pbSGD: Powered Stochastic Gradient Descent Methods for Accelerated Nonconvex Optimization

Beitong Zhou[1] , **Jun Liu**[2] , **Weigao Sun**[1] , **Ruijuan Chen**[1] , **Claire Tomlin**[3] and **Ye Yuan**[1]*
[1]School of Artificial Intelligence and Automation, Huazhong University of Science and Technology
[2]Department of Applied Mathematics, University of Waterloo
[3]Department of Electrical Engineering and Computer Sciences, UC Berkeley
zhoubt@hust.edu.cn, j.liu@uwaterloo.ca, sunweigao@hust.edu.cn, ruijuanchen@hust.edu.cn,
tomlin@berkeley.edu, yye@hust.edu.cn

Abstract

We propose a novel technique for improving the stochastic gradient descent (SGD) method to train deep networks, which we term pbSGD. The proposed pbSGD method simply raises the stochastic gradient to a certain power elementwise during iterations and introduces only one additional parameter, namely, the power exponent (when it equals to 1, pbSGD reduces to SGD). We further propose pbSGD with momentum, which we term pbSGDM. The main results of this paper present comprehensive experiments on popular deep learning models and benchmark datasets. Empirical results show that the proposed pbSGD and pbSGDM obtain faster initial training speed than adaptive gradient methods, comparable generalization ability with SGD, and improved robustness to hyper-parameter selection and vanishing gradients. pbSGD is essentially a gradient modifier via a nonlinear transformation. As such, it is orthogonal and complementary to other techniques for accelerating gradient-based optimization such as learning rate schedules. Finally, we show convergence rate analysis for both pbSGD and pbSGDM methods. The theoretical rates of convergence match the best known theoretical rates of convergence for SGD and SGDM methods on nonconvex functions.

1 Introduction

Stochastic optimization as an essential part of deep learning has received much attention from both the research and industry communities. High-dimensional parameter spaces and stochastic objective functions make the training of deep neural network (DNN) extremely challenging. Stochastic gradient descent (SGD) [Robbins and Monro, 1951] is the first widely used method in this field. It iteratively updates the parameters of a model by moving them in the direction of the negative gradient of the objective evaluated on a mini-batch. Based on SGD, other stochastic optimization algorithms, e.g., SGD with Momentum (SGDM) [Qian, 1999], AdaGrad [Duchi *et al.*, 2011], RMSProp [Tieleman and Hinton, 2012], Adam [Kingma and Ba, 2015] are proposed to train DNN more efficiently.

Despite the popularity of Adam, its generalization performance as an adaptive method has been demonstrated to be worse than the non-adaptive ones. Adaptive methods (like AdaGrad, RMSProp and Adam) often obtain faster convergence rates in the initial iterations of training process. Their performance, however, quickly plateaus on the test data [Wilson *et al.*, 2017]. In [Reddi *et al.*, 2018], the authors provided a convex optimization example to demonstrate that the exponential moving average technique can cause non-convergence in RMSProp and Adam, and they proposed a variant of Adam called AMSGrad, hoping to solve this problem. The authors provide a theoretical guarantee of convergence but only illustrate its better performance on train data. However, the generalization ability of AMSGrad on test data is found to be similar to that of Adam, and a considerable performance gap still exists between AMSGrad and SGD [Chen *et al.*, 2018]. Indeed, the optimizer is chosen as SGD (or with Momentum) in several recent state-of-the-art works in natural language processing and computer vision [Luo *et al.*, 2018; Wu and He, 2018], where in these instances SGD does perform better than adaptive methods. Despite the practical success of SGD, obtaining sharp convergence results in the nonconvex setting for SGD to efficiently escape saddle points (i.e., convergence to second-order stationary points) remains a topic of active research [Jin *et al.*, 2019; Fang *et al.*, 2019].

Related Works. SGD, as the first efficient stochastic optimizer for training deep networks, iteratively updates the parameters of a model by moving them in the direction of the negative gradient of the objective function evaluated on a mini-batch. SGDM brings a Momentum term from the physical perspective, which obtains faster convergence speed than SGD. The Momentum idea can be seen as a particular case of exponential moving average (EMA). Then the adaptive learning rate (ALR) technique is widely adopted but also disputed in deep learning, which is first introduced by AdaGrad. Contrast to the SGD, AdaGrad updates the parameters according to the square roots of the sum of squared coordinates in all the past gradients. AdaGrad can potentially lead to huge gains in terms of convergence [Duchi *et al.*, 2011] when the gradients are sparse. However, it will also lead to rapid learning rate decay when the gradients are dense. RMSProp, which first appeared

*Contact Author

in an unpublished work [Tieleman and Hinton, 2012], was proposed to handle the aggressive, rapidly decreasing learning rate in AdaGrad. It computes the exponential moving average of the past squared gradients, instead of computing the sum of the squares of all the past gradients in AdaGrad. The idea of AdaGrad and RMSProp propelled another representative algorithm: Adam, which updates the weights according to the mean divided by the root mean square of recent gradients, and has achieved enormous success. Recently, research to link discrete gradient-based optimization to continuous dynamic system theory has received much attention [Yuan *et al.*, 2019; Mazumdar and Ratliff, 2018], more of which can be found in the survey [Yang *et al.*, 2019]. While the proposed optimizer excels at improving initial training, it is completely complementary to the use of learning rate schedules [Smith and Topin, 2017; Loshchilov and Hutter, 2016]. We will explore how to combine learning rate schedules with the pbSGD optimizer in future work.

While other popular techniques focus on modifying the learning rates and/or adopting momentum terms in the iterations, we propose to modify the gradient terms via a nonlinear function called the Powerball function by the authors of [Yuan *et al.*, 2019]. In [Yuan *et al.*, 2019], the authors presented the basic idea of applying the Powerball function in gradient descent methods. In this paper, we 1) systematically present the methods for stochastic optimization with and without momentum; 2) conduct comprehensive experiments using popular deep learning models and benchmark datasets; 3) provide convergence analysis results. Another related work was presented in [Bernstein *et al.*, 2018], where the authors presented a version of stochastic gradient descent which uses only the signs of gradients. This essentially corresponds to the special case of pbSGD (or pbSGDM) when the power exponential γ is set to 0. We also point out that despite the name resemblance, the power PowerSign optimizer proposed in [Bello *et al.*, 2017] is a conditional scaling of the gradient, whereas the proposed pbSGD optimizer applies a component-wise transformation to the gradient.

Contributions. This paper proposes a new class of stochastic optimizers for training deep networks. We highlight the main contributions of the paper as follows:

1. We propose the pbSGD, a systematic technique for stochastic optimization. pbSGD simply applies the Powerball function (with only one additional parameter γ) on the stochastic gradient term in SGD. Hence, it is easy to implement and requires no extra memory. We also propose the pbSGDM as a variant of pbSGD with momentum.

2. We conduct extensive experimental studies on multiple popular deep learning tasks and benchmark datasets. The results empirically demonstrate that our methods gain faster convergence rate especially in the early train process compared with the adaptive gradient methods. Meanwhile, the proposed methods show comparable generalization ability compared with SGD and SGDM.

3. We show convergence rates of the proposed pbSGD and pbSGDM in the nonconvex setting. The rates of convergence match the best known rates of convergence for SGD and SGDM methods on nonconvex functions. In fact, the special cases of pbSGD and pbSGDM (when $\gamma = 1$) provide the currently best convergence bounds for SGD and SGDM in the nonconvex setting in terms of both the constants and rates of convergence (see, e.g. [Yan *et al.*, 2018]).

2 Algorithms

In this section, we present the main algorithms proposed in this paper: pbSGD and pbSGDM. pbSGD combines the Powerball function technique with stochastic gradient descent, and pbSGDM is an extension of pbSGD to include a momentum term. The main results of this paper include extensive experiments that demonstrate the promising performance of the proposed algorithms, compared to other popular stochastic optimizers for train deep networks. We shall also state in Section 4 that theoretically both methods converge and attain at least the best known rates of convergence for SGD and SGDM on nonconvex functions.

2.1 pbSGD

Train a DNN with n free parameters can be formulated as an unconstrained optimization problem

$$\min_{x\in\mathbb{R}^n} f(x), \tag{1}$$

where $f(\cdot):\mathbb{R}^n\to\mathbb{R}$ is a function bounded from below. SGD proved itself an efficient and effective solution for high-dimensional optimization problems. It optimizes f by iteratively updating the parameter vector $x_t\in\mathbb{R}^n$ at step t, in the opposite direction of a stochastic gradient $g(x_t,\xi_t)$ (where ξ_t denotes a random variable), which is calculated on t-th mini-batch of train dataset. The update rule of SGD for solving problem (1) is

$$x_{t+1} = x_t - \alpha_t g(x_t,\xi_t), \tag{2}$$

starting from an arbitrary initial point x_1, where α_t is known as the learning rate at step t. In the rest of the article, let $g_t = g(x_t,\xi_t)$ for the sake of notation. We then introduce a nonlinear transformation $\sigma_\gamma(z) = \mathrm{sign}(z)|z|^\gamma$ named as the Powerball function where $\mathrm{sign}(z)$ returns the sign of z, or 0 if $z=0$. For any vector $z=(z_1,\dots,z_n)^\top$, the Powerball function $\sigma_\gamma(z)$ is applied to all elements of z. A parameter $\gamma\in\mathbb{R}$ is introduced to adjust the mechanism and intensity of the Powerball function.

Applying the Powerball function to the stochastic gradient term in the update rule (2) gives the proposed pbSGD algorithm:

$$x_{t+1} = x_t - \alpha_t \sigma_\gamma(g_t), \tag{3}$$

where $\gamma\in[0,1]$ is an additional parameter. Clearly, when $\gamma=1$, we obtain the vanilla SGD (2).

2.2 pbSGDM

By introducing a momentum term [Polyak, 1964; Nesterov, 1983] to (3), we obtain pbSGD with Momentum (pbSGDM), whose update rule is

$$\begin{cases} v_{t+1} = \beta v_t - \alpha_t \sigma(g_t), \\ x_{t+1} = x_t + v_{t+1}, \end{cases} \tag{4}$$

where $v_1 = 0$. Clearly, when $\beta = 0$, pbSDGM (4) reduces to pbSGD (3). Pseudo-code of the proposed pbSGDM is detailed in Algorithm 2.

3 Experiments

The purpose of this section is to demonstrate the efficiency and effectiveness of the proposed pbSGD and pbSGDM algorithms. We conduct experiments of different model architectures on datasets in comparison with widely used optimization methods including the non-adaptive method SGDM and three popular adaptive methods: AdaGrad, RMSprop and Adam. This section is mainly composed of two parts: **(1)** the convergence and generalization experiments and **(2)** the Powerball feature experiments. The setup for each experiment is detailed in Table 1[1]. In the first part, we present empirical study of different deep neural network architectures to see how the proposed methods behave in terms of convergence speed and generalization. In the second part, the experiments are conducted to explore the potential features of pbSGD and pbSGDM, including the improved robustness of hyper-parameters and the further combination with learning rate policies.

To ensure stability and reproducibility, we conduct each experiment at least 5 times from randomly initialization and the averaged results are shown. The settings of hyper-parameters for a specific optimization method that can achieve the best performance on the test set are chosen. When two settings achieve similar test performance, the setting which converges faster is adopted.

We can have the following findings from our experiments in the first part: **(1)** The proposed pbSGD and pbSGDM methods exhibit better convergence rate than other adaptive methods such as Adam and RMSprop. **(2)** Our proposed methods achieve better generalization performance than adaptive methods although are slightly worse than SGDM. **(3)** The Powerball function can help relieve the gradient vanishing phenomenon.

Experiments	Datasets	Architectures
Convergence and Generalization	CIFAR-10	ResNet-50 DenseNet-121
	CIFAR-100	ResNext-29 (16x64d) WideResNet (depth=26, k=10)
	ImageNet	ResNet-50
Powerball Feature	MNIST	13-Layer MLP

Table 1. Models and datasets in our experiments.

3.1 Hyper-parameter Tuning

Since the initial learning rate has a large impact on the performance of optimizers, we implement a logarithmically-spaced grid search strategy around the recommended learning rate for each optimization method, and leave other hyper-parameters to their default settings.

SGDM. The default learning rate for SGDM is 0.01. We tune the learning rate on a logarithmic scale from $\{1, 0.1, 0.01, 0.001, 0.0001\}$. The momentum value in all experiments is set to default value 0.9.

pbSGD, pbSGDM. The learning rates for pbSGD and pbSGDM are chosen from the same range $\{1, 0.1, 0.01, 0.001, 0.0001\}$ as SGDM. The momentum value for pbSGDM is also 0.9. Note that $\gamma = 1$ in Powerball function corresponds to the SGD or SGDM. Based on extensive experiments, we empirically tune γ from $\{0.5, 0.6, 0.7, 0.8, 0.9\}$.

AdaGrad. The learning rates for AdaGrad are $\{0.1, 0.05, 0.01, 0.005, 0.001\}$ and we choose 0 for the initial accumulator value.

RMSprop, Adam. Both have the default learning rate 0.001 and their learning rates are searched from $\{0.01, 0.005, 0.001, 0.0005, 0.0001\}$. The parameters β_1, β_2 and the perturbation value ε are set to default.

As previous findings [Wilson *et al.*, 2017] show, adaptive methods generalize worse than non-adaptive methods and carefully tuning the initial learning rate yields significant improvements for them. To better compare with adaptive methods, once we have found the value that was best performing in adaptive methods, we would try the learning rate between the best learning rate and its closest neighbor. For example, if we tried learning rates $\{0.01, 0.005, 0.001, 0.0005, 0.0001\}$ and 0.0001 was best performing learning rate, we would try the learning rate 0.0002 to see if performance was improved. We iteratively update the learning rate until performance could not improve any more. For all experiments, we used a mini-batch size of 128 (except 256 in the ImageNet experiment).

3.2 Convergence and Generalization

Fig. 1 shows the learning curves of five experiments we have conducted to observe the performance of pbSGD and pbSGDM in comparison with other widely-used optimization methods.

ResNet-50 on CIFAR-10. We trained a ResNet-50 [He *et al.*, 2016] model on CIFAR-10 [Krizhevsky and Hinton, 2009] and our results are shown in Fig. 1(a) and Fig. 1(b). We ran each experiment for a fixed budget of 160 epochs and reduced the learning rate by a factor of 10 after every 60 epochs [Wilson *et al.*, 2017].

As the figure shows, the adaptive methods converged fast and appeared to be performing better than the non-adaptive method SGDM as expected. For pbSGD and pbSGDM, we observed the same tendency in convergence rate as AdaGrad, which outperformed Adam and RMSprop. For the test performance, adaptive methods and our proposed methods still outperform SGDM in the early stage. SGDM achieved a final best overall test accuracy of 94.75%. The pbSGD and pbSGDM achieved test accuracies of 94.17% and 94.13% respectively, which are slightly worse than SGDM. The best adaptive method, Adam, achieved a test accuracy of 93.38%.

WideResNet on CIFAR-100: Next, we conducted experiments on the CIFAR-100 [Krizhevsky and Hinton, 2009] dataset using WideResNet [Zagoruyko and Komodakis, 2016] model. The fixed budget here is 120 epochs and the learning rate reduces by a factor of 10 after every 60 epochs. The results are shown in Fig. 1(e) and Fig. 1(f).

[1]Architectures in generalization and convergence experiments can be found at the following links: **(1)** ResNet-50 and DenseNet-121 on CIFAR-10: https://github.com/kuangliu/pytorch-cifar; **(2)** ResNext-29 and WideResNet on CIFAR-100: https://github.com/junyuseu/pytorch-cifar-models; **(3)** ResNet50 on ImageNet: https://github.com/pytorch/examples/tree/master/imagenet

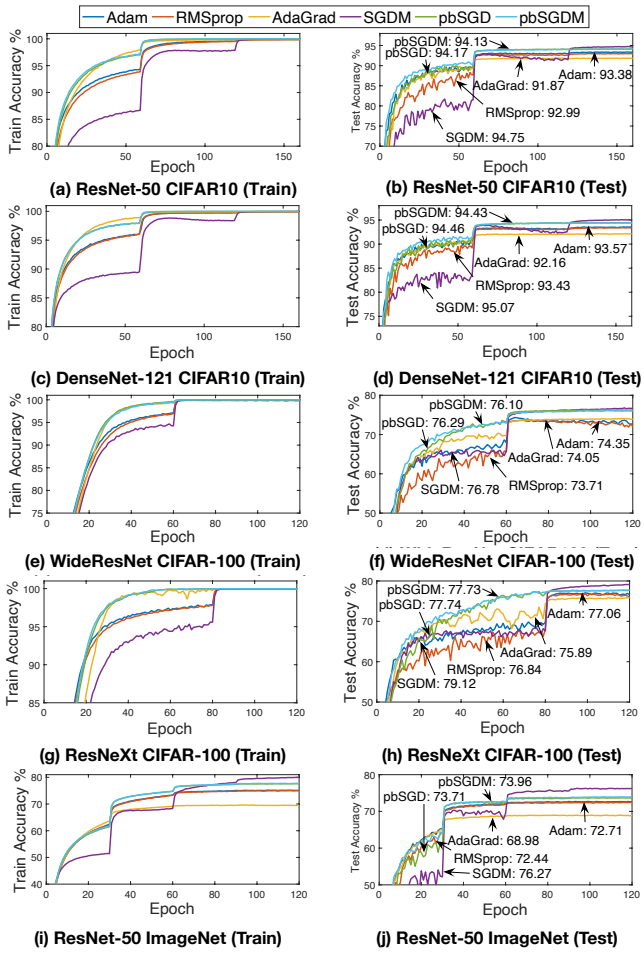

Figure 1. Train and test accuracy for different models and datasets. The annotations indicate the best overall test accuracy for each optimization method. The γ values in five experiments are $0.8, 0.8, 0.8, 0.7, 0.8$ for pbSGD and $0.8, 0.8, 0.8, 0.8, 0.7$ for pbSGDM, respectively.

The performance of the pbSGD and pbSGDM are still promising in both the train set and test set. pbSGD, pbSGDM and AdaGrad had the fastest initial progress. In the test set, pbSGD and pbSGDM had much better test accuracy than all other adaptive methods. SGDM surpassed pbSGD and pbSGDM by epoch 60 when the learning rate decayed. SGDM had the best test accuracy of 76.78%, while pbSGD and pbSGDM achieved accuracies of 76.29% and 76.10%, respectively, with approximately 0.5% gap in test performance compared with SGDM. The best adaptive methods, still Adam, achieved test accuracy of 74.35% and had much larger gap of 2.5% in performance compared to SGDM.

ResNet-50 on ImageNet. Finally, we conducted experiments on the ImageNet dataset [Russakovsky *et al.*, 2015] using ResNet-50 model. The fixed budget here is 120 epochs and the learning rate reduces by a factor of 10 after every 30 epochs. The results are shown in Fig. 1(i) and Fig. 1(j). We observed that pbSGD and pbSGDM gave better convergence rates than adaptive methods while AdaGrad quickly plateaus due to too many parameter updates. For test set, we can notice that although SGDM achieved the best test accuracy of 76.27%, pbSGD and pbSGDM gave the results of 73.71% and 73.96%, which were better than those of adaptive methods.

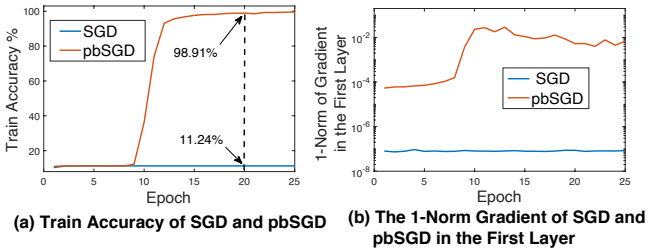

Figure 2. (a) Train accuracy comparison between SGD (learning rate $= 0.1$) and pbSGD (learning rate $= 0.1$, $\gamma = 0.4$) on the 13-layer fully-connected neural network. The arrows annotate the accuracy values of both methods at epoch 20. (b) The 1-norm of stochastic gradients of SGD and pbSGD in the first layer of the fully-connected neural network at every epoch.

Additional experiments. (DenseNet-121 [Huang *et al.*, 2017] on CIFAR-10 and ResNeXt [Xie *et al.*, 2017] on CIFAR100) are shown in Fig. 1(c)(d)(g)(h). We observed similar results as in the other experiments.

3.3 Mitigation of of Gradient Vanishing

In deep learning, the phenomenon of gradient vanishing poses difficulties in training very deep neural networks by SGD. During the training process, the stochastic gradients in early layers can be extremely small due to the chain rule, and this can even completely stop the networks from being trained. Our proposed pbSGD method can relieve the phenomenon of gradient vanishing by effectively rescaling the stochastic gradient vectors.

To validate this, we conduct experiments on the MNIST [LeCun *et al.*, 1998] dataset by using a 13-layer fully-connected neural network with ReLU activation functions. The SGD and proposed pbSGD are compared in terms of train accuracy and 1-norm of gradient vector. As can be observed in Fig. 2, SGD cannot train such a deep network and its train accuracy remained below 12%. By contrast, the train accuracy of pbSGD grows quickly to 98.91% after few epochs of exploration. We further compare the 1-norm of stochastic gradient vector in the first layer for both SGD and pbSGD. It is clear that the Powerball function amplifies the stochastic gradients and helps relieve the gradient vanishing phenomenon.

3.4 Improved Robustness to Hyper-parameter

Through experiments, we have also found that the hyper-parameter γ cannot only accelerate the training process but also improve the optimization robustness, including the robustness of test accuracy and convergence. The dataset and the model in this section are CIFAR-10 and ResNet-50.

Improved robustness of test accuracy w.r.t. learning rates. to verify the robustness of test accuracy for pbSGD, we conduct experiments with γ from 0 to 1.0 and learning rates chosen from $1.0, 0.1, 0.01, 0.001$. SGDM is selected for comparison and the results are summarized in Table. 2. It is observed

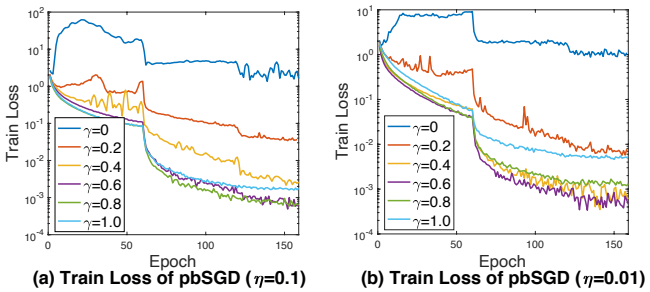

Figure 3. Effects of different γ on convergence. We show the best train loss on CIFAR-10 dataset of ResNet-50 trained with pbSGD. While the γ which achieves the best convergence performance is closely related to the choice of learning rates, a γ chosen in the range of 0.6–1.0 provides better robustness to the change of learning rates.

	$\eta = 1.0$	$\eta = 0.1$	$\eta = 0.01$	$\eta = 0.001$
$\gamma = 0$	10.00	83.93	89.35	91.95
$\gamma = 0.1$	10.00	88.41	91.85	93.38
$\gamma = 0.2$	83.86	91.54	92.52	93.28
$\gamma = 0.3$	89.72	92.53	92.94	93.46
$\gamma = 0.4$	92.46	93.29	93.86	92.59
$\gamma = 0.5$	93.58	93.73	93.76	91.40
$\gamma = 0.6$	94.12	94.07	93.51	90.06
$\gamma = 0.7$	94.40	94.18	93.49	87.56
$\gamma = 0.8$	94.74	94.12	92.69	85.63
$\gamma = 0.9$	94.58	94.11	91.97	83.46
$\gamma = 1.0$	94.89	94.16	90.67	79.87

Table 2. Effects of different γ on test accuracy. We show the best Top-1 accuracy on CIFAR-10 dataset of ResNet-50 trained with pbSGD and SGDM. Although the best choice of γ depends on learning rates, the selections can be quite robust considering the test accuracy.

that the hyper-parameter γ could help regularize the test performance while the learning rate decreases. For example, when $\eta = 0.001$ and $\gamma = 0.6$, pbSGD get the best test accuracy of 90.06% compared with 79.87% accuracy of SGD.

Improved robustness of convergence w.r.t. learning rates. combinations of γ from $0, 0.2, 0.4, 0.6, 0.8, 1.0$ and two learning rate values $0.1, 0.01$ are chosen. The train loss is recorded in Fig. 3 to compare the convergence performance. It is again observed that the hyper-parameter γ (in the range of 0.6–1.0) can improve robustness of convergence with the change of learning rate.

3.5 Combining with Learning Rate Schedules

Motivated by recent advancement in designing learning rate schedules such as CLR [Smith and Topin, 2017] and SGDR [Loshchilov and Hutter, 2016] policies, we conducted some preliminary experiments on combining learning rate schedules with pbSGD to improve its performance. The results are shown in Fig. 4. The selected learning rate schedule is warm restarts introduced in [Loshchilov and Hutter, 2016], which reset the learning rate to the initial value after a cycle

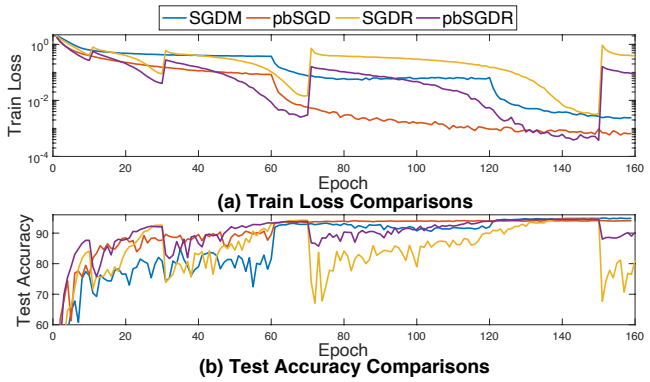

Figure 4. Train loss and test accuracy for SGDR and pbSGDR. Learning rate schedules also help accelerate training of pbSGD and improve the test performance.

of decaying the learning rate with a cosine annealing for each batch. In Fig. 4, SGDM combined with the warm restart policy is named as SGDR, while for pbSGDR we combine pbSGD with the warm restart policy. The hyper-parameter setting is $T_0 = 10$ and $T_{mult} = 2$ for warm restarts [Loshchilov and Hutter, 2016]. We test their performance on CIFAR-10 dataset with ResNet-50.

The results showed that the learning rate policy can improve both the convergence and test performance of pbSGD. Indeed, pbSGDR achieved the lowest train error compared with SGDM, SGDR and pbSGD. The test accuracy for pbSGDR was also improved from the 94.12% of pbSGD to 94.64%. The results demonstrate that the nonlinear transformation of gradients given by the Powerball function is orthogonal and complementary to existing optimization methods. As such, its combination with other techniques could potentially further improve the overall performance.

3.6 Comparison with SGD

We further compare pbSGD with vanilla SGD to build an intuition for how our proposed algorithm boosts the performance. The comparisons are conducted on CIFAR-10 and CIFAR-100 datasets while the experiment settings remain the same as stated above. The results are shown in Fig. 5.

The momentum mechanism accumulates the gradients from past steps to determine the direction to go and thus converges faster than vanilla SGD as the figure shows. pbSGD, on the other hand, implements a nonlinear transformation function to gradients and improves the convergence by a large margin. Moreover, the Powerball function also stabilizes the training process while vanilla SGD suffers from the random perturbation introduced by stochasticity.

3.7 Improved Robustness to Batch Size

A large batch size in stochastic gradient methods can help increase the amount of computation per iteration and accelerate the training process. However, it is reported that increasing the batch size leads to a loss in generalization performance [Keskar *et al.*, 2016]. Here we observe empirically that the introduced Powerball function also improves the robustness to batch size choices.

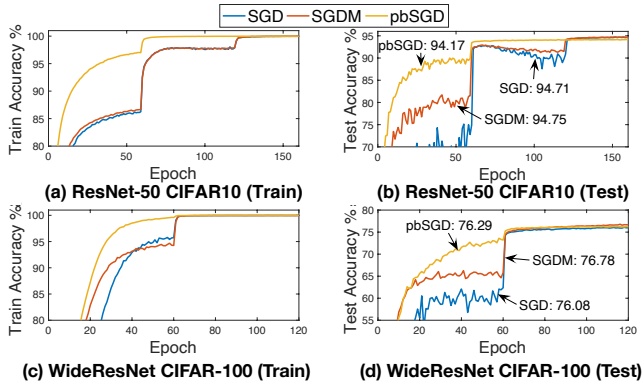

Figure 5. Train and test accuracy for different models and datasets. The annotations indicate the best overall test accuracy for pbSGD, SGDM and vanilla SGD respectively.

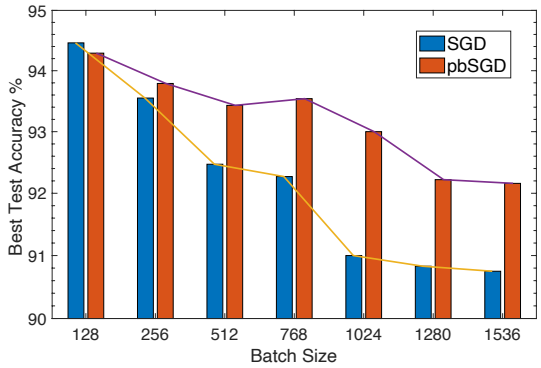

Figure 6. The best test accuracy for SGD and pbSGD under different batch sizes.

CIFAR-10 dataset and ResNet-50 are used for comparisons. The learning rate is set to 0.1 for SGD and pbSGD with $\gamma = 0.8$. Experiment settings remain as default and the best test accuracy is recorded. Fig. 6 shows the results. As we can observe, the generalization performance of SGD drops quickly with the increase of batch size while the best test accuracy of pbSGD is much stabler. It is hypothesized that the Powerball function affects the gradient distributions and compensates for part of the variance loss introduced by the larger batch size. The mechanism and theory behind this are yet to be studied in detail.

4 Convergence Analysis

In this section, we present convergence results of pbSGD and pbSGDM in the nonconvex setting. Given a vector $a \in \mathbb{R}^n$, we denote its i-th coordinate by a_i; we use $\|a\|$ to denote its 2-norm (Euclidean norm) and $\|a\|_p$ to denote its p-norm for $p \geq 1$. Given two vectors $a, b \in \mathbb{R}^n$, we use $a \cdot b$ to denote their inner product. We denote by $\mathbb{E}[\cdot]$ the expectation with respect to the underlying probability space. We start with some standard technical assumptions. First, we assume that the gradient of the objective function f is L-Lipschitz.

Assumption 4.1 *There exists some $L > 0$ such that $||\nabla f(x) - \nabla f(y)|| \leq L\|x - y\|$, for all $x, y \in \mathbb{R}^n$.*

We then assume that a stochastic first-order black-box oracle is accessible as a noisy estimate of the gradient of f at any point $x \in \mathbb{R}^n$, and the variance of the noise is bounded.

Assumption 4.2 *The stochastic gradient oracle gives independent and unbiased estimate of the gradient and satisfies:*

$$\begin{aligned} \mathbb{E}[g(x,\xi)] &= \nabla f(x), \\ \mathbb{E}[\|g(x,\xi) - \nabla f(x)\|^2] &\leq \hat{\sigma}^2, \quad \forall x \in \mathbb{R}^n, \end{aligned} \tag{5}$$

where $\hat{\sigma} \geq 0$ is a constant.

We will be working with a mini-batch size in the proposed pbSGD and pbSGDM. Let n_t be the mini-batch size at the t-th iteration and the corresponding mini-batch stochastic gradient be given by the average of n_t calls to the above oracle. Then by Assumption 4.2 we can show that $\mathbb{E}[\|g_t - \nabla f(x_t)\|^2] \leq \hat{\sigma}^2 / n_t$ for all $t \geq 1$. In other words, we can reduce variance by choosing a larger mini-batch size.

4.1 Convergence Analysis of pbSGD

We now state the main convergence result for the proposed pbSGD.

Theorem 4.1 *Suppose that Assumptions 4.1 and 4.2 hold. Let T be the number of iterations. pbSGD (3) with an adaptive learning rate and mini-batch size $B_t = T$ (independent of a particular step t) can lead to*

$$\mathbb{E}\left[\frac{1}{T}\sum_{t=1}^{T}\|g_t\|_{1+\gamma}^2\right] \leq \frac{2\|\mathbf{1}\|_p}{T}\left[\frac{L(f(x_1) - f^\star)}{1-\varepsilon} + \frac{\hat{\sigma}^2}{2\varepsilon(1-\varepsilon)}\right],$$

where $\varepsilon \in (0,1)$, $p = \frac{1+\gamma}{1-\gamma}$ for $\gamma \in [0,1)$ and $p = \infty$ for $\gamma = 1$.

Proof. By the L-Lipschitz continuity of ∇f and (3),

$$\begin{aligned} f(x_{t+1}) &\leq f(x_t) + \nabla f(x_t)\cdot(x_{t+1} - x_t) + \frac{L}{2}\|x_{t+1} - x_t\|^2 \\ &= f(x_t) - \alpha_t \nabla f(x_t)\cdot\sigma(g_t) + \frac{L}{2}\alpha_t^2\|\sigma(g_t)\|^2 \\ &= f(x_t) - \alpha_t g_t\cdot\sigma(g_t) + \frac{L}{2}\alpha_t^2\|\sigma(g_t)\|^2 \\ &\quad + \alpha_t(g_t - \nabla f(x_t))\cdot\sigma(g_t). \end{aligned}$$

Let $\alpha_t = \frac{g_t\cdot\sigma(g_t)}{L\|\sigma(g_t)\|^2} > 0$. Then

$$f(x_{t+1}) \leq f(x_t) - \frac{1}{2L}\frac{(g_t\cdot\sigma(g_t))^2}{\|\sigma(g_t)\|^2} + \alpha_t(g_t - \nabla f(x_t))\cdot\sigma(g_t). \tag{6}$$

Fix any iteration number $T > 1$ and let $\varepsilon \in (0,1)$ to be chosen. We can estimate

$$\begin{aligned} \alpha_t(g_t - \nabla f(x_t))\cdot\sigma(g_t) &= \frac{g_t\cdot\sigma(g_t)}{L\|\sigma(g_t)\|^2}(g_t - \nabla f(x_t))\cdot\sigma(g_t) \\ &\leq \frac{|g_t\cdot\sigma(g_t)|}{L\|\sigma(g_t)\|^2}\|g_t - \nabla f(x_t)\|\|\sigma(g_t)\| \\ &= \frac{|g_t\cdot\sigma(g_t)|}{L\|\sigma(g_t)\|}\|g_t - \nabla f(x_t)\| \\ &\leq \frac{1}{2L}\left(\varepsilon\frac{(g_t\cdot\sigma(g_t))^2}{\|\sigma(g_t)\|^2} + \frac{1}{\varepsilon}\|g_t - \nabla f(x_t)\|^2\right), \end{aligned}$$

where the last inequality followed from the elementary inequality $2ab \le \varepsilon a^2 + \frac{1}{\varepsilon} b^2$ for any positive real number ε and real numbers a, b. Substituting this into (6) gives

$$f(x_{t+1}) \le f(x_t) - \frac{1-\varepsilon}{2L} \frac{(g_t \cdot \sigma(g_t))^2}{\|\sigma(g_t)\|^2} + \frac{1}{2L\varepsilon} \|g_t - \nabla f(x_t)\|^2. \quad (7)$$

By Hölder's inequality, for $\gamma \in (0,1)$ and with $p = \frac{1+\gamma}{1-\gamma}$ and $q = \frac{1+\gamma}{2\gamma}$, we have

$$\begin{aligned} \|\sigma(g_t)\|^2 &= \sum_{i=1}^n |(g_t)_i|^{2\gamma} \le (\sum_{i=1}^n 1^p)^{\frac{1}{p}} (\sum_{i=1}^n (|(g_t)_i|^{2\gamma})^q)^{\frac{1}{q}} \\ &\le \|\mathbf{1}\|_p (\sum_{i=1}^n |(g_t)_i|^{1+\gamma})^{\frac{2\gamma}{1+\gamma}}. \end{aligned}$$

Hence

$$\begin{aligned} \frac{(g_t \cdot \sigma(g_t)^2}{\|\sigma(g_t)\|^2} &\ge \frac{(\sum_{i=1}^n |(g_t)_i|^{1+\gamma})^2}{\|\mathbf{1}\|_p (\sum_{i=1}^n |(g_t)_i|^{1+\gamma})^{\frac{2\gamma}{1+\gamma}}} \\ &= \frac{(\sum_{i=1}^n |(g_t)_i|^{1+\gamma})^{\frac{2}{1+\gamma}}}{\|\mathbf{1}\|_p} = \frac{\|g_t\|_{1+\gamma}^2}{\|\mathbf{1}\|_p}. \quad (8) \end{aligned}$$

Substituting this into (7) gives

$$f(x_{t+1}) \le f(x_t) - \frac{1-\varepsilon}{2L\|\mathbf{1}\|_p} \|g_t\|_{1+\gamma}^2 + \frac{1}{2L\varepsilon} \|g_t - \nabla f(x_t)\|^2.$$

Taking conditional expectation from both sizes gives

$$\begin{aligned} \mathbb{E}[f(x_{t+1}) - f(x_t) \,|\, x_t] &\le -\frac{1-\varepsilon}{2L\|\mathbf{1}\|_p} \mathbb{E}[\|g_t\|_{1+\gamma}^2] + \frac{1}{2L\varepsilon} \sigma_t^2 \\ &\le -\frac{1-\varepsilon}{2L\|\mathbf{1}\|_p} \mathbb{E}[\|g_t\|_{1+\gamma}^2] + \frac{\hat{\sigma}^2}{2L\varepsilon T}, \end{aligned}$$

where σ_t^2 is the variance of the t-th stochastic gradient approximation computed using the chosen mini-batch size $B_t = T$, which therefore satisfies $\sigma_t^2 \le \frac{\hat{\sigma}^2}{T}$. The conclusion of the theorem is obtained by taking expectation from both sides and performing a telescoping sum. □

Remark 4.1 *In the deterministic case, i.e. when (3) reduces to $x_{t+1} = x_t - \alpha_t \sigma(\nabla f(x_t))$, a similar (and simpler) argument will show that*

$$\frac{1}{T} \sum_{t=1}^T \|\nabla f(x_t)\|_{1+\gamma}^2 \le \frac{2L\|\mathbf{1}\|_p}{T} ((f(x_1) - f^\star).$$

4.2 Convergence Analysis of pbSGDM

We now present convergence analysis for pbSGDM.

Theorem 4.2 *Suppose that Assumptions 4.1 and 4.2 hold. Let T be the number of iterations. For any $\beta \in [0,1)$, pbSGDM (4) with an adaptive learning rate and mini-batch size $B_t = T$ (independent of a particular step t) can lead to*

$$\begin{aligned} &\mathbb{E}\left[\frac{1}{T}\sum_{t=1}^T \|g_t\|_{1+\gamma}^2\right] \\ \le &\frac{2\|\mathbf{1}\|_p}{T}\left[\frac{L(f(x_1)-f^\star)}{1-\varepsilon}\frac{1+\beta}{1-\beta} + \frac{\hat{\sigma}^2}{2\varepsilon(1-\varepsilon)}\right], \end{aligned}$$

where $\varepsilon \in (0,1)$, $p = \frac{1+\gamma}{1-\gamma}$ for $\gamma \in [0,1)$ and $p = \infty$ for $\gamma = 1$.

The proof relies on rewriting the pbSGMD scheme (4) as

$$\begin{cases} z_{t+1} = z_t - \dfrac{\alpha_t}{1-\beta}\sigma(g_t), \\ v_{t+1} = \beta v_t - \alpha_t \sigma(g_t), \end{cases} \quad (9)$$

where $z_t = x_t + \frac{\beta}{1-\beta} v_t$. A key technical lemma on estimating the sum $\sum_{t=1}^T \|v_t\|^2$ for (9) is stated below.

Lemma 4.1 *For $T \ge 1$, we have*

$$\sum_{t=1}^T \|v_t\|^2 \le \frac{1}{(1-\beta)^2} \sum_{t=1}^T \alpha_t^2 \|\sigma(g_t)\|^2.$$

Proof. It is easy to show by induction that, for $t \ge 1$,

$$v_t = -\sum_{i=1}^{t-1} \beta^{t-i-1} \alpha_i \sigma(g_i).$$

Indeed, we have $v_1 = 0$ and $v_2 = -\alpha_1 \sigma(g_1)$. Suppose that the above holds for $t \ge 1$. Then

$$\begin{aligned} v_{t+1} &= \beta v_t - \beta \alpha_t \sigma(g_t) \\ &= \beta \left(-\sum_{i=1}^{t-1} \beta^{t-i-1} \alpha_i \sigma(g_i) \right) - \beta \alpha_t g_t \\ &= -\sum_{i=1}^{t} \beta^{(t+1)-i-1} \alpha_i \sigma(g_i). \end{aligned}$$

Hence

$$\begin{aligned} \sum_{t=1}^T \|v_t\|^2 &= \sum_{t=1}^T \| - \sum_{i=1}^{t-1} \beta^{t-i-1} \alpha_i \sigma(g_i)\|^2 \\ &\le \sum_{t=1}^T \left(\sum_{i=1}^{t-1} \beta^{t-i-1} \|\alpha_i \sigma(g_i)\| \right)^2 \\ &\le \sum_{t=1}^T \sum_{i=1}^{t-1} \beta^{t-i-1} \sum_{i=1}^{t-1} \beta^{t-i-1} \|\alpha_i \sigma(g_i)\|^2 \\ &\le \frac{1}{1-\beta} \sum_{t=1}^T \sum_{i=1}^{t-1} \beta^{t-i-1} \|\alpha_i \sigma(g_i)\|^2 \\ &= \frac{1}{1-\beta} \sum_{i=1}^T \|\alpha_i \sigma(g_i)\|^2 \sum_{t=i+1}^T \beta^{t-i-1} \\ &\le \frac{1}{(1-\beta)^2} \sum_{t=1}^T \alpha_t^2 \|\sigma(g_t)\|^2. \end{aligned}$$

□

Proof of Theorem 4.2. By the L-Lipschitz continuity of ∇f and (9),

$$\begin{aligned} &f(z_{t+1}) \le f(z_t) + \nabla f(z_t) \cdot (z_{t+1} - z_t) + \frac{L}{2}\|z_{t+1} - z_t\|^2 \\ &= f(z_t) - \frac{\alpha_t}{1-\beta} \nabla f(z_t) \cdot \sigma(g_t) + \frac{L}{2} \frac{\alpha_t^2}{(1-\beta)^2} \|\sigma(g_t)\|^2 \\ &= f(z_t) - \frac{\alpha_t}{1-\beta} g_t \cdot \sigma(g_t) - \frac{\alpha_t}{1-\beta} (\nabla f(z_t) - \nabla f(x_t)) \cdot \sigma(g_t) \\ &\quad + \frac{\alpha_t}{1-\beta} (g_t - \nabla f(x_t)) \cdot \sigma(g_t) + \frac{L}{2} \frac{\alpha_t^2}{(1-\beta)^2} \|\sigma(g_t)\|^2. \end{aligned} \quad (10)$$

We have

$$-\frac{\alpha_t}{1-\beta}(\nabla f(z_t)-\nabla f(x_t))\cdot\sigma(g_t) \le \frac{1}{2(1-\beta)}\left[\varepsilon_1\|\nabla f(z_t)-\nabla f(x_t)\|^2+\frac{1}{\varepsilon_1}\alpha_t^2\|\sigma(g_t)\|^2\right], \tag{11}$$

where $\varepsilon_1>0$ is to be chosen. By the L-Lipschitz continuity of ∇f, we have

$$\|\nabla f(z_t)-\nabla f(x_t)\|^2 \le L^2\|z_t-x_t\|^2 = L^2\|\frac{\beta}{1-\beta}v_t\|^2 = L^2\frac{\beta^2}{(1-\beta)^2}\|v_t\|^2. \tag{12}$$

We can also bound

$$\frac{\alpha_t}{1-\beta}(g_t-\nabla f(x_t))\cdot\sigma(g_t) \le \frac{1}{2}\left[\frac{1-\beta}{L\varepsilon(1+\beta)}\|g_t-\nabla f(x_t)\|^2+L\varepsilon\frac{1+\beta}{(1-\beta)^3}\alpha_t^2\|\sigma(g_t)\|^2\right], \tag{13}$$

where $\varepsilon>0$. By inequalities (11)-(13), Lemma 4.1, and a telescoping sum on (10), we get

$$f^\star-f(z_1) \le -\frac{1}{1-\beta}\sum_{t=1}^T\alpha_t g_t\cdot\sigma(g_t) + \frac{1}{2}\left[\frac{\varepsilon_1L^2\beta^2}{(1-\beta)^5}+\frac{1}{\varepsilon_1(1-\beta)}+\frac{L}{(1-\beta)^2}+\frac{L\varepsilon(1+\beta)}{(1-\beta)^3}\right]\times \sum_{t=1}^T\alpha_t^2\|\sigma(g_t)\|^2+\frac{1-\beta}{2L\varepsilon(1+\beta)}\sum_{t=1}^T\|g_t-\nabla f(x_t)\|^2.$$

Setting $\varepsilon_1=\frac{(1-\beta)^2}{L\beta}$ and choosing $\alpha_t=\frac{g_t\cdot\sigma(g_t)}{L\|\sigma(g_t)\|^2}\frac{(1-\beta)^2}{1+\beta}$ lead to

$$f^\star-f(z_1) \le -\frac{(1-\beta)(1-\varepsilon)}{2L(1+\beta)}\sum_{t=1}^T\frac{(g_t\cdot\sigma(g_t))^2}{\|\sigma(g_t)\|^2} + \frac{1-\beta}{2L\varepsilon(1+\beta)}\sum_{t=1}^T\|g_t-\nabla f(x_t)\|^2.$$

which, by taking expectation from both sides and the same estimate (8) as in the proof for Theorem 4.1, leads to

$$f^\star-f(z_1) \le -\frac{(1-\beta)(1-\varepsilon)}{2L\|\mathbf{1}\|_p(1+\beta)}\mathbb{E}\left[\sum_{t=1}^T\|g_t\|_{1+\gamma}^2\right]+\frac{\hat\sigma^2(1-\beta)}{2L\varepsilon(1+\beta)}.$$

The proof is complete by noting $z_1=x_1$. □

Remark 4.2 *Consider the deterministic version of pbSGDM:*

$$\begin{cases} v_{t+1}=\beta v_t-\alpha_t\sigma(\nabla f(x_t)),\\ x_{t+1}=x_t+v_{t+1},\end{cases} \tag{14}$$

where $\beta\in[0,1)$ is a momentum constant and $v_0=0$. A similar proof can show that the deterministic version of pbSGDM above converges according to:

$$\frac{1}{T}\sum_{t=1}^T\|\nabla f(x_t)\|_{1+\gamma}^2\le\frac{2L\|\mathbf{1}\|_p}{T}\frac{1+\beta}{1-\beta}(f(x_1)-f^\star).$$

Remark 4.3 *A careful reader will notice that our estimates of convergence rates for pbSGD and pbSGMD in Theorems 4.1 and 4.2, respectively, are in terms of the stochastic gradients g_t. This is without loss of generality in view of Assumption 4.2, because one can show that $\mathbb{E}[\|\nabla f(x_t)\|_{1+\gamma}^2]\le\frac{D_\gamma}{C_\gamma}\mathbb{E}[\|g_t\|_{1+\gamma}^2]$, where C_γ and D_γ are positive constants depending on γ. In other words, the estimates are equivalent (modulo a constant factor).*

Remark 4.4 *Convergence analysis of stochastic momentum methods for nonconvex optimization is an important but under-explored topic. While our results on convergence analysis do not improve the rate of convergence for stochastic momentum methods in a nonconvex setting, it does match the currently best known rate of convergence [Yan* et al.*, 2018; Bernstein* et al.*, 2018] in special cases ($\gamma=0$ and 1) and offers very concise upper bounds in terms of the constants. The upper bound continuously interpolates the convergence rate for γ varying in $[0,1]$ and β varying in $[0,1)$, which can potentially be attributed to Lemma 4.1, which provides a tight estimate of accumulated momentum terms v_t. We also note that the convergence rates for $\gamma\in(0,1)$ are entirely new and not reported elsewhere before. Even for the special cases ($\gamma=0$ and 1), our proof differs from that of [Yan* et al.*, 2018; Bernstein* et al.*, 2018] and seems more transparent.*

Remark 4.5 *A large mini-batch ($B_t=T$) is assumed for the theoretical convergence results to hold. This is consistent with the convergence analysis in [Bernstein* et al.*, 2018] for the special case $\gamma=0$. We assume this because it enables us to put analysis of PoweredGD and pbSGD in a unified framework so that we can obtain tighter bounds. In practice, a mini-batch size can be fixed as in all the experiments presented in this paper (we used 128 for all experiments, except for 256 in the ImageNet experiments). We also note that our proof requires T^2 gradient calls in T iterations and hence the effective convergence rate is $O(1/\sqrt{T})$, which is consistent with the known rate of convergence for SGD [Ge* et al.*, 2015].*

5 Conclusion

In this paper, we present a simple but universal technique to improve SGD and SGDM for deep neural network training. The main results of the paper demonstrated that this technique has the potential to accelerate initial convergence, improve robustness with hyper-parameter selection, and mitigate the gradient vanishing problem. Moreover, the technique is complementary to other techniques such as learning rate schedules for further improving the convergence and generalization as shown in our preliminary experiments. Our convergence analysis matches the currently best bounds on convergence rates of SGD and SGDM for nonconvex functions. Potential areas of future work include the theoretical justification of the benefits of using pbSGD, adaptive techniques for choosing the hyper-parameter γ, and applications of pbSGD in deep learning.

Acknowledgments

JL acknowledges funding from the NSERC DG, CRC, and ERA programs of Canada.

References

[Bello *et al.*, 2017] Irwan Bello, Barret Zoph, Vijay Vasudevan, and Quoc V Le. Neural optimizer search with reinforcement learning. In *Proc. of ICML*, pages 459–468, 2017.

[Bernstein *et al.*, 2018] Jeremy Bernstein, Yu-Xiang Wang, Kamyar Azizzadenesheli, and Anima Anandkumar. signSGD: Compressed optimisation for non-convex problems. *arXiv:1802.04434*, 2018.

[Chen *et al.*, 2018] Xiangyi Chen, Sijia Liu, Ruoyu Sun, and Mingyi Hong. On the convergence of a class of Adam-type algorithms for non-convex optimization. *arXiv:1808.02941*, 2018.

[Duchi *et al.*, 2011] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul):2121–2159, 2011.

[Fang *et al.*, 2019] Cong Fang, Zhouchen Lin, and Tong Zhang. Sharp analysis for nonconvex SGD escaping from saddle points. *arXiv:1902.00247*, 2019.

[Ge *et al.*, 2015] Rong Ge, Furong Huang, Chi Jin, and Yang Yuan. Escaping from saddle points—online stochastic gradient for tensor decomposition. In *Proc. of COLT*, pages 797–842, 2015.

[He *et al.*, 2016] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proc. of CVPR*, 2016.

[Huang *et al.*, 2017] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proc. of CVPR*, pages 4700–4708, 2017.

[Jin *et al.*, 2019] Chi Jin, Praneeth Netrapalli, Rong Ge, Sham M Kakade, and Michael I Jordan. Stochastic gradient descent escapes saddle points efficiently. *arXiv:1902.04811*, 2019.

[Keskar *et al.*, 2016] 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. *arXiv preprint arXiv:1609.04836*, 2016.

[Kingma and Ba, 2015] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *Proc. of ICLR*, 2015.

[Krizhevsky and Hinton, 2009] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.

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

[Loshchilov and Hutter, 2016] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. *arXiv:1608.03983*, 2016.

[Luo *et al.*, 2018] Liangchen Luo, Wenhao Huang, Qi Zeng, Zaiqing Nie, and Xu Sun. Learning personalized end-to-end goal-oriented dialog. *arXiv:1811.04604*, 2018.

[Mazumdar and Ratliff, 2018] Eric Mazumdar and Lillian J. Ratliff. On the convergence of gradient-based learning in continuous games. *arXiv:1804.05464*, 2018.

[Nesterov, 1983] Yurii E Nesterov. A method for solving the convex programming problem with convergence rate $o(1/k^2)$. In *Dokl. akad. nauk Sssr*, volume 269, pages 543–547, 1983.

[Polyak, 1964] Boris T Polyak. Some methods of speeding up the convergence of iteration methods. *USSR Computational Mathematics and Mathematical Physics*, 4(5):1–17, 1964.

[Qian, 1999] Ning Qian. On the momentum term in gradient descent learning algorithms. *Neural Networks*, 12(1):145–151, 1999.

[Reddi *et al.*, 2018] Sashank J Reddi, Satyen Kale, and Sanjiv Kumar. On the convergence of Adam and beyond. In *Proc. of ICLR*, 2018.

[Robbins and Monro, 1951] Herbert Robbins and Sutton Monro. A stochastic approximation method. *The Annals of Mathematical Statistics*, 22(3):400–407, 1951.

[Russakovsky *et al.*, 2015] Olga Russakovsky, Jia Deng, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.

[Smith and Topin, 2017] Leslie N Smith and Nicholay Topin. Super-convergence: Very fast training of neural networks using large learning rates. *arXiv:1708.07120*, 2017.

[Tieleman and Hinton, 2012] Tijmen Tieleman and Geoffrey Hinton. Lecture 6.5-RMSProp: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural Networks for Machine Learning*, 4(2):26–31, 2012.

[Wilson *et al.*, 2017] Ashia C Wilson, Rebecca Roelofs, Mitchell Stern, Nati Srebro, and Benjamin Recht. The marginal value of adaptive gradient methods in machine learning. In *Proc. of NeurIPS*, pages 4148–4158, 2017.

[Wu and He, 2018] Yuxin Wu and Kaiming He. Group normalization. In *Proc. of ECCV*, pages 3–19, 2018.

[Xie *et al.*, 2017] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *Proc. of CVPR*, pages 1492–1500, 2017.

[Yan *et al.*, 2018] Yan Yan, Tianbao Yang, Zhe Li, Qihang Lin, and Yi Yang. A unified analysis of stochastic momentum methods for deep learning. *Proc. of IJCAI*, 2018.

[Yang *et al.*, 2019] Tao Yang, Xinlei Yi, Junfeng Wu, Ye Yuan, Di Wu, Ziyang Meng, Yiguang Hong, Hong Wang, Zongli Lin, and Karl H. Johansson. A survey of distributed optimization. *Annual Reviews in Control*, 47:278 – 305, 2019.

[Yuan *et al.*, 2019] Ye Yuan, Mu Li, Jun Liu, and Claire Tomlin. On the Powerball method: variants of descent methods for accelerated optimization. *IEEE Control Systems Letters*, 3(3):601–606, 2019.

[Zagoruyko and Komodakis, 2016] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. *arXiv:1605.07146*, 2016.