

New logarithmic step size for stochastic gradient descent

Mahsa Soheil SHAMAE¹, Sajad Fathi HAFSHEJANI (✉)², Zeinab SAEIDIAN³

¹ Department of Computer Science, Faculty of Mathematical Science, University of Kashan, Kashan 87317-53153, Iran

² Department of Applied Mathematics, Shiraz University of Technology, Shiraz 13876-71557, Iran

³ Department of Mathematical Sciences, University of Kashan, Kashan 87317-53153, Iran

© Higher Education Press 2025

Abstract In this paper, we propose a novel warm restart technique using a new logarithmic step size for the stochastic gradient descent (SGD) approach. For smooth and non-convex functions, we establish an $O\left(\frac{1}{\sqrt{T}}\right)$ convergence rate for the SGD. We conduct a comprehensive implementation to demonstrate the efficiency of the newly proposed step size on the FashionMinst, CIFAR10, and CIFAR100 datasets. Moreover, we compare our results with nine other existing approaches and demonstrate that the new logarithmic step size improves test accuracy by 0.9% for the CIFAR100 dataset when we utilize a convolutional neural network (CNN) model.

Keywords stochastic gradient descent, logarithmic step size, warm restart technique

1 Introduction

Stochastic gradient descent (SGD), which dates back to the work by Robbins and Monro [1] is widely observed in training modern Deep Neural Networks (DNNs), which are widely used to achieve state-of-the-art results in multiple problem domains like image classification problems [2,3], object detection [4], and classification automatic machine translation [5].

The value of the step size (or learning rate) is crucial for the convergence rate of SGD. Selecting an appropriate step size value in each iteration ensures that SGD iterations converge to an optimal solution. If the step size value is too large, it may prevent SGD iterations from reaching the optimal point. Conversely, excessively small step size values can lead to slow convergence or mistakenly identify a local minimum as the optimal solution [6]. To address these challenges, various schemes have been proposed. One popular approach is the Armijo line search method, initially introduced for SGD by Vaswani et al. [7], which provides theoretical results for strong-convex, convex, and non-convex objective functions. Gower et al. [8] proposed another approach that combines a constant step size with a decreasing step size. Their algorithm starts with a constant step size and switches to a decreasing step size after $4k$ iterations, where k denotes the problem's

condition number. This strategy guarantees convergence for strongly convex functions but requires knowledge of the condition number and is not applicable to non-convex functions. Additionally, the decay step size is commonly used in practice for solving non-convex problems encountered during DNN training [2,9].

Smith in [10] proposed an efficient method for setting the step size known as the cyclical learning rate. Training neural networks with cyclical learning rates instead of fixed learning rates can lead to significant improvements in accuracy without the need for manual tuning, often requiring fewer iterations. Loshchilov and Hutter [11] utilized this strategy and proposed a *warm restart* technique for SGD that does not need to apply gradient information for updating step size in each iteration. The key idea behind warm restarts is that in each restart, the learning rate is initialized to a specific value (denoted as η_0) and scheduled to decrease. Moreover, it has been shown that a warm restarted SGD takes 2–4 times less time than a traditional learning rate adjustment strategy [12]. In recent years, numerous step sizes with warm restarts have been introduced [6,13]. Vrbančič extended the idea of the cosine step size and proposed three different step sizes with warm restarts [12].

The convergence rate, from a theoretical standpoint, is a crucial metric for assessing the effectiveness and efficiency of an algorithm. In the case of the SGD algorithm, the convergence rate depends on various factors, such as the type of objective function (e.g., strong convex, convex, or non-convex) and the value of the step size used in each iteration. For convex and smooth functions with Lipschitz continuous gradients, it has been proven that the SGD algorithm can achieve a convergence rate of $O\left(\frac{1}{\sqrt{T}}\right)$ [14,15]. The rate of convergence for a strongly convex function is $O\left(\frac{1}{T}\right)$ [16,17]. When the goal is smooth and non-convex functions, Ghadimi and Lan [15] provided an $O\left(\frac{1}{\sqrt{T}}\right)$ rate of convergence for SGD by a constant step size, i.e., $\eta_t = \frac{1}{\sqrt{T}}$. In another work, Li et al. [18] established an $O\left(\frac{1}{\sqrt{T}}\right)$ rate of convergence for both cosine and exponential step sizes which were first considered in [11]. Moreover, their empirical results confirm that both step sizes have the best accuracy and training loss

achievements. Ge et al. [19] considered decay step size for solving the least squares problems. Wang et al. [20] took into account the influence of the probability distribution, denoted as $\frac{\eta_t}{\sum_{i=1}^T \eta_i}$, on both the implementation and theoretical aspects of the algorithm. They demonstrated that within this framework, the SGD algorithm with the exponential step size achieves a convergence rate of $O\left(\frac{\ln T}{\sqrt{T}}\right)$. Notably, their findings reveal that assigning a higher probability to the final iterations through the step size leads to enhancements in accuracy and improvements in the loss function.

In the context of many decay step sizes, the convergence rate of the SGD algorithm is commonly analyzed by considering specific values for the parameters associated with the step size. Several studies, including [15,18,20], have explored and derived convergence rates for the SGD algorithm by imposing constraints or assumptions on the initial step size or the parameters related to the step size. These conditions play a crucial role in ensuring the convergence properties of the algorithm. Table 1 presents the convergence rates of various decay step sizes. Specifically, the second column of the table indicates the conditions that need to be satisfied by the initial step size η_0 or the parameters associated with the step size in order to achieve the desired convergence properties. As presented in Table 1, to achieve a convergence rate of order $O\left(\frac{1}{\sqrt{T}}\right)$, Li et al. [18] made the assumption $\eta_0 = \frac{1}{Lc}$ and $c \propto O(\sqrt{T})$. This assumption implies that the value of the initial step size, denoted as η_0 , will be very small. It is evident that for large values of T , the initial step size will be extremely tiny, potentially impacting the algorithm's efficiency.

1.1 Contribution

Motivated by the aforementioned studies, our work introduces a novel logarithmic step size for stochastic gradient descent (SGD) with the warm restarts technique. The main contributions of this paper can be summarized as follows:

- The new proposed step size offers a significant advantage over the cosine step size [18] in terms of its probability distribution, denoted as $\frac{\eta_t}{\sum_{i=1}^T \eta_i}$ in Theorem 1. This distribution plays a crucial role in determining the likelihood of selecting a specific output during the iterations. Fig. 1(a) illustrates that the cosine step size assigns a higher probability of selection to the initial iterations but substantially reduces the probability for

Table 1 Convergence rates of SGD based on the specific values for the initial step size or the parameters associated with the step size

Step sizes	Conditions	Convergence rate	Reference
$\eta_t = \frac{\eta_0}{2} \left(1 + \cos\left(\frac{\pi t}{T}\right)\right)$	$\eta_0 = \frac{1}{Lc}, c = O(\sqrt{T})$	$O\left(\frac{1}{\sqrt{T}}\right)$	[18]
$\eta_t = \eta_0 \left(\frac{\beta}{T}\right)^{\frac{t}{T}}$	$\eta_0 = \frac{1}{Lc}, c = O(\sqrt{T})$	$O\left(\frac{1}{\sqrt{T}}\right)$	[18]
$\eta_t = \eta_0 \left(\frac{\beta}{T}\right)^{\frac{t}{T}}$	$\beta = O(\sqrt{T})$	$O\left(\frac{\ln T}{\sqrt{T}}\right)$	[20]
$\eta_t = \text{Constant}$	$\eta_0 = \frac{1}{\sqrt{T}}$	$O\left(\frac{1}{\sqrt{T}}\right)$	[15]
New proposed method	$\eta_0 = \frac{1}{Lc}, c = O\left(\frac{\sqrt{T}}{\ln T}\right)$	$O\left(\frac{1}{\sqrt{T}}\right)$	New

the final iterations, approaching zero. In contrast, the new step size proves to be more effective for the final iterations, as they enjoy a higher probability of selection compared to the cosine step size. Consequently, the new step size method outperforms the cosine step size method when it comes to the final iterations, benefiting from their increased likelihood of being chosen as the selected solution.

- For the new step size, we establish the convergence results of the SGD algorithm. By considering that $c \propto O\left(\frac{\sqrt{T}}{\ln T}\right)$, which leads to the initial value of the step size is greater than the initial value of the step length mentioned in [18], we demonstrate a convergence rate of $O\left(\frac{1}{\sqrt{T}}\right)$ for a smooth non-convex function. This convergence rate meets the best-known rate of convergence for smooth non-convex functions.
- We conduct a comparative experimental analysis of the new logarithmic step size to nine popular step sizes, including Armijo line search [7], cosine step size [18], Adam, $\frac{1}{t}$, $\frac{1}{\sqrt{t}}$, constant step size, reduceLRonPlateau, stagewise - 1 milestone, stagewise - 2 milestones. We compare the performance of our proposed approach with that of the state-of-the-art methods on the FashionMinst, CIFAR10, and CIFAR100 datasets. We demonstrate the effectiveness of the newly proposed step size in enhancing the accuracy and training loss of the SGD on popular datasets such as FashionMNIST, CIFAR10, and CIFAR100 (e.g., see Fig. 1(c)). Notably, we observe a significant accuracy improvement of 0.9% specifically for the CIFAR100 dataset when we use a convolutional neural network model.

The remaining sections of this paper are structured as follows: In Section 2, we introduce the new logarithmic step size and discuss its properties. Section 3 is dedicated to the analysis of convergence rates for the proposed step size on smooth non-convex functions. We establish that the SGD method achieves an impressive convergence rate of $O\left(\frac{1}{\sqrt{T}}\right)$. In Section 4, we present and discuss the numerical results obtained using the new decay step size. Finally, in Section 5, we summarize our findings and draw conclusions based on our study.

In this paper, we use the following notational conventions:

- The Euclidean norm of a vector is denoted by $\|\cdot\|$.
- The nonnegative orthant and positive orthant of \mathbb{R}^n are denoted by \mathbb{R}_+^n and \mathbb{R}_{++}^n , respectively.
- We use the notation $f(t) = O(g(t))$ to indicate that there exists a positive constant ω such that $f(t) \leq \omega g(t)$ for all $t \in \mathbb{R}_{++}$.

2 Problem set-up

We consider the following optimization problem:

$$\min_{x \in \mathbb{R}^d} f(x) = \min_{x \in \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^n f_i(x), \quad (1)$$

where $f_i: \mathbb{R}^d \rightarrow \mathbb{R}$ is the loss function for the i th training

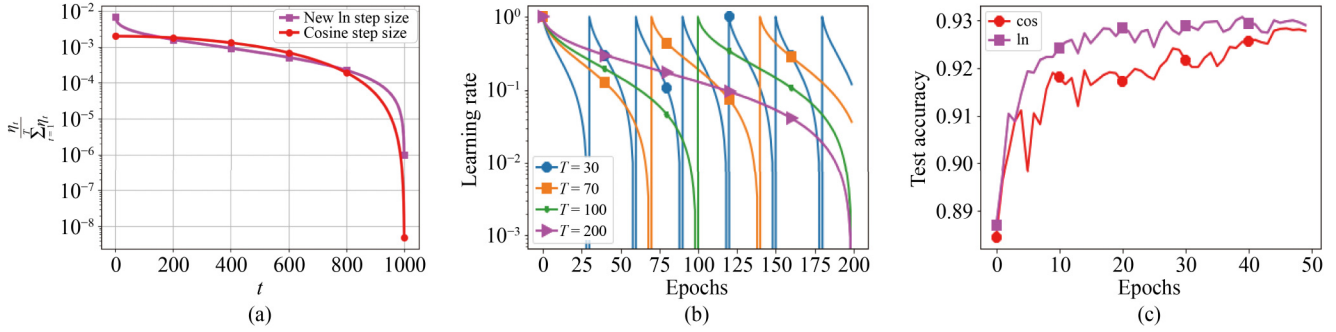


Fig. 1 (a) Representation of the value $\frac{\eta_t}{\sum_{i=1}^T \eta_i}$ for both the new step size and cosine step size in [18]; (b) warm restarts simulated every $T = 30$ (blue line), $T = 70$ (orange line), $T = 100$ (green line), and $T = 200$ (magenta line) epochs with $\eta_0 = 1$; (c) the comparison of the warm restart Algorithm 1 with the new proposed step size and cosine step size on the FashionMnist dataset

sample over the variable $x \in \mathbb{R}^d$ and n denotes the number of samples. This minimization problem is central in machine learning. Several iterative approaches for solving Eq. (1) are known [21], and SGD is particularly popular when the dimensionality, n , is extremely large. SGD uses a random training sample $i_k \in \{1, 2, \dots, n\}$ to update x using the rule:

$$x_{k+1} = x_k - \eta_k \nabla f_{i_k}(x_k), \quad (2)$$

in which η_k is the step size used in iteration k and $\nabla f_{i_k}(x)$ is the (average) gradient of the loss function(s) [7].

Our analysis and results in this paper are based on the following assumptions.

- **Assumption 1** The function f is differentiable and L -smooth, that is, for all $x, y \in \mathbb{R}^d$ there exists a constant $L > 0$ such that:

$$\begin{aligned} \|\nabla f(x) - \nabla f(y)\| &\leq L\|x - y\|, \\ f(y) &\leq f(x) + \langle \nabla f(x), y - x \rangle + \frac{L}{2}\|y - x\|^2. \end{aligned}$$

- **Assumption 2** For any random $t \in [1, T]$, we have

$$\mathbb{E}_t [\|g_t - \nabla f(x_t)\|^2] \leq \sigma^2. \quad (3)$$

- **Assumption 3** The objective function f is bounded below on \mathbb{R}^d .

2.1 The new step size

It has been observed that larger values of the step size provide the model with sufficient energy to escape from critical points in the initial iterations, while smaller step sizes guide the model towards well-behaved local minima in the final iterations [6]. However, when the step length of a function

takes on large values over multiple iterations and rapidly tends to zero towards the end, it adversely affects the behavior of the distribution function denoted as $\frac{\eta_t}{\sum_{i=1}^T \eta_i}$. Consequently, based on the conditions described in Theorem 1, the probability of selecting points from the final iterations decreases significantly compared to the initial iterations. To address these issues, we propose an appropriate step size that its values gradually decrease during the iterations. In this context, we introduce a logarithmic step size that exhibits slower convergence to zero compared to many other step sizes, yet converges faster than the cosine step size. The new step size is defined as follows:

Definition 1 For a constant η_0 , the new logarithmic step size for SGD is given by

$$\eta_t = \eta_0 \left(1 - \frac{\ln t}{\ln T}\right), \quad t \in [1, T], \quad (4)$$

where T represents the number of iterations in each inner loop of the algorithm and t represents the number of epochs performed since the last restart. To use the warm restarts policy, we follow the model proposed in [10], which states that the training process is divided into l cycles (i.e., the outer loop of Algorithm 1), and the algorithm runs each cycle in T epochs. Each cycle begins with the largest value of step size η_0 . During the running of a cycle, the value of the learning rate decreases by using Eq. (4). In addition, when $t = T$ the step size η will be output, i.e., $\eta_T = 0$. The behaviours of the new proposed step size for different values of T are shown in Fig. 1(b). It is clear that for small values of T , that is, $T = \{30, 70\}$, the algorithm performs more inner loop, and the

Algorithm 1 SGD with warm restarts logarithmic step size

- 1 Input: Initial Step size η_0 , initial point x_0^1 , the number of outer and inner iterations, i.e., l and T .
 - 2 Set: $T_0 = T$
 - 3 **for** $i = 0, \dots, l$ **do**
 - 4 **for** $t = 1, \dots, T$ **do**
 - 5 Set $\eta_t = \eta_0 \left(1 - \frac{\ln t}{\ln T}\right)$ and update x_i^{t+1} .
 - 6 **End**
 - 7 $x_{i+1}^1 = x_i^T$
 - 8 **End**
 - 9 $x^* = x_i^T$
-

values of step size quickly go to zeros, but when $T = 200$, the algorithm performs one inner loop and the step size goes to zeros very slowly.

The lemma that follows gives us some properties of the function $\ln x$. We will utilize them frequently in the rest of the paper.

Lemma 1 For the function $\ln x$, we have

- $\ln xy = \ln x + \ln y$.
- $\ln \frac{x}{y} = \ln x - \ln y$.
- $\frac{x-1}{x} \leq \ln x$, for all $x \geq 1$.

3 Algorithm and convergence

We use the warm restart Algorithm 1 with condition $T_0 = T_1 = \dots = T_l$ which means the algorithm runs with the same number of epochs in the inner loop. Algorithm 1 begins with the given initial step size η_0 , the number of inner iterations T , the number of outer epochs l , and the starting point x_0^1 . The algorithm consists of two loops, i.e., outer and inner loops. In each inner loop, the SGD with the logarithmic step size is performed.

3.1 Convergence results for smoothness function

We present the convergence bound for SGD with logarithmic step sizes when the objective function is a smooth and non-convex function. The lemma that follows provides a lower bound for $\sum_{t=1}^T \eta_t$.

Lemma 2 For the new proposed step size given by Eq. (4), we have

$$\sum_{t=1}^T \eta_t \geq \eta_0 \frac{T+1}{2 \ln T}.$$

The next lemma gives an upper bound for $\sum_{t=1}^T \eta_t^2$.

Lemma 3 For the step size given by Eq. (4), we have

$$\sum_{t=1}^T \eta_t^2 \leq \eta_0^2 \frac{2T}{\ln^2 T}. \quad (5)$$

Lemma 4 Under Assumptions 1 and 2, and $\eta_t \leq \frac{1}{cL}$. Then, Algorithm 1 with the new proposed step size guarantees:

$$\frac{\eta_t}{2} \mathbb{E}[\|\nabla f(x_t)\|^2] \leq \mathbb{E}[f(x_t)] - \mathbb{E}[f(x_{t+1})] + \frac{L\eta_t^2 \sigma^2}{2}.$$

The next theorem gives an upper bound for $\mathbb{E}\|\nabla f(\bar{x}_T)\|^2$.

Theorem 1 Under Assumptions 1 and 2, and $c > 1$. SGD with the new proposed step sizes with $\eta_0 = \frac{1}{cL}$ guarantees:

$$\begin{aligned} & \mathbb{E}\|\nabla f(\bar{x}_T)\|^2 \\ & \leq \frac{4cL \ln T}{(T+1)} (f(x_1) - f^*) + \frac{4\sigma^2 T}{Lc(T+1) \ln T}, \end{aligned}$$

where \bar{x}_T is a random iterate drawn from the sequence $\{x_t\}_{t=1}^T$ with probability $\mathbb{P}[\bar{x}_T = x_t] = \frac{\eta_t}{\sum_{t=1}^T \eta_t}$.

Proof Using the definition of \bar{x}_T and Lemma 4, we have

$$\begin{aligned} \mathbb{E}[\|\nabla f(\bar{x}_T)\|^2] &= \frac{\eta_t \mathbb{E}[\|\nabla f(x_t)\|^2]}{\sum_{t=1}^T \eta_t} \\ &\leq \frac{2 \sum_{t=1}^T [\mathbb{E}[f(x_t)] - \mathbb{E}[f(x_{t+1})]]}{\sum_{t=1}^T \eta_t} \\ &+ \frac{L\sigma^2 \sum_{t=1}^T \eta_t^2}{\sum_{t=1}^T \eta_t} \\ &\leq \frac{2(f(x_1) - f^*)}{\sum_{t=1}^T \eta_t} + \frac{L\sigma^2 \sum_{t=1}^T \eta_t^2}{\sum_{t=1}^T \eta_t} \\ &\leq \frac{4 \ln T (f(x_1) - f^*)}{\eta_0(T+1)} + \frac{4L\sigma^2 \eta_0 T}{(T+1) \ln T} \\ &= \frac{4cL \ln T}{(T+1)} (f(x_1) - f^*) + \frac{4\sigma^2 T}{Lc(T+1) \ln T}, \end{aligned}$$

where the third inequality is obtained by using Lemmas 2 and 3. \square

As mentioned earlier, selecting a smaller value for the parameter c compared to the cosine function used in [18] enables us to achieve the optimal convergence rate for SGD based on the new logarithmic step size. Specifically, when considering $c \propto O\left(\frac{\sqrt{T}}{\ln T}\right)$, we have the following result:

Corollary 1 Based on the Theorem 1, if $\sigma \neq 0$, setting $c \propto \frac{\sqrt{T}}{\ln T}$, we have

$$\mathbb{E}\|\nabla f(\bar{x}_T)\|^2 \leq \frac{4L}{\sqrt{T}} (f(x_1) - f^*) + \frac{4\sigma^2}{L\sqrt{T}}.$$

Now, leveraging the results obtained from Theorem 1, we can compute the convergence rate for the warm restart SGD algorithm.

Corollary 2 (SGD with warm restarts): Under Assumptions A1, A2, and A3, for a given value of T , where $\sum_{i=1}^l T_i = lT$ and $\eta_0 = \frac{1}{cL}$, Algorithm 1 ensures the following convergence guarantees:

$$\mathbb{E}\|\nabla f(\bar{x}_T)\|^2 \leq \frac{4lcL \ln T}{T} (f(\bar{x}_1) - f^*) + \frac{4\sigma^2 l}{Lc \ln T},$$

where $f(\bar{x}_1) = \max_i \{f(x_{1_i})\}$ for $i = 1, 2, \dots, l$.

4 Numerical results

4.1 Experiments on MNIST, CIFAR10 and CIFAR100

In this section, we evaluate our proposed algorithm's performance on image classification tasks. We compare the performance of the proposed approach with that of the state-of-the-art methods on the FashionMinst, CIFAR10, and CIFAR100 datasets [3]. FashionMinst is a dataset consisting of a training set of 60000 examples of grayscale images and a test set of 10000 examples. Each image is 28×28 pixels. We use the convolutional neural network (CNN) model for the classification task on this dataset. This model has two convolutional layers with kernel sizes of 5×5 and padding of 2, two max-pooling layers with kernel sizes of 2×2 , and two fully connected layers with 1024 hidden nodes. The activation function of the hidden nodes is the rectified linear unit

(ReLU). To prevent overfitting, we use a dropout with a probability of 0.5 in the hidden layer of the deep model. The number of input neurons is 746, and the number of output neurons is 10. To compare the performance of various algorithms, we use the cross-entropy function as a loss function and an accuracy metric.

The CIFAR10 dataset consists of 60,000 of 32×32 color images of 10 classes, each with 6000 images. There are 50000 training images and 10000 test images. The batch size is 128. It means that each epoch of the training includes 390 iterations. The deep learning architecture used for evaluating the performance of the algorithms on this dataset is the 20-layer residual neural network (ResNet) [22]. This model uses cross-entropy as the loss function.

The CIFAR100 dataset is just like the CIFAR10, except it has 100 classes containing 600 natural images each. There are 500 training images and 100 testing images per class. There are 50000 images available for training and 10000 images available for testing. Randomly cropped and flipped images are used for training. We have conducted two series of experiments on this dataset based on two deep learning models. The first one is the same as that used on FashionMnist which is described above. The reason for using this model may be the limitation in the hardware available to run the experiments. The second one is a DenseNet-BC model with 100 layers and a growth rate of 12 [9].

To optimize the hyperparameter η_0 of the new proposed step decay on the CIFAR100 dataset, we employed a two-stage grid search on the validation set. Initially, we explored a rough grid and picked the hyperparameters that produced the best validation outcomes. Subsequently, we conducted a more refined search centered around the most effective hyperparameters identified in the first stage and chose the optimal set of hyperparameters as the final selection. For the starting step size η_0 , we used a coarse search grid of $\{0.00001, 0.0001, 0.001, 0.01, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1\}$, and a fine grid of $\{0.41, 0.42, 0.43, 0.44, 0.45, 0.46, 0.47, 0.48, 0.49, 0.51, 0.52, 0.53, 0.54, 0.55, 0.56, 0.57, 0.58, 0.59\}$. The best value of η_0 was obtained as 0.53. We compared the performance of our proposed method with state-of-the-art methods whose hyperparameter values were fine-tuned in a previous study [18]. We adopted the same hyperparameter values in our own numerical studies. The values of the hyperparameters are given in Table 2. In the experiments on the CIFAR100 dataset using Convolutional Neural Network

(CNN), the parameter η_0 is set 0.09, which is similar to the cosine step decay method. The momentum parameter is consistently set to 0.9 across all methods and datasets mentioned. Additionally, weight-decay is set to 0.0001 for FashionMnist and CIFAR10, and 0.0005 for CIFAR100 in all mentioned methods. Furthermore, a batch size of 128 is used in all of the experiments mentioned. The parameter T_0 is the ratio of training samples to the batch size. The experiments are repeated with random seeds for 5 times to eliminate the influence of stochasticity.

4.2 Methods

We consider SGD with the following step sizes:

- $\eta_t = \text{constant}$
- $\eta_t = \frac{\eta_0}{1+\alpha\sqrt{t}}$
- $\eta_t = \frac{\eta_0}{1+\alpha t}$
- $\eta_t = \frac{\eta_0}{2} \left(1 + \cos \frac{\pi t}{T}\right)$
- $\eta_t = \eta_0 \left(1 - \frac{\ln t}{\ln T}\right)$

which are respectively named with SGD constant step size, $O\left(\frac{1}{\sqrt{t}}\right)$ step Size, $O\left(\frac{1}{t}\right)$ step Size, cosine, and the new logarithmic step size. Parameter t represents the iteration number of the inner loop and each outer iteration consists of iterations for training on mini-batches. Moreover, we compare the results of the new proposed logarithmic step size with Adam [23], SGD+Armijo [7], PyTorch’s ReduceLRonPlateau scheduler5 (abbreviated as ReduceLRonPlateau), and stagewise step size. Note that, similar to [18], the term “stagewise” refers to the Stagewise - 1 Milestone and Stagewise - 2 Milestone methods. (As a side note, since we use Nesterov momentum in all SGD variants, the stagewise step decay basically covers the performance of multistage accelerated algorithms (e.g., [24]). In all experiments, we follow the setting proposed in [18].

4.3 Results and discussion

We compare the methods in two groups as illustrated in Figs. 2 and 3 for more clarity of figures. In Fig. 2, the new proposed step size achieves the training loss close to zero as the best-known method such as SGD+Armijo after epoch number 40 in the FashionMnist dataset while its test accuracy outperforms all methods. In the CIFAR10 dataset, the new

Table 2 The hyperparameter values for the methods employed on the FashionMnist, CIFAR10, and CIFAR100 datasets

Methods	FashionMNIST			CIFAR10			CIFAR100		
	η_0	α	c	η_0	α	c	η_0	α	c
Constant step size	0.007	–	–	0.07	–	–	0.07	–	–
$O\left(\frac{1}{t}\right)$ step size	0.05	0.00038	–	0.1	0.00023	–	0.8	0.004	–
$O\left(\frac{1}{\sqrt{t}}\right)$ step size	0.05	0.00653	–	0.2	0.07907	–	0.1	0.015	–
Adam	0.0009	–	–	0.0009	–	–	0.0009	–	–
SGD+Armijo	0.5	–	0.1	2.5	–	0.1	5	–	0.5
ReduceLRonPlateau	0.04	0.5	–	0.07	0.1	–	0.1	0.5	–
Stagewise - 1 Milestone	0.04	0.1	–	0.1	0.1	–	0.07	0.1	–
Stagewise - 2 Milestones	0.04	0.1	–	0.2	0.1	–	0.07	0.1	–
Cosine step size	0.05	–	–	0.25	–	–	0.09	–	–
Ln step size	0.05	–	–	0.25	–	–	0.53	–	–

logarithmic step size is as good as the best previously studied method (i.e., $O(\frac{1}{t})$ step size). In the CIFAR10 dataset, the new proposed approach outperforms all the other methods both based on the training loss and the test accuracy. On the other hand, the new logarithmic step size achieves the best training loss, but the stagewise step size works well in terms of accuracy than the other step size for the CIFAR10 dataset as illustrated in Fig. 3.

As previously mentioned, we assessed the performance of the new proposed step size on the CIFAR100 dataset using two deep models for image classification. The results, depicted in Figs. 2 and 3, demonstrate that the new proposed step size outperforms all other methods in both groups when the deep model is a convolutional neural network (CNN). Overall, the use of the DenseNet-BC model results in better performance for all step decays, as shown in Figs. 4 and 5.

Notably, the new proposed step size performs better than the state-of-the-art method (i.e., cosine step decay).

Tables 3–5 demonstrate the average of the final training loss and test accuracy obtained by 5 runs starting from different random seeds on FashionMNIST, CIFAR10, and CIFAR100 datasets, respectively.

From Tables 3–5, we can conclude that:

- From Table 3, the SGD+Armijo step size obtained the best training loss, however, the proposed step size improved the test accuracy of the SGD by 0.03%.
- Table 4 shows that the new proposed step size has the best training loss and the cosine step size has the best test accuracy.
- From Table 5 we can conclude that the new proposed step size can improve the test accuracy of the SGD by 0.9%.

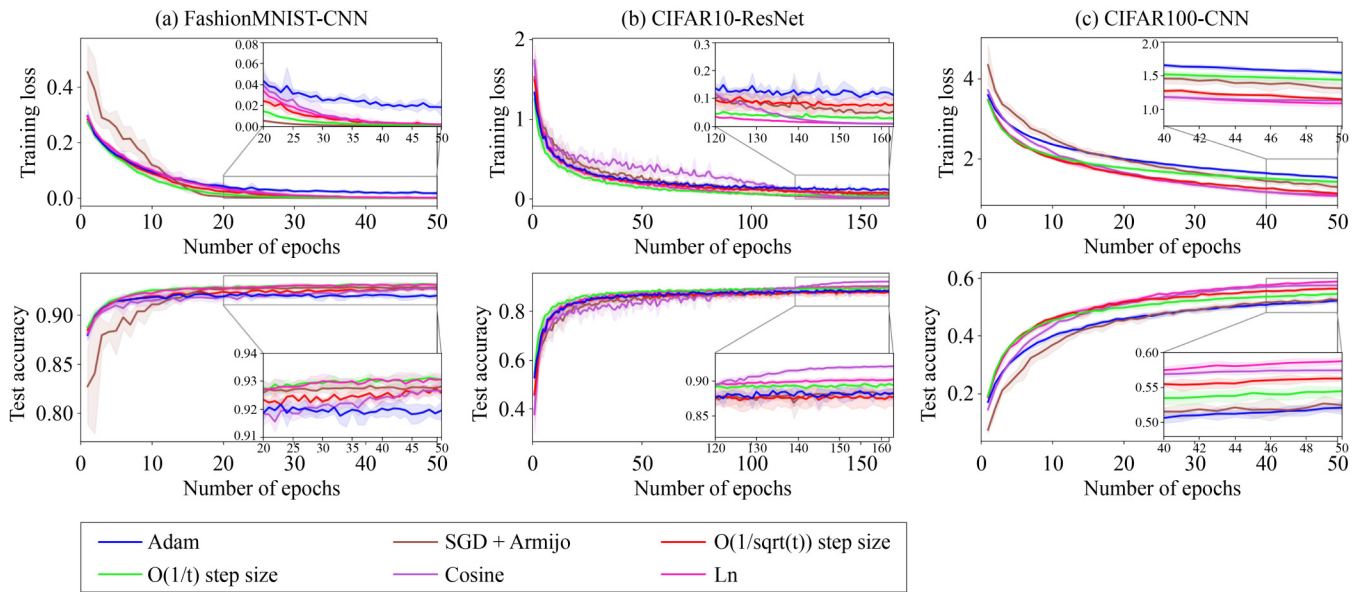


Fig. 2 Comparison of new proposed step size and five other step sizes on FashionMinst (a), CIFAR10 (b), and CIFAR100 (c) datasets

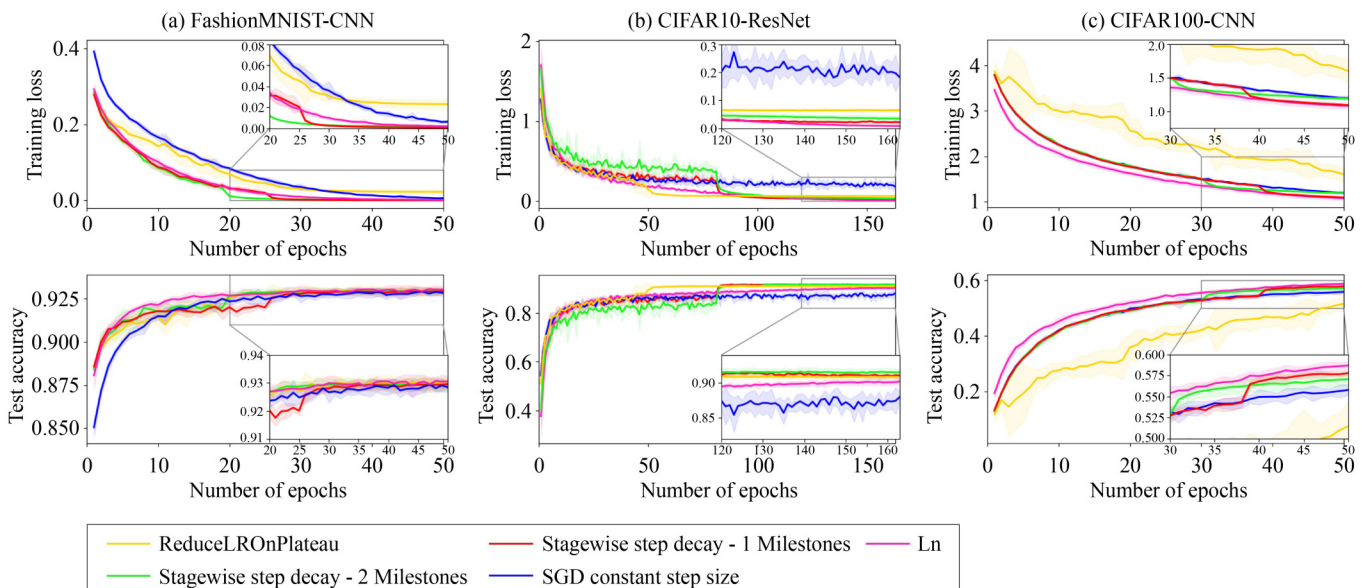


Fig. 3 Comparison of new proposed step size and four other step sizes on FashionMinst (a), CIFAR10 (b), and CIFAR100 (c) datasets

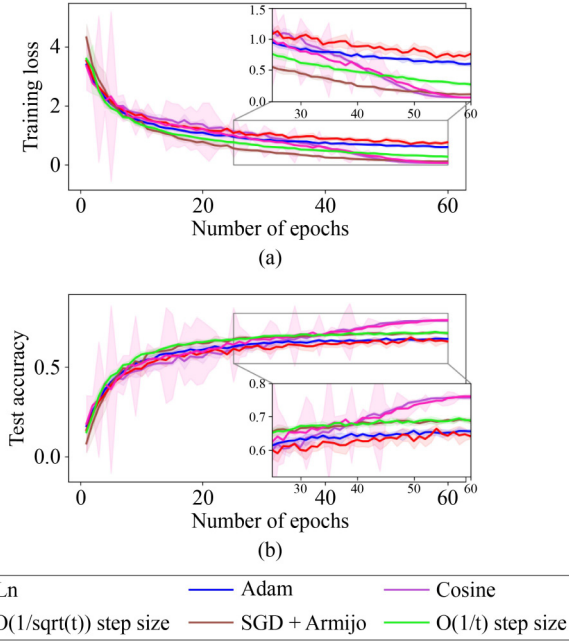


Fig. 4 Comparison of new proposed step size and five other step sizes on CIFAR100 dataset using the DenseNet-BC model

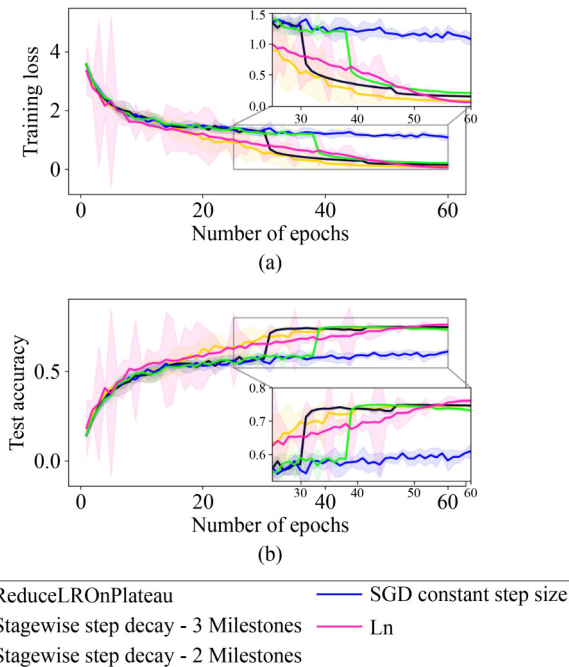


Fig. 5 Comparison of new proposed step size and four other step sizes on CIFAR100 dataset using the DenseNet-BC model

- Based on the information presented in Table 6, it can be inferred that the new proposed step size leads to a 0.2% improvement in training loss and a 0.11% improvement in test accuracy compared to the cosine step decay method.

5 Conclusion

We introduced a new logarithmic step size with warm restarts technique for the SGD. We showed that the new step size goes

Table 3 Average final training loss and test accuracy on the FashionMNIST dataset. The 95% confidence intervals of the mean loss and accuracy value over 5 runs starting from different random seeds

Methods	Training loss	Test accuracy
Constant Step Size	0.0007 ± 0.0003	0.9299 ± 0.0016
$O\left(\frac{1}{t}\right)$ Step Size	0.0006 ± 0.0000	0.9311 ± 0.0012
$O\left(\frac{1}{\sqrt{t}}\right)$ Step Size	0.0011 ± 0.0003	0.9275 ± 0.0007
Adam	0.0131 ± 0.0017	0.9166 ± 0.0019
SGD+Armijo	6.73e-05 ± 0.0000	0.9277 ± 0.0012
ReduceLRonPlateau	0.0229 ± 0.0032	0.9299 ± 0.0014
Stagewise - 1 Milestone	0.0004 ± 0.0000	0.9303 ± 0.0007
Stagewise - 2 Milestones	0.0015 ± 0.0000	0.9298 ± 0.0014
Cosine step size	0.0004 ± 1.1e-05	0.9284 ± 0.0005
Ln step size	0.0007 ± 0.0000	0.9314 ± 0.0018

Table 4 Average final training loss and test accuracy obtained by 5 runs starting from different random seeds and ± is followed by an estimated margin of error under 95% confidence on the CIFR10 dataset

Methods	Training loss	Test accuracy
Constant Step Size	0.1870 ± 0.0257	0.8776 ± 0.0060
$O\left(\frac{1}{t}\right)$ Step Size	0.0261 ± 0.0021	0.8946 ± 0.0020
$O\left(\frac{1}{\sqrt{t}}\right)$ Step Size	0.2634 ± 0.3493	0.8747 ± 0.0077
Adam	0.1140 ± 0.0060	0.8839 ± 0.0031
SGD+Armijo	0.0463 ± 0.0169	0.8834 ± 0.0059
ReduceLRonPlateau	0.0646 ± 0.0041	0.9081 ± 0.0019
Stagewise - 1 Milestone	0.0219 ± 0.0012	0.9111 ± 0.0034
Stagewise - 2 Milestones	0.0344 ± 0.0062	0.9151 ± 0.0039
Cosine step size	0.0102 ± 0.0007	0.9201 ± 0.0017
Ln step size	0.0089 ± 0.0011	0.9012 ± 0.0027

Table 5 Average final training loss and test accuracy obtained by 5 runs starting from different random seeds and ± is followed by an estimated margin of error under 95% confidence on the CIFR100 dataset using the convolutional neural network (CNN) model

Methods	Training loss	Test accuracy
Constant Step Size	1.19599 ± 0.0183	0.5579 ± 0.0039
$O\left(\frac{1}{t}\right)$ Step Size	1.43381 ± 0.0305	0.54436 ± 0.0038
$O\left(\frac{1}{\sqrt{t}}\right)$ Step Size	1.14579 ± 0.0125	0.5624 ± 0.0020
Adam	1.53433 ± 0.0180	0.5206 ± 0.0053
SGD+Armijo	1.3046 ± 0.0476	0.5244 ± 0.0053
ReduceLRonPlateau	1.6000 ± 0.055	0.5148 ± 0.1929
Stagewise - 1 Milestone	1.1889 ± 0.0077	0.5706 ± 0.0049
Stagewise - 2 Milestones	1.09061 ± 0.0040	0.5779 ± 0.0017
Cosine step size	1.12694 ± 0.0081	0.5739 ± 0.0025
Ln step size	1.0805 ± 0.0327	0.5869 ± 0.0035

to zero more slowly than most existing step sizes. We showed that the new logarithmic step size achieves the rate of $O\left(\frac{1}{\sqrt{t}}\right)$ for the SGD with a smooth non-convex objective function. Finally, to prove the effectiveness of the new proposed step size in practice, we did a wide range of implementations on three famous datasets, i.e., FashionMinst, CIFAR10, and CIFAR100 datasets, and compared the obtained results with 9 other step sizes. For the CIFAR100 datasets, the proposed step size improved the accuracy of the algorithm by 0.9% when we utilized a convolutional neural network model. For two other datasets, the new proposed step size obtained good results.

Table 6 Average final training loss and test accuracy obtained by 5 runs starting from different random seeds and \pm is followed by an estimated margin of error under 95% confidence on the CIFR100 dataset using the DenseNet-BC model

Methods	Training loss	Test accuracy
Constant Step Size	1.0789 \pm 0.06336	0.6089 \pm 0.01485
$O\left(\frac{1}{t}\right)$ Step Size	0.2709 \pm 0.02047	0.6885 \pm 0.0051
$O\left(\frac{1}{\sqrt{t}}\right)$ Step Size	0.7584 \pm 0.04068	0.6411 \pm 0.00469
Adam	0.6008 \pm 0.02396	0.6555 \pm 0.00430
SGD+Armijo	0.1162 \pm 0.01175	0.6869 \pm 0.00460
ReduceLROnPlateau	0.0863 \pm 0.00896	0.7440 \pm 0.00695
Stagewise - 1 Milestone	0.15300 \pm 0.00382	0.7456 \pm 0.00188
Stagewise - 2 Milestones	0.21032 \pm 0.01067	0.73102 \pm 0.00402
Cosine step size	0.0622 \pm 0.00349	0.7568 \pm 0.00311
Ln step size	0.06011 \pm 0.0018	0.7579 \pm 0.0022

Further investigation into the convergence rate of SGD based on the logarithmic step size is warranted for convex and strongly convex objective functions. Additionally, while this paper focuses on the convergence rate for smooth non-convex functions that do not satisfy the PL condition, it would be interesting to examine the convergence rate for smooth non-convex functions that satisfy the PL condition.

Acknowledgments The authors would like to thank the editors and anonymous reviewers for their constructive comments. The research of the first author is partially supported by the University of Kashan (1143902/2).

Competing interests The authors declare that they have no competing interests or financial conflicts to disclose.

Appendix A

Proof The proof of items 1 and 2 can be found in [25]. To prove the item (3), we start by defining the function $h(x)$ and demonstrating its positivity for all $x \geq 1$. Let us define

$$h(x) := \ln x - \frac{x-1}{x}.$$

Next, we calculate the derivative of $h(x)$:

$$h'(x) = \frac{1}{x} - \frac{1}{x^2} = \frac{x-1}{x^2}. \quad (\text{A1})$$

It is evident that $h'(x) \geq 0$ for all $x \geq 1$, indicating that $h(x)$ is an increasing function for $x \geq 1$. To show that $h(x)$ is positive, we evaluate $h(1)$, which yields $h(1) = 0$. Since $h'(x) > 0$ and $h(1) = 0$, we conclude that $h(x) \geq 0$ for all $x \geq 1$. Consequently, we have $\ln x \geq \frac{x-1}{x}$ for all $x \geq 1$. \square

Proof of Lemma 2 To prove, we use the definition of the new proposed step size and we have

$$\begin{aligned} \sum_{t=1}^T \eta_t &= \eta_0 \sum_{t=1}^T \left(1 - \frac{\ln t}{\ln T}\right) = \eta_0 \sum_{t=1}^T \left(\frac{\ln T - \ln t}{\ln T}\right) \\ &= \frac{\eta_0}{\ln T} \sum_{t=1}^T (\ln T - \ln t). \end{aligned}$$

Now, we use the second item of Lemma 1, that is, $\ln x - \ln y = \ln \frac{x}{y}$, and we have

$$\sum_{t=1}^T \eta_t = \frac{\eta_0}{\ln T} \sum_{t=1}^T \ln \frac{T}{t}.$$

Next, we are going to work on $\ln \frac{T}{t}$. In this regard, use the third item of Lemma 1, i.e., $\ln x \geq \frac{x-1}{x}$ for all $x \geq 1$. Since $\frac{T}{t} \geq 1$ therefore, we have $\ln \frac{T}{t} \geq \frac{\frac{T}{t}-1}{\frac{T}{t}} = \frac{T-t}{T}$. It implies that

$$\sum_{t=1}^T \eta_t \geq \frac{\eta_0}{T \ln T} \sum_{t=1}^T (T-t).$$

Now, we use the fact that $\sum_{t=1}^T (T-t) = \sum_{t=1}^T t$ and conclude that

$$\sum_{t=1}^T \eta_t \geq \frac{\eta_0}{T \ln T} \sum_{t=1}^T t = \frac{\eta_0(T+1)}{2 \ln T},$$

where the last equality is obtained from the fact that $\sum_{t=1}^T t = \frac{T(T+1)}{2}$. \square

Proof of Lemma 3 To establish the lemma, we employ the fact that for the function $f(x)$, we have

$$\sum_{i=1}^T f(x_i) \leq 1 + \int_1^T f(x) dx.$$

Consequently, for the newly proposed step size, we obtain

$$\eta_0^2 \sum_{t=1}^T \left(1 - \frac{\ln t}{\ln T}\right)^2 \leq \eta_0 \left[1 + \int_1^T \left(1 - \frac{\ln t}{\ln T}\right)^2 dt\right]. \quad (\text{A2})$$

To obtain an upper bound for (7) we will utilize the following integrals:

$$\begin{aligned} \int \ln x dx &= x \ln x - x + C, \\ \int \ln^2 x dx &= x \ln^2 x - 2x \ln x + 2x + C, \end{aligned} \quad (\text{A3})$$

where C represents a constant.

The right-hand side of inequality (A2) can be expressed as

$$= \eta_0^2 \left[1 + \int_1^T \frac{\ln^2 T - 2 \ln T \ln t + \ln^2 t}{\ln^2 T} dt\right]. \quad (\text{A4})$$

Now, utilizing Eq. (A3) and considering that $\ln 1 = 0$, we can represent Eq. (A4) as

$$\begin{aligned} &= \frac{\eta_0^2}{\ln^2 T} [\ln^2 T + (T-1) \ln^2 T] \\ &\quad - 2 \frac{\eta_0^2}{\ln^2 T} [\ln T (T \ln T - T + 1)] \\ &\quad + \frac{\eta_0^2}{\ln^2 T} [T \ln^2 T - 2T \ln T + 2T - 2] \\ &= \frac{\eta_0^2 [2T - 2 \ln T - 2]}{\ln^2 T} \\ &\leq \eta_0^2 \frac{2T}{\ln^2 T}. \end{aligned}$$

It implies that

$$\sum_{i=1}^T \eta_i^2 \leq \eta_0^2 \frac{2T}{\ln^2 T}.$$

□

Proof of Lemma 4 We know that $x_{t+1} = x_t - \eta_t \hat{g}_t$, where $\mathbb{E}[\hat{g}_t] = \nabla f(x_t)$. Using the smoothness of f , we have

$$\begin{aligned} f(x_{t+1}) &\leq f(x_t) + \langle \nabla f(x_t), x_{t+1} - x_t \rangle \\ &\quad + \frac{L}{2} \|x_{t+1} - x_t\|^2 \\ &\leq f(x_t) + \langle \nabla f(x_t), -\eta_t \hat{g}_t \rangle \\ &\quad + \frac{L}{2} \|x_{t+1} - x_t\|^2 \\ &\leq f(x_t) - \eta_t \langle \nabla f(x_t), \hat{g}_t \rangle + \frac{L\eta_t^2}{2} \|\hat{g}_t\|^2. \end{aligned} \quad (\text{A5})$$

The stochastic gradient oracle is variance-bounded by σ^2 , that is $\mathbb{E}[\|\hat{g}_t - \nabla f(x_t)\|^2] \leq \sigma^2$. Taking expectation on both sides of (A5) and applying $\mathbb{E}[\hat{g}_t] = \nabla f(x_t)$ and the variance-bounded assumption gives

$$\begin{aligned} &\mathbb{E}[f(x_{t+1})] \\ &\leq \mathbb{E}[f(x_t)] - \eta_t \mathbb{E}[\|\nabla f(x_t)\|^2] + \frac{L\eta_t^2}{2} \mathbb{E}[\|\hat{g}_t\|^2] \\ &\leq \mathbb{E}[f(x_t)] - \eta_t \mathbb{E}[\|\nabla f(x_t)\|^2] \\ &\quad + \frac{L\eta_t^2}{2} \mathbb{E}[\|\hat{g}_t - \nabla f(x_t) + \nabla f(x_t)\|^2]. \end{aligned}$$

Using the fact that gradient \hat{g}_t is unbiased, i.e., $\mathbb{E}[\hat{g}_t] = \nabla f(x_t)$. Therefore, we have

$$\begin{aligned} &\mathbb{E}[\|\hat{g}_t - \nabla f(x_t)\|^2] \\ &= \mathbb{E}[\|\hat{g}_t - \nabla f(x_t)\|^2] + \mathbb{E}[\|\nabla f(x_t)\|^2] \\ &\quad + 2\langle \hat{g}_t - \nabla f(x_t), \nabla f(x_t) \rangle \\ &= \mathbb{E}[\|\hat{g}_t - \nabla f(x_t)\|^2] + \mathbb{E}[\|\nabla f(x_t)\|^2]. \end{aligned}$$

Therefore, we can conclude that

$$\begin{aligned} \mathbb{E}[f(x_{t+1})] &\leq \mathbb{E}[f(x_t)] - \eta_t \mathbb{E}[\|\nabla f(x_t)\|^2] \\ &\quad + \frac{L\eta_t^2}{2} (\mathbb{E}[\|\hat{g}_t - \nabla f(x_t)\|^2] + \mathbb{E}[\|\nabla f(x_t)\|^2]) \\ &\leq \mathbb{E}[f(x_t)] + \left(-\eta_t + \frac{L\eta_t^2}{2}\right) \mathbb{E}[\|\nabla f(x_t)\|^2] \\ &\quad + \frac{L\eta_t^2}{2} \mathbb{E}[\|\hat{g}_t - \nabla f(x_t)\|^2] \\ &\leq \mathbb{E}[f(x_t)] + \left(-\eta_t + \frac{L\eta_t^2}{2}\right) \mathbb{E}[\|\nabla f(x_t)\|^2] \\ &\quad + \frac{L\eta_t^2 \sigma^2}{2}. \end{aligned}$$

Using this fact that $\eta_t \leq \frac{1}{cL} \leq \frac{1}{L}$, we have $-\eta_t + \frac{L\eta_t^2}{2} \leq -\frac{\eta_t}{2}$. It implies that

$$\begin{aligned} \frac{\eta_t}{2} \mathbb{E}[\|\nabla f(x_t)\|^2] &\leq \mathbb{E}[f(x_t)] - \mathbb{E}[f(x_{t+1})] \\ &\quad + \frac{L\eta_t^2 \sigma^2}{2}. \end{aligned}$$

□

Proof of Corollary 2

Corollary 1 is true for $i = 1, 2, \dots, l$. Therefore, we have

$$\begin{aligned} \min_i (\mathbb{E}[\|\nabla f(\bar{x}_T)\|^2]) &\leq \sum_{i=1}^l \mathbb{E}[\|\nabla f(\bar{x}_{T_i})\|^2] \\ &\leq \sum_{i=1}^l \left(\frac{4cL\ln T}{(T)} (f(x_{1_i}) - f^*) + \frac{4\sigma^2 T}{Lc(T+1)\ln T} \right) \\ &\leq l \max_i \left(\frac{4cL\ln T}{(T+1)} (f(x_{1_i}) - f^*) + \frac{4\sigma^2 T}{Lc(T+1)\ln T} \right) \\ &= \frac{4lcL\ln T}{T} (f(\bar{x}_1) - f^*) + \frac{4\sigma^2 l}{Lc\ln T}, \end{aligned}$$

in which $f(\bar{x}_1) = \max_i \{f(x_{1_i})\}$. □

References

- Robbins H, Monro S. A stochastic approximation method. *The Annals of Mathematical Statistics*, 1951, 22(3): 400–407
- Krizhevsky A, Sutskever I, Hinton G E. ImageNet classification with deep convolutional neural networks. *Communications of the ACM*, 2017, 60(6): 84–90
- Krizhevsky A, Hinton G. Learning multiple layers of features from tiny images. Toronto: University of Toronto, Department of Computer Science, 2009
- Redmon J, Farhadi A. Yolo9000: better, faster, stronger. In: *Proceedings of 2017 IEEE Conference on Computer Vision and Pattern Recognition*. 2017, 6517–6525
- Zhang J, Zong C. Deep neural networks in machine translation: an overview. *IEEE Intelligent Systems*, 2015, 30(5): 16–25
- Mishra P, Sarawadekar K. Polynomial learning rate policy with warm restart for deep neural network. In: *Proceedings of 2019 IEEE Region 10 Conference (TENCON)*. 2019, 2087–2092
- Vaswani S, Mishkin A, Laradji I, Schmidt M, Gidel G, Lacoste-Julien S. Painless stochastic gradient: Interpolation, line-search, and convergence rates. In: *Proceedings of the 33rd International Conference on Neural Information Processing Systems*. 2019, 335
- Gower R M, Loizou N, Qian X, Sailanbayev A, Shulgin E, Richtárik P. SGD: General analysis and improved rates. In: *Proceedings of the 36th International Conference on Machine Learning*. 2019, 5200–5209
- Huang G, Liu Z, Van Der Maaten L, Weinberger K Q. Densely connected convolutional networks. In: *Proceedings of 2017 IEEE Conference on Computer Vision and Pattern Recognition*. 2017, 2261–2269
- Smith L N. Cyclical learning rates for training neural networks. In: *Proceedings of 2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*. 2017, 464–472
- Loshchilov I, Hutter F. SGDR: Stochastic gradient descent with warm restarts. In: *Proceedings of the 5th International Conference on Learning Representations*. 2017
- Vrbanič G, Podgorelec V. Efficient ensemble for image-based identification of pneumonia utilizing deep CNN and SGD with warm restarts. *Expert Systems with Applications*, 2022, 187: 115834
- Xu G, Cao H, Dong Y, Yue C, Zou Y. Stochastic gradient descent with step cosine warm restarts for pathological lymph node image classification via PET/CT images. In: *Proceedings of the 5th IEEE International Conference on Signal and Image Processing (ICSIP)*. 2020, 490–493
- Nemirovski A, Juditsky A, Lan G, Shapiro A. Robust stochastic approximation approach to stochastic programming. *SIAM Journal on Optimization*, 2009, 19(4): 1574–1609
- Ghadimi S, Lan G H. Stochastic first-and zeroth-order methods for nonconvex stochastic programming. *SIAM Journal on Optimization*, 2013, 23(4): 2341–2368

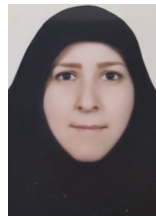
16. Bach F, Moulines E. Non-asymptotic analysis of stochastic approximation algorithms for machine learning. In: Proceedings of the 24th International Conference on Neural Information Processing Systems. 2011, 451–459
17. Rakhlin A, Shamir O, Sridharan K. Making gradient descent optimal for strongly convex stochastic optimization. In: Proceedings of the 29th International Conference on International Conference on Machine Learning, 2011
18. Li X, Zhuang Z, Orabona F. A second look at exponential and cosine step sizes: Simplicity, adaptivity, and performance. In: Proceedings of the 38th International Conference on Machine Learning. 2021, 6553–6564
19. Ge R, Kakade S M, Kidambi R, Netrapalli P. The step decay schedule: A near optimal, geometrically decaying learning rate procedure for least squares. In: Proceedings of the 33rd International Conference on Neural Information Processing Systems, 2019, 1341
20. Wang X, Magnússon S, Johansson M. On the convergence of step decay step-size for stochastic optimization. In: Proceedings of the 35th Conference on Neural Information Processing Systems, 2021, 14226–14238
21. Nocedal J, Wright S J. Numerical Optimization. New York: Springer, 1999
22. He K, Zhang X, Ren S, Sun J. Deep residual learning for image recognition. In: Proceedings of 2016 IEEE Conference on Computer Vision and Pattern Recognition. 2016, 770–778
23. Kingma D P, Ba J. Adam: a method for stochastic optimization. In: Proceedings of the 3rd International Conference on Learning Representations. 2015
24. Aybat N S, Fallah A, Gurbuzbalaban M, Ozdaglar A. A universally optimal multistage accelerated stochastic gradient method. In: Proceedings of the 33rd International Conference on Neural Information Processing Systems. 2019, 765
25. Thomas G B, Finney R L, Weir M D, Giordano F R. Thomas' Calculus, Early Transcendentals. 10th ed. Boston: Addison Wesley, 2002



Mahsa Soheil Shamaee received her BSc in Applied Mathematics from Alzahra University, Iran in 2009 and her MSc and PhD from Amirkabir University of Technology (Tehran Polytechnic), Iran all in the field of Computer Science in 2012 and 2018, respectively. Dr. Shamaee has been an Assistant Professor of Computer Science Department, University of Kashan, Iran since 2020. Her research interests include soft computing, reinforcement learning, wireless networks, multi-agent systems, and machine learning.



Sajad Fathi Hafshejani received a PhD degree in Mathematics from Shiraz University of Technology, Iran and currently serves as a PIMs postdoctoral fellow at the University of Lethbridge, Canada. His primary research interests encompass convex and non-convex optimization problems, machine learning, quantum computing, and interior point methods.



Zeinab Saeidian received her BSc and MSc in Applied Mathematics from University of Tehran, Iran in 2008 and 2010, respectively. Also, she received her PhD from K. N. Toosi University of Technology, Iran in the field of Applied Mathematics-Optimization in 2015. Dr. Saeidian has been an Assistant Professor of Applied Mathematics Department, University of Kashan, Iran since 2017. Her research interests include Nonlinear Optimization, Combinatorial optimization, Network Flows, and Machine Learning.