled RNN can be considered as another type of deep network where the stretching direction is horizontal while the weights (parameters) are shared. In this section, we test the performance of the proposed algorithm on the word-level language modeling task with a popular type of RNN, i.e. single directional Long-Short Term Memory (LSTM) networks [Hochreiter and Schmidhuber, 1997]. The data set under use is Penn Tree Bank (PTB) [Marcus et al., 1993] data, which, after preprocessed, contains 929k training words, 73k validation and 82k test words. The vocabulary size is about 10k. The LSTM has 2 layers, each containing 200 hidden units. The word embedding has 200 dimensions which is trained from scratch. The batch size is 100. We vary the length of the backprop through time (BPTT) within the range $\{40, 400, 1000\}$. To prevent overfitting, we add a dropout regularization with rate 0.5 under all the settings.

The results are shown in Figure 4. The conclusions drawn from those figures are again similar to those in the last two experiments. However, the slightly different yet cheering observations is that the AdamNG is uniformly better than all the other competitors with any training sequence length. The superiority in terms of convergence speedup exists in both training and testing.

## 4.4 Sentiment Analysis with Convolution Neural Network

In the experiments of previous sections, we mainly focus on the comparison of convergence speed among different methods. In this section, we are going to quantitatively show that the proposed NGAS framework also enables the neural network to have better generalization ability. To achieve this, the network is not necessarily as deep as previous experiments. The task here is the sentiment analysis with convolution neural network. The dataset under use is Rotten Tomatoes[5] Pang and Lee [2005], a movie review dataset containing 10,662 documents, with half positive and half negative. We randomly select around 90% for training and 10% for validation. The model is a single layer convolution neural network that follows the setup of [Kim, 2014]. The word embedding under use is randomly initialized and of 128-dimension.

---

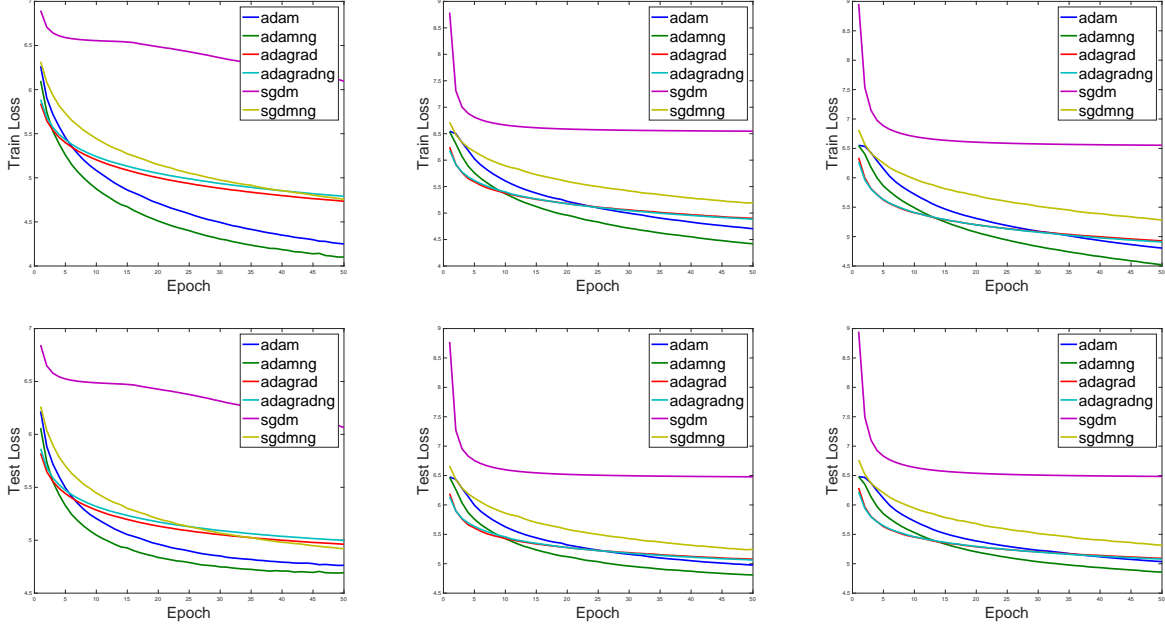[5]http://www.cs.cornell.edu/people/pabo/movie-review-data/

9

Figure 4: The training and testing objective curves on Penn Tree Bank dataset with LSTM recurrent neural networks. The first row is the training objective while the second is the testing. From left to right, the training sequence (BPTT) length are respectively 40, 400 and 1000. Dropout with 0.5 is imposed.

For each algorithm, we run 150 epochs on the training data, and report the best validation accuracy in Table 1. The messages conveyed by the table is three-fold. Firstly, the algorithms using normalized gradient achieve much better validation accuracy than their unnormalized versions. Secondly, those with adaptive stepsize always obtain better accuracy than those without. This is easily seen by the comparison between Adam and SGDM. The last point is the direct conclusion from the previous two that the algorithm using normalized gradient with adaptive step sizes, namely AdamNG, outperforms all the remaining competitors.

| Algorithm | AdamNG | Adam | AdaGradNG | AdaGrad | SGDMNG | SGDM |
|---|---|---|---|---|---|---|
| Validation Accuracy | **77.11%** | 74.02% | 71.95% | 69.89% | 71.95% | 64.35% |

Table 1: The Best validation accuracy achieved by the different algorithms.

# 5    Conclusion

In this paper, we propose a generic algorithm framework for first order optimization. It is particularly effective for addressing the vanishing and exploding gradient challenge in training with non-convex loss functions, such as in the context of convolutional and recurrent neural networks. Our method is based on normalizing the gradient to establish the descending direction regardless of its magnitude,

and then separately estimating the ideal step size adaptively. This method is quite general and may be applied to different types of networks and various architectures. Although the primary application of the algorithm is deep neural network training, we provide a convergence for the new method under the convex setting.

Empirically, the proposed method exhibits very promising performance in training different types of networks (convolutional, recurrent) across multiple well-known data sets (image classification, natural language processing, sentiment analysis, etc.). In general, the positive performance differential compared to the state of the art is most striking for very deep networks, as shown in our comprehensive experimental study.

# References

Yoshua Bengio. Learning deep architectures for ai. *Foundations and trends® in Machine Learning*, 2(1):1–127, 2009.

John Duchi, Michael I Jordan, and Brendan McMahan. Estimation, optimization, and parallelism when data is sparse. In *NIPS*, pages 2832–2840, 2013.

John C. Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12:2121–2159, 2011.

John C Duchi, Alekh Agarwal, and Martin J Wainwright. Dual averaging for distributed optimization: convergence analysis and network scaling. *Automatic Control, IEEE Transactions on*, 57(3):592–606, 2012.

Elad Hazan, Kfir Y. Levy, and Shai Shalev-Shwartz. Beyond convexity: Stochastic quasi-convex optimization. In *NIPS*, pages 1594–1602, 2015.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8): 1735–1780, 1997.

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

Yoon Kim. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*, 2014.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014. URL http://arxiv.org/abs/1412.6980.

Krzysztof C Kiwiel. Convergence and efficiency of subgradient methods for quasiconvex minimization. *Mathematical programming*, 90(1):1–25, 2001.

Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, 2009.

Yann Lecun, Leon Bottou, Yoshua Bengio, and Patrick Haffner? Gradient-based learning applied to document recognition. In *Proceedings of the IEEE*, pages 2278–2324, 1998.

Kfir Y. Levy. The power of normalization: Faster evasion of saddle points. *CoRR*, abs/1611.04831, 2016. URL http://arxiv.org/abs/1611.04831.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. Building a large annotated corpus of english: The penn treebank. *Computational Linguistics*, 19(2):313–330, 1993.

Nesterov. Minimization methods for nonsmooth convex and quasiconvex functions. *Matekon*, 29: 519–531, 1984.

Behnam Neyshabur, Ruslan Salakhutdinov, and Nathan Srebro. Path-sgd: Path-normalized optimization in deep neural networks. In *NIPS*, pages 2422–2430, 2015.

Bo Pang and Lillian Lee. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of the 43rd annual meeting on association for computational linguistics*, pages 115–124. Association for Computational Linguistics, 2005.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. In *ICML*, pages 1310–1318, 2013.

Tim Salimans and Diederik P. Kingma. Weight normalization: A simple reparameterization to accelerate training of deep neural networks. In *NIPS*, 2016.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014. URL `http://arxiv.org/abs/1409.1556`.