Journal of Computational and Applied Mathematics 451 (2024) 116083

Contents lists available at ScienceDirect

Journal of Computational and Applied Mathematics

journal homepage: www.elsevier.com/locate/cam

A stochastic gradient method with variance control and variable learning rate for Deep Learning

Giorgia Franchini [a], Federica Porta [a], Valeria Ruggiero [b], Ilaria Trombini [b,c,*], Luca Zanni [a,d]

[a] Dipartimento di Scienze Fisiche, Informatiche e Matematiche, Università di Modena e Reggio Emilia, Via Campi, 213/A, Modena, 41125, Italy
[b] Dipartimento di Matematica e Informatica, Università di Ferrara, Via Machiavelli, 30, Ferrara, 44121, Italy
[c] Dipartimento di Scienze Matematiche, Fisiche e Informatiche, Università di Parma, Parco Area delle Scienze, 7/A, Parma, 43124, Italy
[d] Institute of Informatics and Telematics, National Research Council, Pisa, 56124, Italy

ARTICLE INFO

Keywords:
Stochastic gradient method
Line search
Hyperparameters tuning
Deep learning

ABSTRACT

In this paper we study a stochastic gradient algorithm which rules the increase of the mini-batch size in a predefined fashion and automatically adjusts the learning rate by means of a monotone or non-monotone line search procedure. The mini-batch size is incremented at a suitable a priori rate throughout the iterative process in order that the variance of the stochastic gradients is progressively reduced. The a priori rate is not subject to restrictive assumptions, allowing for the possibility of a slow increase in the mini-batch size. On the other hand, the learning rate can vary non-monotonically throughout the iterations, as long as it is appropriately bounded. Convergence results for the proposed method are provided for both convex and non-convex objective functions. Moreover it can be proved that the algorithm enjoys a global linear rate of convergence on strongly convex functions. The low per-iteration cost, the limited memory requirements and the robustness against the hyperparameters setting make the suggested approach well-suited for implementation within the deep learning framework, also for GPGPU-equipped architectures. Numerical results on training deep neural networks for multi-class image classification show a promising behaviour of the proposed scheme with respect to similar state of the art competitors.

1. Introduction

Many machine learning and deep learning applications, including for example classification tasks and neural networks training, involve the solution of the following optimization problem

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

where n is typically very large and $f_i : \mathbb{R}^d \to \mathbb{R}$, $1 \le i \le n$, are continuously differentiable. The collection of $\{f_i\}$ is typically related to a set of n data, which in machine and deep learning applications is called training set. In recent literature, there are numerous efforts to propose efficient algorithms to address problem (1), see e.g. the reviews [1,2]. The optimal solution of (1) is commonly achieved by employing stochastic gradient methods. These schemes are specifically designed to reduce the computational cost associated with

* Corresponding author at: Dipartimento di Matematica e Informatica, Università di Ferrara, Via Machiavelli, 30, Ferrara, 44121, Italy.
E-mail addresses: giorgia.franchini@unimore.it (G. Franchini), federica.porta@unimore.it (F. Porta), valeria.ruggiero@unife.it (V. Ruggiero), ilaria.trombini@unife.it (I. Trombini), luca.zanni@unimore.it (L. Zanni).

https://doi.org/10.1016/j.cam.2024.116083
Received 14 December 2023; Received in revised form 5 June 2024
Available online 14 June 2024

the evaluation of the objective function and its derivatives. Indeed they exploit estimates of the true gradients by using subsets of the data. Among these approaches, the most widely employed scheme is the so called stochastic gradient descent (SGD) method, firstly proposed in [3]. Despite the prevalent use of SGD, it is well known that both the convergence and the performance of the algorithm are strongly dependent on the setting of appropriate values for its *hyperparameters*, that is all these parameters which have to be selected for running the iterative scheme, such as the learning rate and the mini-batch size; this task can be very challenging and computationally expensive. In this paper we present a SGD-like method to solve the optimization problem (1), where the mini-batch size progressively increases along the iterations and the learning rate can vary with a non monotonic behaviour.

Related works. Stochastic optimization methods that progressively change the mini-batch have been studied by several authors. Hereafter we recall the approaches which gained significant popularity in recent literature or served as partial inspiration for this work. In [4,5], the authors show that the SGD algorithm exhibits linear convergence when the mini-batch size grows geometrically. The main limitation of employing a geometric rate to update the mini-batch is the too rapid growth of its size. In [5–7], the so called *norm test* and its variants have been proposed as a practical strategy to increase the mini-batch size. The aim of the norm test is to promote search directions that are close to the true gradient. More recently [8], Bollapragada and coauthors proposed to replace the norm test by another procedure, called *augmented inner product test*, which ensures that the search direction is a descent direction with high probability. A different idea to increase the mini-batch size has been followed in [9,10]. The resulting method aims at dynamically reducing the variance of the stochastic gradients along the iterations. The dynamic sample methods developed in [6–10] ensure the convergence of SGD, under proper assumptions on both the objective function and the learning rate, and allow for a relatively slow growth of the mini-batch size. However, they require checking for a condition which may be computationally expensive and memory demanding, particularly when dealing with both deep neural networks and very large-scale datasets. A possibility to overcome this difficulty has been investigated in [11]. In this work the authors analyse a different implementation of the algorithm developed in [9,10], specifically designed to be more suitable for the deep learning framework. However the numerical experiments presented in [11] are only provided for either machine learning applications or training of neural networks with few layers.

On the other hand, the selection of the learning rate for the SGD method has gained significant attention and effort from many researchers in the last decade. Convergence results of SGD for properly bounded and fixed learning rate have been stated in several papers. Nevertheless, how to select the suitable value for the learning rate is a non trivial task and can require very demanding computational strategies. For this reason, convergence results under the assumptions that the learning rate can vary (even adaptively) throughout the iterations could be relevant. We first recall that, when employing a suitable learning rate diminishing as $\mathcal{O}(1/k)$ and a fixed mini-batch size in the SGD method, the expected value of the optimality gap, for strongly convex objective functions, or the expected sum of gradients for general objective functions, converges to 0 at a sublinear rate of $\mathcal{O}(1/k)$ [1].

We first recall that, when employing a suitable learning rate diminishing as $\mathcal{O}(1/k)$ and a fixed mini-batch size, the SGD method's expected value of the optimality gap, for strongly convex objective functions, or the expected sum of gradients for general objective functions, converges to 0 at a sublinear rate of $\mathcal{O}(1/k)$ [1]. Unfortunately, the selection of a diminishing sequence of learning rate can result in a very slow and expensive learning process. In [12,13] the authors prove the convergence of SGD in the case of learning rate fixed by means of a monotone line search procedure. However, a practical implementation of the scheme proposed in [12] involves the tuning of certain hyperparameters related to the mini-batch size used for computing the stochastic gradient and function estimates. An inadequate selection of this hyperparameters can potentially have adverse effects on numerical performance. The results presented in [13] are instead based on the assumption of a fixed mini-batch size, but they require the objective function to satisfy the so called *interpolation condition* or the so called *strong growth condition*. Both these hypotheses are strong and are not easily met by the objective functions commonly used in deep learning applications. We remark that in [8] the authors employ a line search procedure to fix the learning rate in the practical implementation of their algorithm, but, as mentioned above, the convergence proof is given under the assumption of a fixed learning rate. Finally, the algorithm analysed in [9] practically exploits a line search procedure as well. However, the theoretical requirement on the function to be minimized is guaranteed by the practical use of the line search only in the case of convex objective functions.

Contributions. In this paper we provide convergence results for the SGD algorithm under the assumptions that the variance of the stochastic gradients is reduced along the iterations and the learning rate, properly bounded, is fixed by means of a line search procedure which can be either monotone or non-monotone. The theoretical analysis is carried out for different types of objective functions, including non-convex, convex, and strongly convex ones. Neither the interpolation condition nor the strong growth condition are forced. The requirement on the variance of the stochastic gradients can be practically guaranteed by means of an increasing strategy for the mini-batch size. In more detail, by extending the analysis in [11], the mini-batch size increases according to an a priori rate which avoids the computational demanding procedures of the dynamic sampling strategies suggested in [6–10]. The a priori rate of increase for the mini-batch size does not need to satisfy restrictive assumptions. This, along with the possibility to study a proper rate of increase before the starting of the iterative process, makes the approach flexible of being adjusted to fit both the specific architecture to be used and the application to be considered. On the other hand, the line search procedure allows to automatically adapt the learning rate by guaranteeing that it is properly bounded from below. Furthermore, thanks to the line search, it becomes possible to approximate the theoretically imposed upper bound on the learning rate effectively, eliminating the need for manual fine-tuning of this hyperparameter. The considered method has been employed to optimize deep neural networks for multi-class image classification problems by showing promising results and robustness against the tuning of the hyperparameters.

Notations. Given a multi-value random variable s in a given probability space $(\Theta, \mathcal{F}, \mathcal{P})$, $\mathbb{E}_k[s]$ denotes the conditional expected value of s with respect to the σ-algebra generated by $x^{(0)}, \dots, x^{(k)}$, hereafter denoted by $\mathcal{F}_k$. On the other hand, $\mathbb{E}[s]$ denotes the total expected value of the random variable s.

Outline of the paper. The paper is organized in three sections. In Section 2, we consider a general SGD-like algorithm, equipped by a line search procedure, and we prove its theoretical convergence results. In Section 3, we provide the details of a practical implementation of the theoretical scheme, especially tailored for dealing with deep learning applications. Finally, in Section 4 we evaluate the robustness of the proposed method in training residual and dense networks for multi-class problems. A comparison with some state-of-the-art methods is reported for several well known datasets. Finally conclusions and future perspectives are given.

2. The general algorithm and its convergence analysis

In this section we consider a general SGD-like algorithm of the form

$$x^{(k+1)} = x^{(k)} - \alpha_k \nabla f_{\mathcal{N}_k}(x^{(k)}) \tag{2}$$

where $\mathcal{N}_k$ is a sub-sample of indices randomly and uniformly drawn from $\{1, \dots, n\}$ of size N_k, α_k is a positive learning rate and $\nabla f_{\mathcal{N}_k}(\cdot)$ is the average gradient of the loss functions with indices in $\mathcal{N}_k$, namely

$$f_{\mathcal{N}_k}(\cdot) = \frac{1}{N_k} \sum_{i \in \mathcal{N}_k} f_i(\cdot).$$

The learning rate α_k is selected by means of the line search procedure described in Algorithm 1. Differently from the SGD approach analysed in [13], the line search can be either monotone or non-monotone depending on the value of M.

Algorithm 1 Line search procedure for α_k

Given $\alpha_{max} > 0, M > 0, \beta, \gamma \in (0, 1)$

STEP 1. Set $\alpha = \alpha_{max}$

STEP 2. WHILE $f_{\mathcal{N}_k}(x^{(k)} - \alpha \nabla f_{\mathcal{N}_k}(x^{(k)})) > \max_{0 \le j \le \min\{k, M\}} f_{\mathcal{N}_{k-j}}(x^{(k-j)}) - \gamma \alpha \|\nabla f_{\mathcal{N}_k}(x^{(k)})\|^2$

$\alpha = \alpha\beta$

END

STEP 3. $\alpha_k = \alpha$

Hereafter we provide several convergence results for the considered method by detailing additional requirements needed on both the mini-batch size N_k and the learning rate α_k. Firstly we detail the main assumptions considered throughout the paper and usually imposed in the stochastic framework.

Assumptions

(A) $f(x)$ is lower bounded by f^* and $X^* = \operatorname{argmin}_x f(x) \neq \emptyset$.
(B) Any function $f_i(x)$ has a gradient L_i-Lipschitz continuous.
(C) The stochastic gradient at $x^{(k)}$ is an unbiased estimate of $\nabla f(x^{(k)})$, namely $\mathbb{E}_k[\nabla f_{\mathcal{N}_k}(x^{(k)})] = \nabla f(x^{(k)})$; furthermore, $\mathbb{E}_k[f_{\mathcal{N}_k}(x^{(k)})] = f(x^{(k)})$.

Consequently, f has a gradient L-Lipschitz continuous, with $L \le \frac{1}{n}\sum_{i=1}^n L_i$. We denote with $L_{max} = \max_i L_i$ and we observe $L \le L_{max}$. We emphasize that under the Assumption B, $\nabla f_{\mathcal{N}_k}(x)$ is Lipschitz continuous with parameter $L_{\mathcal{N}_k} \le \frac{1}{N_k}\sum_{i \in \mathcal{N}_k} L_i \le L_{max}$.

We firstly recall a classical result from stochastic analysis, namely the so called Robbins–Siegmund lemma. It will be employed for the convergence analysis.

Lemma 2.1 (*[14, Lemma 11, Sec 2.2.2]*). *Let v_k, u_k, η_k, β_k be nonnegative random variables and let*

$$\mathbb{E}(v_{k+1} \big| S_k) \le (1 + \eta_k) v_k - u_k + \beta_k \qquad a.s.$$

$$\sum_{k=0}^{\infty} \eta_k < \infty \quad a.s., \qquad \sum_{k=0}^{\infty} \beta_k < \infty \quad a.s.,$$

where $\mathbb{E}(v_{k+1} \big| S_k)$ denotes the conditional expectation for the given $v_0, \dots, v_k$, $u_0, \dots, u_k$, $\eta_0, \dots, \eta_k$, $\beta_0, \dots, \beta_k$. Then

$$v_k \longrightarrow v \quad a.s, \qquad \sum_{k=0}^{\infty} u_k < \infty \quad a.s,$$

where $v \ge 0$ is some random variable.

The next lemma we prove is crucial for the theoretical analysis of the considered method. Particularly it establishes the well-definiteness of the procedure in Algorithm 1 and provides a strictly positive lower bound for the learning rate α_k. Furthermore, it is worth to note that Lemma 2.2 is a consequence of the Descent Lemma [15, Lemma 6.9.1] which states that if $h : \mathbb{R}^d \to \mathbb{R}$ has a L-Lipschitz continuous gradient then it holds

$$h(y) \leq h(x) + \nabla h(x)^T (y - x) + \frac{L}{2}\|x - y\|^2, \quad \forall x, y \in \mathbb{R}^d.$$

Lemma 2.2. *Under the Assumption B, the line search procedure detailed in Algorithm 1 is well-posed and*

$$0 < \tilde{\alpha} \leq min\left\{\alpha_{max}, \frac{2\beta(1-\gamma)}{L_{\mathcal{N}_k}}\right\} \leq \alpha_k \leq \alpha_{max}, \tag{3}$$

where $\tilde{\alpha} = min\left\{\alpha_{max}, \frac{2\beta(1-\gamma)}{L_{max}}\right\}$.

Proof. If α_{max} satisfies the condition at Step 3 of Algorithm 1, then $\alpha_k = \alpha_{max}$. On the other hand, in view of iteration (2) and the Descent Lemma applied to $f_{\mathcal{N}_k}$, we have

$$\begin{aligned} f_{\mathcal{N}_k}(x^{(k)} - \alpha \nabla f_{\mathcal{N}_k}(x^{(k)})) &\leq f_{\mathcal{N}_k}(x^{(k)}) - \alpha\left(1 - \frac{L_{\mathcal{N}_k}\alpha}{2}\right)\|\nabla f_{\mathcal{N}_k}(x^{(k)})\|^2 \\ &\leq \max_{0\leq j\leq \min\{k,M\}} f_{\mathcal{N}_{k-j}}(x^{(k-j)}) - \alpha\left(1 - \frac{L_{\mathcal{N}_k}\alpha}{2}\right)\|\nabla f_{\mathcal{N}_k}(x^{(k)})\|^2. \end{aligned} \tag{4}$$

As a consequence, if

$$1 - \frac{L_{\mathcal{N}_k}\alpha}{2} \geq \gamma \quad \Longleftrightarrow \quad \alpha \leq \frac{2(1-\gamma)}{L_{\mathcal{N}_k}}$$

then the inequality at Step 2 of Algorithm 1 is automatically satisfied. We can deduce that the considered line search procedure is well defined.

We observe that if α_{max} does not satisfy the inequality at Step 2 of Algorithm 1, then there exists $\ell > 0$ such that $\alpha_{max}\beta^\ell \leq \frac{2(1-\gamma)}{L_{\mathcal{N}_k}}$ and hence $\alpha_k = \alpha_{max}\beta^\ell$. Moreover $\alpha_k/\beta = \alpha_{max}\beta^{\ell-1}$ does not satisfy the inequality defining the line search. As a consequence

$$\begin{aligned} \frac{\gamma\alpha_k}{\beta} &> \frac{\max_{0\leq j\leq \min\{k,M\}} f_{\mathcal{N}_{k-j}}(x^{(k-j)}) - f_{\mathcal{N}_k}(x^{(k)} - \alpha_k/\beta \nabla f_{\mathcal{N}_k}(x^{(k)}))}{\|\nabla f_{\mathcal{N}_k}(x^{(k)})\|^2} \\ &\geq \frac{\alpha_k}{\beta}\left(1 - \frac{L_{\mathcal{N}_k}\alpha_k}{2\beta}\right) \end{aligned}$$

where the second inequality follows from (4). Consequently,

$$\alpha_k > \frac{2\beta(1-\gamma)}{L_{\mathcal{N}_k}} \geq \min\left\{\alpha_{max}, \frac{2\beta(1-\gamma)}{L_{\mathcal{N}_k}}\right\}.$$

Since $L_{\mathcal{N}_k} \leq L_{max}$, by denoting with $\tilde{\alpha}$ the lower bound for α_k we can conclude that

$$0 < \tilde{\alpha} = \min\{\alpha_{max}, \frac{2\beta(1-\gamma)}{L_{max}}\} \leq \alpha_k \leq \alpha_{max}. \tag{5}$$

□

For $\gamma \leq 1/2$ and $\alpha_{max} > \frac{(2\beta(1-\gamma))}{L_{max}}$, α_k is at least as large as $\beta/L_{\mathcal{N}_k}$.

In the following theorem we provide a convergence result for the SGD scheme (2) combined with the line search in Algorithm 1 in the general non-convex case. In more detail the result is similar to that of [13, Theorem 3], but the assumption that the strong growth condition (SGC) [16] holds has been lightened. In more detail the objective function f satisfies the SGC with constant ρ if

$$\mathbb{E}_k[\|\nabla f_{\mathcal{N}_k}(w)\|^2] \leq \rho\|\nabla f(w)\|^2 \quad \forall w \in \mathbb{R}^d.$$

In this paper we consider a less restrictive requirement on the stochastic gradients. Indeed we assume that

$$\mathbb{E}_k[\|\nabla f_{\mathcal{N}_k}(x^{(k)}) - \nabla f(x^{(k)})\|^2] \leq \varepsilon_k \tag{6}$$

where $\{\varepsilon_k\}$ is a proper summable sequence. We remark that, under the assumption of unbiased stochastic gradient estimates, it holds that

$$\mathbb{E}_k[\|\nabla f_{\mathcal{N}_k}(x^{(k)}) - \nabla f(x^{(k)})\|^2] = \mathbb{E}_k[\|\nabla f_{\mathcal{N}_k}(x^{(k)})\|^2] - \|\nabla f(x^{(k)})\|^2. \tag{7}$$

The requirement (6) is analogous to the one employed in [9] and forces the variance of the stochastic gradients to be progressively reduced. In Section 3 we detail how to practically approximate condition (6). This approximation is different to the one suggested in [9], as it is specifically designed for the deep learning framework.

Theorem 2.1. *Let assume that A, B, C hold. Moreover, suppose that (6) holds where $\{\varepsilon_k\}$ is an exogenous positive sequence such that $\sum_{k=0}^{+\infty}\varepsilon_k < \infty$. The sequence obtained by the SGD-like scheme (2) equipped by the line search in Algorithm 1 with $\alpha_{max} < \frac{2}{L}$ achieves the rate:*

$$min_{k=0,\dots,T-1}\ \mathbb{E}[\|\nabla f(x^{(k)})\|^2] \le \frac{1}{\delta T}(f(x^{(0)}) - f^*) + \frac{\xi}{\delta T}\sum_{k=0}^{T-1}\varepsilon_k, \tag{8}$$

where $\delta = 2\tilde{\alpha} - L\alpha_{max}^2 > 0$. Moreover it holds that

(i) $\sum_{k=0}^{+\infty}\mathbb{E}[\|\nabla f(x^{(k)})\|^2] < +\infty$;
(ii) $\sum_{k=0}^{\infty}\|\nabla f(x^{(k)})\|^2 < +\infty$ *a.s. and* $\lim_{k\to+\infty}\|\nabla f(x^{(k)})\| = 0$ *a.s.*

Proof. We firstly prove that $\delta = 2\tilde{\alpha} - L\alpha_{max}^2$ is strictly positive.

Case 1: $\alpha_{max} \le \frac{2\beta(1-\gamma)}{L_{max}}$. Then, in view of (5), $\tilde{\alpha} = \alpha_{max}$ and, under the assumption $\alpha_{max} < \frac{2}{L}$, there holds

$$\delta = 2\alpha_{max} - L\alpha_{max}^2 > 0.$$

Case 2: $\alpha_{max} > \frac{2\beta(1-\gamma)}{L_{max}}$. Then $\tilde{\alpha} = \frac{2\beta(1-\gamma)}{L_{max}}$ and

$$\delta = \frac{4\beta(1-\gamma)}{L_{max}} - L\alpha_{max}^2.$$

To guarantee that $\delta > 0$, the parameter α_{max} must obey the following condition

$$\alpha_{max} \in \left(0, \sqrt{\frac{4\beta(1-\gamma)}{L_{max}L}}\right).$$

To avoid contradiction with the case assumption $\alpha_{max} > \frac{2\beta(1-\gamma)}{L_{max}}$, we require that

$$\sqrt{\frac{4\beta(1-\gamma)}{L_{max}L}} > \alpha_{max} > \frac{2\beta(1-\gamma)}{L_{max}}$$

and hence

$$\frac{2}{L}\sqrt{\frac{L\beta(1-\gamma)}{L_{max}}} > \alpha_{max} > \frac{2}{L}\frac{\beta L(1-\gamma)}{L_{max}}.$$

This holds when

$$\sqrt{\frac{L\beta(1-\gamma)}{L_{max}}} < 1$$

and

$$\alpha_{max} < \frac{2}{L}.$$

We observe that, since $L/L_{max} \le 1$ and $\beta \in (0,1)$, then $\sqrt{\frac{L\beta(1-\gamma)}{L_{max}}} < 1$ holds for $0 < \gamma < 1$.

Now we prove item (i). From the Assumption B on f, the definition of the iteration update (2) and Lemma 2.2, we obtain

$$\begin{aligned}
f(x^{(k+1)}) - f(x^{(k)}) &\le -\alpha_k \nabla f(x^{(k)})^T \nabla f_{\mathcal{N}_k}(x^{(k)}) + \frac{L\alpha_k^2}{2}\|\nabla f_{\mathcal{N}_k}(x^{(k)})\|^2 \\
&= \frac{\alpha_k}{2}(\|\nabla f(x^{(k)}) - \nabla f_{\mathcal{N}_k}(x^{(k)})\|^2 - \|\nabla f(x^{(k)})\|^2 - \|\nabla f_{\mathcal{N}_k}(x^{(k)})\|^2) + \\
&+ \frac{L\alpha_k^2}{2}\|\nabla f_{\mathcal{N}_k}(x^{(k)})\|^2 \\
&\le \frac{\alpha_{max}}{2}\|\nabla f(x^{(k)}) - \nabla f_{\mathcal{N}_k}(x^{(k)})\|^2 - \frac{\tilde{\alpha}}{2}(\|\nabla f(x^{(k)})\|^2 + \|\nabla f_{\mathcal{N}_k}(x^{(k)})\|^2) + \\
&+ \frac{L\alpha_{max}^2}{2}\|\nabla f_{\mathcal{N}_k}(x^{(k)})\|^2.
\end{aligned}$$

Taking the conditional expectation with respect to $\mathcal{F}_k$ and using (7), we obtain

$$\begin{aligned}
2\mathbb{E}_k[f(x^{(k+1)}) - f(x^{(k)})] &\le \alpha_{max}\mathbb{E}_k[\|\nabla f(x^{(k)}) - \nabla f_{\mathcal{N}_k}(x^{(k)})\|^2] + \\
&- \tilde{\alpha}\mathbb{E}_k[(\|\nabla f(x^{(k)})\|^2 + \|\nabla f_{\mathcal{N}_k}(x^{(k)})\|^2)] + \\
&+ L\alpha_{max}^2\mathbb{E}_k[\|\nabla f_{\mathcal{N}_k}(x^{(k)})\|^2] \\
&= (\alpha_{max} + L\alpha_{max}^2 - \tilde{\alpha})\mathbb{E}_k[\|\nabla f_{\mathcal{N}_k}(x^{(k)})\|^2] + \\
&- (\alpha_{max} + \tilde{\alpha})\|\nabla f(x^{(k)})\|^2.
\end{aligned}$$

Applying condition (6), we get

$$\begin{aligned}2\mathbb{E}_k[f(x^{(k+1)}) - f(x^{(k)})] &\leq (\alpha_{max} + L\alpha_{max}^2 - \tilde{\alpha})(\|\nabla f(x^{(k)})\|^2 + \varepsilon_k) + \\ &- (\alpha_{max} + \tilde{\alpha})\|\nabla f(x^{(k)})\|^2 \\ &= (L\alpha_{max}^2 - 2\tilde{\alpha})\|\nabla f(x^{(k)})\|^2 + (\alpha_{max} + L\alpha_{max}^2 - \tilde{\alpha})\varepsilon_k \\ &= -(2\tilde{\alpha} - L\alpha_{max}^2)\|\nabla f(x^{(k)})\|^2 + (\alpha_{max} + L\alpha_{max}^2 - \tilde{\alpha})\varepsilon_k.\end{aligned}$$

By denoting the positive constant $(\alpha_{max} + L\alpha_{max}^2 - \tilde{\alpha})$ with ξ and assuming that $\delta = 2\tilde{\alpha} - L\alpha_{max}^2 > 0$, we can conclude

$$\|\nabla f(x^{(k)})\|^2 \leq \frac{2}{\delta}\mathbb{E}_k[f(x^{(k)}) - f(x^{(k+1)})] + \frac{\xi}{\delta}\varepsilon_k. \tag{9}$$

Taking total expectation and summing from $k = 0$ to $K - 1$ we can write

$$\begin{aligned}\frac{1}{K}\sum_{k=0}^{K-1}\mathbb{E}[\|\nabla f(x^{(k)})\|^2] &\leq \frac{2}{\delta K}\sum_{k=0}^{K-1}\mathbb{E}[f(x^{(k)}) - f(x^{(k+1)})] + \frac{\xi}{\delta K}\sum_{k=0}^{K-1}\mathbb{E}[\varepsilon_k] \\ &\leq \frac{2}{\delta K}(f(x^{(0)}) - \mathbb{E}[f(x^{(K)})]) + \frac{\xi}{\delta K}\sum_{k=0}^{K-1}\varepsilon_k \\ &\leq \frac{2}{\delta K}(f(x^{(0)}) - f^*) + \frac{\xi}{\delta K}\sum_{k=0}^{K-1}\varepsilon_k.\end{aligned} \tag{10}$$

In view of (10), Eq. (8) directly follows.

Furthermore, item (i) follows from (10) by neglecting to both members the term $\frac{1}{K}$.

Now we prove item (ii). Given $f^* \leq f(x)$, $\forall x$ and taking the conditional expected value in (9), it holds that

$$\mathbb{E}_k[f(x^{(k+1)}) - f^*] \leq f(x^{(k)}) - f^* - \frac{\delta}{2}\|\nabla f(x^{(k)})\|^2 + \frac{\xi}{2}\varepsilon_k. \tag{11}$$

Since $\delta > 0$, Lemma 2.1 allows to conclude that $\sum_{k=0}^{\infty}\|\nabla f(x^{(k)})\|^2 < +\infty$, a.s. which implies that $\lim_{k\to+\infty}\|\nabla f(x^{(k)})\| = 0$ a.s. □

Now we give a convergence result in the case of convexity of the objective function.

Theorem 2.2. *Under the assumptions A, B, C, the convexity of f_i, $i = 1, \dots, n$, and $\alpha_{max} < \frac{2}{L}$, if condition (6) holds where $\{\varepsilon_k\}$ is an exogenous positive sequence such that $\sum_{k=0}^{+\infty}\sqrt{\varepsilon_k} < +\infty$, then the sequence $\{x^{(k)}\}$ generated by the SGD-like scheme (2) equipped by the line search in Algorithm 1 converges a.s. to a solution of the problem (1).*

Proof. Let x^* be a solution of the problem (1). The following inequalities are true.

$$\begin{aligned}\|x^{(k+1)} - x^*\|^2 &= \|x^{(k)} - x^*\|^2 + \alpha_k^2\|\nabla f_{\mathcal{N}_k}(x^{(k)})\|^2 + \\ &+ 2\alpha_k\nabla f_{\mathcal{N}_k}(x^{(k)})^T(x^* - x^{(k)}) \\ &= \|x^{(k)} - x^*\|^2 + \alpha_k^2\|\nabla f_{\mathcal{N}_k}(x^{(k)})\|^2 + \\ &+ 2\alpha_k\nabla f(x^{(k)})^T(x^* - x^{(k)}) + \\ &+ 2\alpha_k(\nabla f_{\mathcal{N}_k}(x^{(k)}) - \nabla f(x^{(k)}))^T(x^* - x^{(k)}) \\ &\leq \|x^{(k)} - x^*\|^2 + \alpha_k^2\|\nabla f_{\mathcal{N}_k}(x^{(k)})\|^2 + \\ &+ 2\alpha_k(f(x^*) - f(x^{(k)})) + 2\alpha_k\frac{\sqrt{\varepsilon_k}}{2}\|x^* - x^{(k)}\|^2 + \\ &+ 2\alpha_k\frac{1}{2\sqrt{\varepsilon_k}}\|\nabla f_{\mathcal{N}_k}(x^{(k)}) - \nabla f(x^{(k)})\|^2 \\ &\leq (1 + \alpha_{max}\sqrt{\varepsilon_k})\|x^{(k)} - x^*\|^2 + \alpha_{max}^2\|\nabla f_{\mathcal{N}_k}(x^{(k)})\|^2 + \\ &+ \alpha_{max}\frac{1}{\sqrt{\varepsilon_k}}\|\nabla f_{\mathcal{N}_k}(x^{(k)}) - \nabla f(x^{(k)})\|^2,\end{aligned} \tag{12}$$

where the first equality follows by the definition of $x^{(k+1)}$ in (2), the second one follows by adding and subtracting $2\alpha_k\nabla f(x^{(k)})^T(x^* - x^{(k)})$, the first inequality follows by the convexity of f and the property $a^Tb \leq \frac{\|a\|^2}{2\xi} + \frac{\xi\|b\|^2}{2}$, $\forall a, b \in \mathbb{R}^d$, $\xi \neq 0$, and the last inequality follows from the fact that $f(x^*) - f(x^{(k)}) \leq 0$ and $\alpha_k \leq \alpha_{max}$.

By taking the conditional expected value in (12), it follows that

$$\begin{aligned}\mathbb{E}_k[\|x^{(k+1)} - x^*\|^2] &\leq (1 + \alpha_{max}\sqrt{\varepsilon_k})\|x^{(k)} - x^*\|^2 + \alpha_{max}^2\|\nabla f(x^{(k)})\|^2 + \\ &+ \alpha_{max}^2\varepsilon_k + \alpha_{max}\sqrt{\varepsilon_k}.\end{aligned} \tag{13}$$

Since $\sum_{k=0}^{+\infty}\|\nabla f(x^{(k)})\|^2 < +\infty$ a.s. (*(ii)* in Theorem 2.1), we can invoke Lemma 2.1 with $\eta_k = \alpha_{max}\sqrt{\varepsilon_k}$, $u_k = 0$ and $\beta_k = \alpha_{max}^2\|\nabla f(x^{(k)})\|^2 + \alpha_{max}^2\varepsilon_k + \alpha_{max}\sqrt{\varepsilon_k}$. As a consequence, we can state that the sequence $\{\|x^{(k)} - x^*\|\}$ converges a.s. for any $x^* \in X^*$. Finally, with the same final argument in the proofs of Theorem 3 in [9] and Theorem 2.1 in [17], we can conclude that there exists

$\bar{x} \in X^*$ such that the sequence $\{x^{(k)}\}$ converges to $\bar{x}$ as $k \to \infty$ a.s. For the sake of completeness we report the details needed to conclude the proof.

Let $\{x_i^*\}_i$ be a countable subset of the relative interior $\mathrm{ri}(X^*)$ that is dense in X^*. From the almost sure convergence of $\|x^{(k)} - x^*\|$, $x^* \in X^*$, we have that for each i, the probability $\mathrm{Prob}(\{\|x^{(k)} - x_i^*\|\}$ is not convergent$) = 0$. Therefore, we observe that

$$\begin{aligned}&\mathrm{Prob}(\forall i\ \exists b_i \text{ s.t. } \lim_{k\to+\infty} \|x^{(k)} - x_i^*\| = b_i) = 1 - \mathrm{Prob}(\{\|x^{(k)} - x_i^*\|\} \text{ is not convergent})\\ &\geq 1 - \sum_i \mathrm{Prob}(\{\|x^{(k)} - x_i^*\|\} \text{ is not convergent}) = 1,\end{aligned}$$

where the inequality follows from the union bound, i.e. for each i, $\{\|x^{(k)} - x_i^*\|\}$ is a convergent sequence a.s. For a contradiction, suppose that there are convergent subsequences $\{u_{k_j}\}_{k_j}$ and $\{v_{k_j}\}_{k_j}$ of $\{x^{(k)}\}$ which converge to their limiting points u^* and v^* respectively, with $\|u^* - v^*\| = r > 0$. By Theorem 2.1, u^* and v^* are stationary; in particular, since f is convex, they are minimum points, i.e. $u^*, v^* \in X^*$. Since $\{x_i^*\}_i$ is dense in X^*, we may assume that for all $\epsilon > 0$, we have $x_{i_1}^*$ and $x_{i_2}^*$ are such that $\|x_{i_1}^* - u^*\| < \epsilon$ and $\|x_{i_2}^* - v^*\| < \epsilon$. Therefore, for all k_j sufficiently large,

$$\|u_{k_j} - x_{i_1}^*\| \leq \|u_{k_j} - u^*\| + \|u^* - x_{i_1}^*\| < \|u_{k_j} - u^*\| + \epsilon < 2\epsilon.$$

On the other hand, for sufficiently large k_j, we have

$$\|v_{k_j} - x_{i_1}^*\| \geq \|v^* - u^*\| - \|u^* - x_{i_1}^*\| - \|v_{k_j} - v^*\| > r - \epsilon - \|v_{k_j} - v^*\| > r - 2\epsilon.$$

This contradicts with the fact that $\|x^{(k)} - x_{i_1}^*\|$ is convergent a.s. Therefore, we must have $u^* = v^*$, hence there exists $\bar{x} \in X^*$ such that $\|x^{(k)} - \bar{x}\| \longrightarrow 0$ for $k \longrightarrow \infty$ a.s. □

Theorem 2.3 concerns the rate of convergence of the scheme.

Theorem 2.3. *Under the assumptions A, B, C, the convexity of f_i, $i = 1, \dots, n$, and $\alpha_{max} < \frac{2}{L}$, if condition (6) holds where $\{\varepsilon_k\}$ is an exogenous positive sequence such that $\sum_{k=0}^{+\infty} \sqrt{\varepsilon_k} < +\infty$, then the sequence $\{x^{(k)}\}$ generated by the SGD-like scheme (2) equipped by the line search procedure in Algorithm 1 is such that*

$$\mathbb{E}[f(\bar{x}^{(K)}) - f(x^*)] = \mathcal{O}\left(\frac{1}{K}\right)$$

where $\bar{x}^{(K)} = \frac{1}{K+1}\sum_{k=0}^{K} x^{(k)}$.

If moreover $\sum_{k=0}^{\infty} k\varepsilon_k < \infty$, it holds that

$$\mathbb{E}[f(x^{(k)}) - f(x^*)] = \mathcal{o}\left(\frac{1}{k}\right).$$

Proof. If we do not neglect $f(x^*) - f(x^{(k)})$ in (12) and in the subsequent inequality, we obtain

$$\begin{aligned}\mathbb{E}_k[\|x^{(k+1)} - x^*\|^2] \leq (1 + \alpha_{max}\sqrt{\varepsilon_k})\|x^{(k)} - x^*\|^2 + \alpha_{max}^2\|\nabla f(x^{(k)})\|^2 +\\ + \alpha_{max}^2\varepsilon_k + \alpha_{max}\sqrt{\varepsilon_k} + 2\tilde{\alpha}\mathbb{E}_k(f(x^*) - f(x^{(k)})).\end{aligned}$$

By summing this inequality from 0 to K and taking the total expectation, it holds that

$$\begin{aligned}\sum_{k=0}^{K}\mathbb{E}[f(x^{(k)}) - f(x^*)] &\leq \frac{1}{2\tilde{\alpha}}\left(\|x^{(0)} - x^*\|^2 - \mathbb{E}[\|x^{(K+1)} - x^*\|^2]\right) +\\ &+ \frac{\alpha_{max}^2}{2\tilde{\alpha}}\sum_{k=0}^{K}\mathbb{E}[\|\nabla f(x^{(k)})\|^2] + \frac{\alpha_{max}}{2\tilde{\alpha}}\sum_{k=0}^{K}\sqrt{\varepsilon_k}\mathbb{E}[\|x^{(k)} - x^*\|^2] +\\ &+ \frac{\alpha_{max}^2}{2\tilde{\alpha}}\sum_{k=0}^{K}\varepsilon_k + \frac{\alpha_{max}}{2\tilde{\alpha}}\sum_{k=0}^{K}\sqrt{\varepsilon_k}\\ &\leq \frac{1}{2\tilde{\alpha}}\|x^{(0)} - x^*\|^2 + \frac{\alpha_{max}^2}{2\tilde{\alpha}}\sum_{k=0}^{K}\mathbb{E}[\|\nabla f(x^{(k)})\|^2] +\\ &+ \frac{\alpha_{max}}{2\tilde{\alpha}}\sum_{k=0}^{K}\sqrt{\varepsilon_k}\mathbb{E}[\|x^{(k)} - x^*\|^2] + \frac{\alpha_{max}^2}{2\tilde{\alpha}}\sum_{k=0}^{K}\varepsilon_k +\\ &+ \frac{\alpha_{max}}{2\tilde{\alpha}}\sum_{k=0}^{K}\sqrt{\varepsilon_k}\\ &\leq \frac{1}{2\tilde{\alpha}}\|x^{(0)} - x^*\|^2 + \frac{\alpha_{max}^2}{2\tilde{\alpha}}S + \frac{\alpha_{max}}{2\tilde{\alpha}}\tilde{\varepsilon}M + \frac{\alpha_{max}^2}{2\tilde{\alpha}}\tilde{\varepsilon} +\\ &+ \frac{\alpha_{max}}{2\tilde{\alpha}}\tilde{\varepsilon}\end{aligned} \tag{14}$$

where the third inequality follows

- by setting $S = \sum_{k=0}^{+\infty} \mathbb{E}[\|\nabla f(x^{(k)})\|^2]$ (in view of item (i) of Theorem 2.1);
- from the fact that $\mathbb{E}[\|x^{(k)} - x^*\|^2]$ is a convergent sequence. Indeed by taking the total expectation in (13), Lemma 2 in Section 2.2.1 of [14] allows to state that $\mathbb{E}[\|x^{(k)} - x^*\|^2]$ converges and thus there exists M such that $\mathbb{E}[\|x^{(k)} - x^*\|^2] < M$;
- by setting $\bar{\varepsilon} = \sum_{k=0}^{+\infty} \varepsilon_k$ and $\tilde{\varepsilon} = \sum_{k=0}^{+\infty} \sqrt{\varepsilon_k}$.

Setting $\overline{x}^{(K)} = \frac{1}{K+1} \sum_{k=0}^{K} x^{(k)}$, from the Jensen's inequality, we observe that $\mathbb{E}(f(\overline{x}^{(K)})) \leq \frac{1}{K+1} \sum_{k=0}^{K} \mathbb{E}(f(x^{(k)}))$. As a consequence, by dividing (14) by $K+1$, we can write

$$\mathbb{E}\left(f(\overline{x}^{(K)}) - f(x^*)\right) \leq \frac{1}{K+1}\left(\frac{1}{2\tilde{\alpha}}\|x^{(0)} - x^*\|^2 + \frac{\alpha_{max}^2}{2\tilde{\alpha}} S\right) + \\ + \frac{1}{K+1}\left(\frac{\alpha_{max}}{2\tilde{\alpha}}\tilde{\varepsilon} M + \frac{\alpha_{max}^2}{2\tilde{\alpha}}\bar{\varepsilon} + \frac{\alpha_{max}}{2\tilde{\alpha}}\tilde{\varepsilon}\right).$$

Thus, we obtain the $\mathcal{O}(1/K)$ ergodic convergence rate of $\mathbb{E}\left(f(\overline{x}^{(K)}) - f(x^*)\right)$.

Since $0 \leq f(x^{(0)}) - f(x^*)$, inequality (14) implies that

$$\begin{aligned} \mathbb{E}\left[\sum_{k=1}^{K} f(x^{(k)})\right] - K f(x^*) &= \sum_{k=1}^{K} \mathbb{E}[f(x^{(k)}) - f(x^*)] \\ &\leq \sum_{k=0}^{K} \mathbb{E}[f(x^{(k)}) - f(x^*)] \\ &\leq \frac{1}{2\tilde{\alpha}}\|x^{(0)} - x^*\|^2 + \frac{\alpha_{max}^2}{2\tilde{\alpha}} S + \frac{\alpha_{max}}{2\tilde{\alpha}}\tilde{\varepsilon} M + \frac{\alpha_{max}^2}{2\tilde{\alpha}}\bar{\varepsilon} + \\ &\quad + \frac{\alpha_{max}}{2\tilde{\alpha}}\tilde{\varepsilon}. \end{aligned} \tag{15}$$

Now we determine a lower bound for $\mathbb{E}\left[\sum_{k=1}^{K} f(x^{(k)})\right]$. Inequality (9) allows to state that $\mathbb{E}\left[f(x^{(k)}) - f(x^{(k+1)})\right] + \frac{\xi}{2}\varepsilon_k \geq 0$. Thus, we get

$$\begin{aligned} 0 &\leq \sum_{k=1}^{K} k\mathbb{E}\left[f(x^{(k)}) - f(x^{(k+1)})\right] + \frac{\xi}{2}\sum_{k=1}^{K} k\varepsilon_k \\ &= \sum_{k=1}^{K} \mathbb{E}[f(x^{(k)})] - K\mathbb{E}[f(x^{(K+1)})] + \frac{\xi}{2}\sum_{k=1}^{K} k\varepsilon_k. \end{aligned}$$

Then, by denoting with $\Sigma = \frac{\xi}{2}\sum_{k=0}^{+\infty} k\varepsilon_k$, it follows that

$$K\mathbb{E}[f(x^{(K+1)})] \leq \sum_{k=1}^{K} \mathbb{E}[f(x^{(k)})] + \Sigma.$$

By combining the last inequality with (15), we can conclude that

$$\mathbb{E}[f(x^{(K+1)}) - f(x^*)] \leq \frac{1}{K}\left(\frac{1}{2\tilde{\alpha}}\|x^{(0)} - x^*\|^2 + \frac{\alpha_{max}^2}{2\tilde{\alpha}} S\right) + \\ + \frac{1}{K}\left(\frac{\alpha_{max}}{2\tilde{\alpha}}\tilde{\varepsilon} M + \frac{\alpha_{max}^2}{2\tilde{\alpha}}\bar{\varepsilon} + \frac{\alpha_{max}}{2\tilde{\alpha}}\tilde{\varepsilon} + \Sigma\right). \qquad \square$$

In the last part of this section we provide a convergence result under the hypothesis that the objective function satisfies the Polyak-Lojasiewicz property, that is

$$\|\nabla f(x)\|^2 \geq 2c(f(x) - f^*), \quad \forall x \in \mathbb{R}^d$$

where $c > 0$ and $f^* = \inf_{x \in \mathbb{R}^d} f(x)$. Before introducing the main theorem we recall the following technical lemma.

Lemma 2.3. *Let $\{\varepsilon_k\}$ be a positive sequence converging to zero R-linearly. Then for every $\zeta \in (0,1)$ and $q \in \mathbb{N}$*

$$r_k = \sum_{j=1}^{k+1} \zeta^{j-1}\varepsilon_{q+k-j}$$

converges to zero R-linearly.

Proof. We observe that

$$r_k = \sum_{j=1}^{k+1} \zeta^{j-1}\varepsilon_{q+k-j} = \sum_{j=1}^{k} \zeta^{j-1}\varepsilon_{q+k-j} + \zeta^k \varepsilon_{q-1}.$$

Lemma 4.2 in [18] implies that the sequence

$$s_k = \sum_{j=1}^{k} \zeta^{j-1} \varepsilon_{q+k-j}$$

converges to zero R-linearly. As a consequence, the thesis follows. □

Lemma 2.4. *Given a function $h : \mathbb{R}^d \to \mathbb{R}$ which satisfies the PL condition with constant $c > 0$ and whose gradient is L-Lipschitz continuous, then*

$$c \leq L.$$

Proof. The proof of this lemma relies on the results reported in Appendix A of [19]. Particularly, denoted by x_p the projection of $x \in \mathbb{R}^d$ onto the solution set of $\operatorname{argmin}_{x \in \mathbb{R}^d} h(x)$, then the PL condition implies the so called quadratic growth condition, namely

$$h(x) - h^* \geq \frac{c}{2} \|x_p - x\|^2, \forall x.$$

As a consequence, the following inequalities hold

$$\frac{c^2}{2} \|x - x_p\|^2 \leq c(h(x) - h(x_p)) \leq \frac{1}{2} \|\nabla h(x) - \nabla h(x_p)\|^2 \leq \frac{L^2}{2} \|x - x_p\|^2,$$

where the first inequality follows from the quadratic growth condition, the second inequality follows from the PL condition and the last inequality follows from the Lipschitz continuity of the gradient of h. Then the thesis easily follows. □

Theorem 2.4. *Under the assumptions A, B, C, the convexity of f_i, $i = 1, \dots, n$, and $\alpha_{max} < \frac{1}{L}$, if the objective function satisfies the PL condition and condition (6) holds where $\{\varepsilon_k\}$ is an exogenous positive sequence converging to zero R-linearly, then the sequence $\{x^{(k)}\}$ generated by the SGD-like scheme (2) equipped by the line search in Algorithm 1 is such that*

$$\mathbb{E}[f(x^{(k+1)}) - f^*] \leq (1 - \delta c)^k (f(x^{(0)}) - f^*) + \tau \rho^k \tag{16}$$

where $\delta = 2\tilde{\alpha} - L\alpha_{max}^2$, $\delta c < 1$, $\tau > 0$ and $\rho \in [0, 1)$.

Proof. At first we show that $\delta c < 1$. Indeed

$$\delta = 2\tilde{\alpha} - L\alpha_{max}^2 \leq 2\alpha_{max} - L\alpha_{max}^2 < \frac{1}{L},$$

where the second inequality follows by noting that $2L\alpha_{max} - L^2\alpha_{max}^2 - 1 = -(L\alpha_{max} - 1)^2 < 0$. As a consequence, $\delta c < 1$ since $c \leq L$ (Lemma 2.4). Now we prove inequality (16). In view of inequality (11), it holds that

$$\begin{aligned}
\mathbb{E}_k[f(x^{(k+1)}) - f^*] &\leq f(x^{(k)}) - f^* - \frac{\delta}{2} \|\nabla f(x^{(k)})\|^2 + \frac{\xi}{2} \varepsilon_k \\
&\leq f(x^{(k)}) - f^* - \delta c (f(x^{(k)}) - f^*) + \frac{\xi}{2} \varepsilon_k \\
&= (1 - \delta c)(f(x^{(k)}) - f^*) + \frac{\xi}{2} \varepsilon_k
\end{aligned}$$

where the second inequality follows from the PL condition for f. By taking the total expected value in the previous inequality, we get

$$\begin{aligned}
\mathbb{E}[f(x^{(k+1)}) - f^*] &\leq (1 - \delta c)\mathbb{E}[f(x^{(k)}) - f^*] + \frac{\xi}{2} \varepsilon_k \\
&\leq (1 - \delta c) \left[(1 - \delta c)\mathbb{E}[f(x^{(k-1)}) - f^*] + \frac{\xi}{2} \varepsilon_{k-1} \right] + \frac{\xi}{2} \varepsilon_k \\
&\dots \\
&\leq (1 - \delta c)^k (f(x^{(0)}) - f^*) + \frac{\xi}{2} \sum_{j=0}^{k} (1 - \delta c)^j \varepsilon_{k-j} \\
&= (1 - \delta c)^k (f(x^{(0)}) - f^*) + \frac{\xi}{2} \sum_{j=1}^{k+1} (1 - \delta c)^{j-1} \varepsilon_{1+k-j}.
\end{aligned}$$

Lemma 2.3 (with $\zeta = 1 - \delta c$ and $q = 1$) implies that there exist $\sigma > 0$ and $\rho \in [0, 1)$ such that

$$\sum_{j=1}^{k+1} (1 - \delta c)^{j-1} \varepsilon_{k+1-j} \leq \sigma \rho^k, \quad \forall k \in \mathbb{N}.$$

As a consequence, by denoting $\tau = \frac{\xi}{2}\sigma$, the thesis follows. □

3. A specific implementation: the Deep-LISA method

The algorithm presented in the previous section is a SGD-like scheme equipped by a line search whose convergence, in the case of general objective function, has been proved under the following hypotheses:

(i) $0 < \alpha_k \leq \alpha_{max} < \frac{2}{L}$;
(ii) $\mathbb{E}_k[\|\nabla f_{\mathcal{N}_k}(x^{(k)}) - \nabla f(x^{(k)})\|^2] \leq \varepsilon_k$, where $\{\varepsilon_k\}$ is an exogenous positive sequence such that $\sum_{k=0}^{+\infty} \varepsilon_k < +\infty$.

The aim of this section is to provide the details to practically realize this theoretical scheme. The resulting implementation is reported in Algorithm 2. Algorithm 2 can be viewed as an improved version of the LIne search based Stochastic gradient Algorithm (LISA) suggested in [9,10], especially tailored for dealing with deep learning applications. For this reason we denote Algorithm 2 as Deep-LISA method. Hereafter we provide a detailed description of the Deep-LISA steps.

Algorithm 2 Deep-LISA algorithm

Given $T > 0$, $n_{max} > 0$, $x^{(0)} \in \mathbb{R}^d$, $0 < N_0 < n$, $m \in \mathbb{N}$, $0 < \alpha_{min} < \alpha_0 < \alpha_{max}$, $\beta, \beta_2 \in (0,1)$, $\gamma > 0$, $M > 0$ and a positive sequence $\{\varepsilon_k\}_{k\in\mathbb{N}}$, $\sum_k \varepsilon_k < +\infty$, $C > 0$, $\hat{k} = 0$.

For $t = 1, 2, \ldots, T$

Step 1. Choose the size N_t according to

$$N_t = \min\left\{n_{max}, \max\left\{\left\lceil \frac{C}{\varepsilon_{\hat{k}+\lceil \frac{n}{N_{t-1}} \rceil}} \right\rceil, N_0\right\}\right\}$$

Step 2. Divide the training set into $\lceil \frac{n}{N_t} \rceil$ mini-batch of cardinality N_t.

Step 3. For $k = \hat{k}, \ldots, \hat{k} + \lceil \frac{n}{N_t} \rceil$-1

Step 3a. Choose a mini-batch $\mathcal{N}_k$ of size N_t.

Step 3b. Compute $f_{\mathcal{N}_k}(x^{(k)})$, $g_{\mathcal{N}_k}(x^{(k)})$.

Step 3c. Compute $\bar{x}^{(k)} = x^{(k)} - \alpha_k g_{\mathcal{N}_k}(x^{(k)})$.

If

$$f_{\mathcal{N}_k}(\bar{x}^{(k)}) \leq \max_{0\leq j\leq \min\{k,M\}} f_{\mathcal{N}_{k-j}}(x^{(k-j)}) - \gamma\alpha_k \|g_{\mathcal{N}_k}(x^{(k)})\|^2$$

Then go to Step 3d .

Else set $\alpha_k \leftarrow \beta\alpha_k$ and go to Step 3c.

Step 3d. Set $x^{(k+1)} = \bar{x}^{(k)}$ and $\alpha_{k+1} = \min\left(\alpha_{max}, \max\left(\frac{\alpha_k}{\beta_2}, \alpha_{min}\right)\right)$.

End For

Step 4. $\hat{k} = k + 1$

End For

Selection of the mini-batch size (Step 1 and Step 2)

Condition (ii) on the variance of the stochastic gradients has been already considered in [9,10] for the LISA method. However the practical implementation developed in these works involves a procedure which may be computationally expensive and memory demanding, especially for deep learning applications involving large-scale datasets. Indeed, in order to guarantee the requirement (ii), the mini-batch size N_k is progressively increased along the iterations so that

$$\frac{1}{N_k(N_k-1)} \sum_{i\in\mathcal{N}_k} \|\nabla f_i(x^{(k)}) - \nabla f_{\mathcal{N}_k}(x^{(k)})\|^2 \leq \varepsilon_k. \tag{17}$$

The control of the condition (17) at each iteration can be computationally expensive and memory intensive. Indeed the evaluation of the left hand side requires storing a matrix whose ith column contains the gradient $\nabla f_i(x^{(k)})$. For a neural network with a huge number of parameters, this $d \times N_k$ matrix can become difficult to be treated, especially by hardware accelerators, from both the computational and the memory resources point of view. By extending the analysis carried out in [11], we develop a new criterion to practically realize condition (ii) which avoids the drawbacks of the approach adopted for the LISA method and suits the architecture and the programming language typically employed in deep learning applications. The analysis is based on the following theoretical result. If the variance of each stochastic gradient $\nabla f_i(x^{(k)})$, $i \in \mathcal{N}_k$, is bounded by $C \geq 0$, then for arbitrary $i \in \mathcal{N}_k$ it holds that

$$\begin{aligned} \mathbb{E}_k[\|\nabla f_{\mathcal{N}_k}(x^{(k)}) - \nabla f(x^{(k)})\|^2] &= \frac{1}{N_k^2} \sum_{i \in \mathcal{N}_k} \mathbb{E}_k \left[\left\| \nabla f_i(x^{(k)}) - \nabla f(x^{(k)}) \right\|^2 \right] \\ &\leq \frac{C N_k}{N_k^2} = \frac{C}{N_k}, \end{aligned} \tag{18}$$

where the first equality follows from the properties of the variance of a linear combination of i.i.d. variables [20, pg. 183]. In view of (18), condition (ii) can be practically realized by forcing that

$$N_k \geq \frac{C}{\varepsilon_k}, \quad \forall k, \tag{19}$$

where ε_k is any positive sequence such that, at least, $\sum_k \varepsilon_k < +\infty$. As a consequence, the mini-batch size increases according to an a priori rate which is not subject to restrictive assumptions, also allowing for a slow increase. We underline that in this practical implementation the sequence ε_k is supposed to be non-increasing.

From Step 1 of Algorithm 2, it is possible to note that the criterion to update the mini-batch size is not limited to increase it by means of an a priori fixed exogenous sequence $\{\varepsilon_k\}$. Indeed Step 1 of Algorithm 2 takes into account the requirements of the main data loaders used to efficiently handle large training sets in the Python language. In more detail, the increase of the mini-batch size requires the data loader to partition the training set into clusters of samples matching this size, among which the mini-batch will be randomly selected. This process results in a very significant memory traffic and a consequent high computational time. For this reason we decide to properly keep fixed the mini-batch size throughout an entire epoch where by epoch we mean a set of consecutive accesses inspecting the whole training set. By avoiding frequently changing, almost every iteration, the mini-batch size, we aim at reducing the number of times the data loader needs to divide the training set in smaller clusters. Particularly, at the beginning of a new epoch, i.e., a new pass of the training set, we compute the value of the mini-batch size required at the end of that epoch in order to guarantee that inequality (19) holds. This overestimated value is then employed for all the iterations within such epoch. Hereafter we clarify the details of such approach by starting from the beginning of the iterative process. Given N_0 as the size of the initial mini-batch, n as the cardinality of the training set and n_{max} as the memory constraint of the architecture, the maximum number of iterations required to perform a complete inspection of the training set (first epoch) will be at most $\lceil \frac{n}{N_0} \rceil$. As a consequence, the mini-batch size for the entire first epoch can be set as

$$N_1 = \min \left\{ n_{max}, \max \left\{ \left\lceil \frac{C}{\varepsilon_{\lceil \frac{n}{N_0} \rceil}} \right\rceil, N_0 \right\} \right\};$$

consequently, it is assured that the last mini-batch size used in the first epoch satisfies (19). Actually, if $\left\lceil \frac{C}{\varepsilon_{\lceil \frac{n}{N_0} \rceil}} \right\rceil > N_0$, the total number of iterations K_1 of the first epoch will be less than the estimated number $\lceil \frac{n}{N_0} \rceil$. However requirement (19) for $k = K_1$ will still be satisfied since $\frac{C}{\varepsilon_{\lceil \frac{n}{N_0} \rceil}} > \frac{C}{\varepsilon_{K_1}}$. We emphasize that through a careful analysis of the mini-batch increase based on the $\{\varepsilon_k\}$ sequence and the features of the application under consideration, the size n_{max} should never be employed. We will deepen this aspect with an example later on. In general, denoted by $\hat{k}$ the current number of iterations performed, the size of the mini-batch for the tth epoch can be selected as

$$N_t = \min \left\{ n_{max}, \max \left\{ \left\lceil \frac{C}{\varepsilon_{\hat{k}+\lceil \frac{n}{N_{t-1}} \rceil}} \right\rceil, N_0 \right\} \right\}.$$

Once the value of N_t is defined, the training set is divided into groups of $\lceil \frac{n}{N_t} \rceil$ samples (Step 2) which will be used as mini-batches in the $\lceil \frac{n}{N_t} \rceil$ iterations of the successive epoch.

We conclude the arguments related to the selection of the mini-batch size by remarking the flexibility of our practical criterion. Indeed the possibility to a priori predict the growth of the mini-batch size allows to select the more suitable $\{\varepsilon_k\}$ sequence depending on the size of the dataset, the total number of epochs to perform and the memory capacity constraints of the architecture. Particularly, the selection of the $\{\varepsilon_k\}$ sequence can be determined by simulating various growth patterns based on the initial sample size. To illustrate this aspect with a practical example, we can consider a scenario where the training set consists of 50000 elements, and a total number of 50 epochs (50 complete inspections of the training set) are planned for the optimization phase. If we adopt

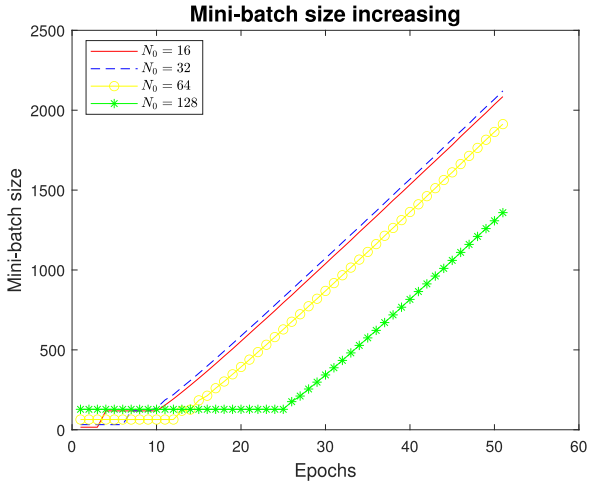

Fig. 1. Increase of the mini-batch size according to the rule $\varepsilon_k = 10^3 \cdot 0.999^k$ for different initial mini-batch sizes N_0.

$\varepsilon_k = 10^3 \cdot 0.999^k$ and $C = 10$, as proposed for the numerical experiments described in [10], the corresponding increase of the mini-batch size satisfying

$$N_t = \max\left\{\frac{C}{\varepsilon_{\hat{k}+\lceil \frac{n}{N_{t-1}} \rceil}}, N_0\right\},$$

is reported in Fig. 1. We stress that the values of the curves plotted in Fig. 1 can be computed without running the algorithm. Indeed the values of N_t and the progressive number of performed iterations $\hat{k}$ can be a priori computed once the sequence $\{\varepsilon_k\}$ and N_0 have been fixed. From Fig. 1, it is quite evident how the sequence $\varepsilon_k = 10^3 \cdot 0.999^k$ yields to a very different increase of the mini-batch size depending on N_0. Indeed for $N_0 = 128$, the mini-batch size does not change along the first 25 epochs, i.e., half of the planned epochs. This suggests that the chosen value for N_0 may be too large for the specific ε_k used. Particularly in the initial optimization phase, a smaller value for N_0 might be employed to exploit the potential of using stochastic gradients with a larger variance, as allowed by the theoretical results. On the other hand, if $N_0 = 16$ or $N_0 = 32$, the final sample size exceeds 2000 elements and this could represent a problem when dealing with GPGPUs (General Purpose Graphic Processor Unit) and their memory capacity constraints. For example, in our experiments we use an NVIDIA GeForce GTX 1650, capable of storing up to a maximum of approximately 1400 examples of a dataset of RGB images of size $32 \times 32 \times 3$. In view of these considerations, since the assessment of the effects in the mini-batch size increase of different $\{\varepsilon_k\}$ sequences is not computationally costly, we should analyse a different $\{\varepsilon_k\}$ sequence which allows the increase of the mini-batch size starting from the early epochs and ensures that the final mini-batch size remains below the GPGPU memory constraints. Particularly we consider the behaviour of the sequence $\varepsilon_k = 0.9995^k$ and $C = 10$. The corresponding increase of the mini-batch size per epoch for different values of N_0 is reported in Fig. 2. This second $\{\varepsilon_k\}$ sequence appears more convenient than the one previously considered since the memory capacity constraints of the employed GPGPU are not violated and a more effective increase of the mini-batch size is also achieved in the first epochs. We stress that, differently from the practical implementation suggested in [9,10] and based on inequality (17), this a priori approach to evaluate the behaviour of the mini-batch size in terms of $\{\varepsilon_k\}$ avoids the computational demanding process of fine-tuning the sequence through a trial-and-error procedure. Indeed by employing (17) to control the increase of the mini-batch size, its final value can be known only at the end of the iterative process, possibly violating the memory capacity constraints of the considered architecture.

Implementation of the iterations related to each epoch (Step 3)

Step 3 of Algorithm 2 is devoted to the stochastic gradient iterations. We remark that the variable $\hat{k}$ keeps track of the total number of iterations performed during the whole iterative process. Once the mini-batch of size N_t has been selected (Step 3a) among the ones realized in Step 2, $f_{\mathcal{N}_k}(x^{(k)})$ and $g_{\mathcal{N}_k}(x^{(k)})$ are computed (Step 3b) and $x^{(k+1)}$ is obtained by means of a line search based stochastic gradient iteration (Step 3c). Lemma 2.2 ensures that the line search is well posed. Moreover, in view of the Lipschitz continuity of $f_{\mathcal{N}_k}$ and the related Descent Lemma, the line search is exploited to practically approximate the bound for the learning rate recalled in (i). A similar approach has been also exploited in [8–10,16]. The attempt value of the learning rate to initialize the line search is set as

$$\alpha_k = \min\left(\alpha_{max}, \max\left(\frac{\alpha_{k-1}}{\beta_2}, \alpha_{min}\right)\right)$$

where $\beta_2 \in (0, 1)$. Hence, the possibility to increase the value of the learning rate with respect to the previous iteration is allowed. If the attempt value does not satisfy the line search condition, then it is reduced until this condition is satisfied.

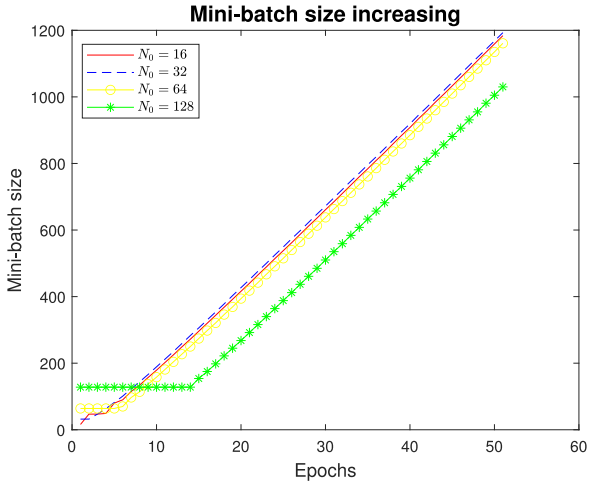

Fig. 2. Increase of the mini-batch size according to $\varepsilon_k = 0.9995^k$ for different initial mini-batch sizes N_0.

4. Numerical experiments

In this section, we describe the results of a numerical experimentation aimed to evaluate the effectiveness of the proposed method and its robustness with respect to the choice of the various hyperparameters involved. The numerical experiments focus on the use of residual neural networks for multi-class applications. The aim is to experimentally verify whether the implementation of the proposed method adapts to the challenges posed in the context of deep networks and large datasets, taking into account the typical complexity of these structures as discussed in Section 3. The numerical experiments have been carried out on a Laptop with AMD Ryzen 7 3750H, Radeon Vega Mobile Gfx; GPGPU NVIDIA GeForce GTX 1650. For all the experiments we used the Python language with the PyTorch framework.

We report a comparison between the Deep-LISA algorithm discussed in the previous Section 3 and other SGD-like methods already present in the literature. In particular, we consider the standard SGD method (with prefixed both mini-batch and learning rate) and the SGD version coupled with an Armijo-type line search, as proposed in [13, Algorithm 1] and hereafter denoted by SGD+Armijo. In more detail, the SGD+Armijo iteration is defined as in (2) where the size of the mini-batch $\mathcal{N}_k$ is constant along the iterations and a-priori fixed. The considered Armijo-type line search is monotone and, for $k > 0$, it is initialized by the following value

$$\alpha_k = \min\left(\alpha_{max}, \alpha_{k-1} \cdot \theta^{\frac{N_0}{n}}\right), \tag{20}$$

where $\theta > 1$ is a tunable hyperparameter. We refer the reader to [13] for further details. Finally, we remark that the LISA algorithm suggested in [10] appears to be unfeasible when the considered Neural Network is not very simple, and the training set is very large. The capacity constraints related to the hierarchical memories create a bottleneck in verifying condition (17). For this reason we exclude the LISA algorithms from the list of competitors.

A basic consideration for evaluating the results of the comparison concerns the concept of epoch, often used as a measure of computational complexity. This quantity hides the costs due to the usage of central memory and the memories of accelerators (as GPGPU). Consequently, in the following experiments, we decided to consider the execution time as performance measure, in order to take into account all factors related to the learning process (computational costs and hierarchical memory traffic).

In the numerical experiments, we consider two datasets and one Neural Network architecture. The two datasets are:

- **CIFAR-10** [21] which consists of 60000 RGB-images of size 32×32 belonging to 10 different classes; the training set and the test set contain 50000 and 10000 images, respectively;
- **CIFAR-100** [22] which consists of 60000 RGB-images of size 32×32 belonging to 100 different classes; the training set and the test set contain 50000 and 10000 images, respectively.

The considered Neural Network architectures is:

- **ResNet18** [23]: in the original version, the network has $d = 11169162$ trainable parameters.

The Resnet class of networks was developed to handle large image dataset, such as ImageNet where each element has $224 \times 224 \times 3$ pixels. To be able to handle the smaller images of CIFAR10 and 100, the structure of Resnet18 has been modified. In particular to avoid any side-effect due to batch normalization, we removed the related layers from the Neural Network. On the other hand, the net is subject to overfitting given the high number of parameters. Therefore, we included a regularization term in the objective function of the model through weight decay with parameter $5 \cdot 10^{-4}$ and data augmentation on the training set

Table 1
Averaged execution time per epoch for different mini-batch size in training ResNet18 on CIFAR10 by SGD.

Mini-batch size	8	16	32	64	128	256	512	1024
Time (s)	219	148	103	96	88	85	84	84

Table 2
Accuracy, total time and number of epochs achieved by SGD with different combinations of mini-batch size and learning rate in training ResNet18 on CIFAR10.

Mini-batch size	8	16	32	64	128	256	512	1024
	Learning rate 0.005							
Accuracy	0.7362	0.7240	0.6999	0.6094	0.5943	0.5004	0.4498	0.3884
Total time (s.)	4423	4294	4224	4228	4272	4207	4199	4205
Number of epochs	20	29	41	44	48	50	50	50
	Learning rate 0.01							
Accuracy	0.7514	0.7604	0.7345	0.6553	0.6416	0.5455	0.4885	0.4187
Total time (s.)	4401	4294	4224	4226	4225	4250	4200	4193
Number of epochs	20	29	41	43	48	50	50	50
	Learning rate 0.05							
Accuracy	0.7688	**0.7856**	0.7819	0.7145	0.7229	0.6123	0.5522	0.4647
Total time (s.)	4398	4235	4223	4225	4231	4214	4250	4206
Number of epochs	20	28	40	44	47	49	50	50
	Learning rate 0.1							
Accuracy	0.100	0.7713	0.7764	0.7205	0.7267	0.4577	0.5819	0.4557
Total time (s.)		4256	4223	4224	4264	4201	4196	4201
Number of epochs		28	41	44	49	50	50	50

Table 3
Averaged time per epochs, accuracy, total time and number of epochs achieved by SGD+Armijo with different mini-batch sizes in training ResNet18 on CIFAR10.

Mini-batch size	16	32	64	128
Time for epoch (s.)	202	133	120	114
Accuracy	0.6877	**0.7028**	0.5114	0.4167
Total time (s.)	4236	4255	4201	4217
Number of epochs	21	33	35	37

(Random Horizontal Flip, Random Crop and Normalize). A similar approach has been followed also in the available code related to [13]. Finally, we removed the first maxpool and reduced the first convolution (3×3 kernel size with stride 1).

ResNet18 on CIFAR10. In this paragraph, we analyse the performance of ResNet18 on the CIFAR10 dataset.

Following the suggestion in [24], which demonstrates that hyperparameters are strictly dependent on the neural network and computer architecture, Table 1 presents the variation in averaged execution time per epoch for several mini-batch sizes when using the standard SGD method, run for 50 epochs. The maximum mini-batch size, equal to 1024, is approximately the maximum number of examples handled by the employed GPGPU. The results reported in Table 1 highlight the impact of the memory traffic on the execution time per epoch by confirming the importance of comparing performance measures in terms of execution time. Moreover, in view of these considerations, for the following numerical experiments we established a time-budget of approximately 4200 seconds, corresponding to the minimum time required by SGD to perform 50 epochs.

Now we present the results achieved by SGD with different choices for the mini-batch size and the learning rate with the aim of finding the best hyperparameters configuration in terms of both accuracy and execution time. For all the configurations, the SGD method is stopped at the conclusion of that epoch when the time-budget of 4200 seconds was reached or if the maximum number of 50 epochs was performed. In Table 2, the accuracy on the test set, the computational time needed to achieve that accuracy and the total number of epochs performed are reported for each combination of mini-batch size and learning rate. The results presented in Table 2 confirm that a good performance of the SGD method is strictly dependent on the values of the hyperparameters. Even a slight adjustment to one of the hyperparameters can lead to a drastic shift in SGD's behaviour. Simply doubling the learning rate or halving the mini-batch size can induce the algorithm to diverge (see for example $\alpha_k = 0.1$ and $N_k = 8$). We remark that we also analysed the behaviour of SGD for the learning rate set at 0.5 and 1.0, by always observing divergence for these values. Finally, we emphasize that to obtain the configuration which provides the best final accuracy we spent a total computational time of about 272100 seconds, equivalent to more than three days.

A similar investigation has been carried out for the SGD+Armijo scheme [13]. In this version of SGD the mini-batch size is constant; this value is equal to 128 for all the numerical experiments in [13]. The following analysis is aimed to devise the optimal mini-batch size whereas the other hyperparameters have been fixed as in the original paper. In particular, in the inequality defining

Table 4
Accuracy, total time and number of epochs achieved by Deep-LISA with different settings in training ResNet18 on CIFAR10.

Deep-LISA setting	1	2	3	4	5	6
Accuracy	0.7678	0.7673	0.7568	0.7692	**0.7918**	0.7897
Total time (s.)	4231	4317	4201	4207	4301	4217
Number of epochs	35	37	34	35	31	31

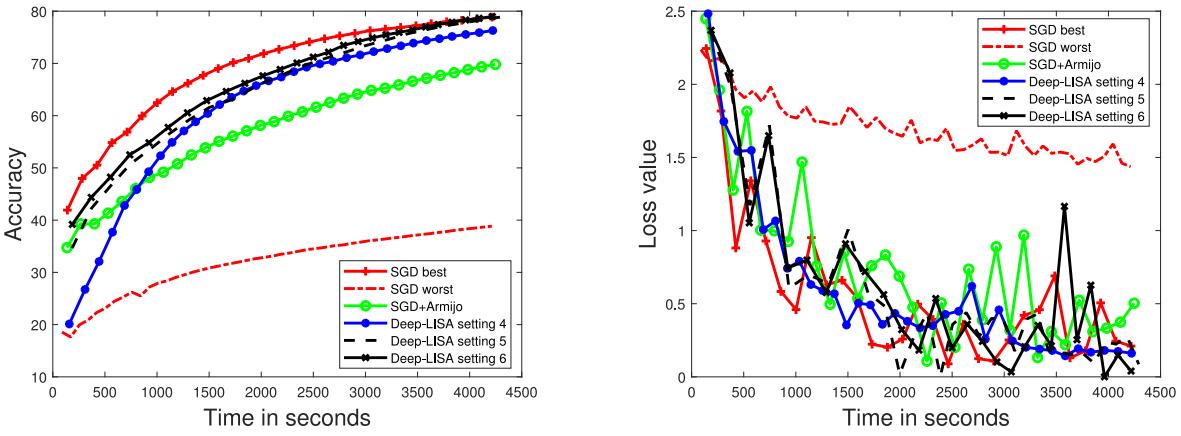

Fig. 3. Resnet18 on CIFAR10: accuracy on the test set (left panel) and loss values $f_{\mathcal{N}_k}(x^{(k)})$ (right panel) with respect to the execution time (in seconds) obtained by standard SGD (with the optimal and the worst settings), SGD+Armijo, Deep-LISA (with settings 4, 5 and 6).

the monotone line search

$$f_{\mathcal{N}_k}(x^{(k+1)}) \leq f_{\mathcal{N}_k}(x^{(k)}) - \gamma\alpha_k \|g_{\mathcal{N}_k}(x^{(k)})\|^2,$$

the parameter γ is set as 0.1, the value of β to reduce the learning rate is fixed as 0.9, the increase factor θ in (20) is 2, $\alpha_{max} = 10$ and $\alpha_0 = 1$. In Table 3, we report the obtained results. From Table 3 we observe a significant increase of the averaged time per epoch with respect to that of the standard SGD due to the presence of the line search. However, SGD+Armijo does not require the computational expensive phase of tuning the optimal value for the learning rate. For the next comparisons, we consider a mini-batch size of 32 for SGD+Armijo, corresponding to the obtained highest accuracy.

We discuss now the hyperparameters setting for Deep-LISA. In view of the conclusions of Section 3, the sequence $\{\varepsilon_k\}$ is set as 0.9995^k and $C = 10$, by taking into account the size of each example, the number of epochs, the GPGPU memory capacity and the initial value of the mini-batch size (selected as either 32 or 64). The hyperparameters related to the Deep-LISA line search have been fixed as $\beta = 0.5$, $\beta_2 = 3.0$, $\alpha_0 = \alpha_{max}$ and $\gamma = 0.1$. Both monotone and non-monotone line search procedures have been considered. The particular value of M is specified later on. In order to verify the robustness of Deep-LISA with respect to the values of N_0, α_{min} and α_{max}, we report the obtained numerical results for the following hyperparameters combinations coupled with $M = 1$:

1. $N_0 = 64$, $\alpha_{min} = 5 \cdot 10^{-3}$, $\alpha_{max} = 1.0$;
2. $N_0 = 32$, $\alpha_{min} = 5 \cdot 10^{-3}$, $\alpha_{max} = 1.0$;
3. $N_0 = 64$, $\alpha_{min} = 5 \cdot 10^{-3}$, $\alpha_{max} = 5.0$;
4. $N_0 = 64$, $\alpha_{min} = 5 \cdot 10^{-4}$, $\alpha_{max} = 1.0$.

Moreover we also consider a fifth and a sixth configuration where the Deep-LISA algorithm is coupled with the learning rate initialization (20) instead of the one reported at STEP 3D. For these last cases we have

5. $N_0 = 64$, $\alpha_{max} = 10$, $\theta = 2$, $M = 1$;
6. $N_0 = 64$, $\alpha_{max} = 10$, $\theta = 2$, $M = 5$.

In Table 4 we detail the accuracy on the test set, the total execution time and the number of epochs provided by Deep-LISA with the different considered settings and a time-budget of 4200 seconds. As for SGD+Armijo the averaged time per epoch is higher than the one of SGD, in view of the presence of the line search. The final accuracy achieved by Deep-LISA does not significantly change depending on the considered settings and it is comparable to the ones of standard SGD and SGD+Armijo both equipped with the optimal setting.

Fig. 3 shows the accuracy on the test set and the values of $f_{\mathcal{N}_k}$ obtained by means of the compared methods with respect to the execution time (in seconds). For the standard SGD method, the results for the optimal setting and the worst one, are reported. Moreover for Deep-LISA, the settings 4 and 5 are shown. As highlighted by the comparison among the Tables 2–4, the Deep-LISA

Table 5
Accuracy, top-5 accuracy, top-10 accuracy, total time and number of epochs achieved by SGD with different combinations of mini-batch size and learning rate in training ResNet18 on CIFAR100.

Mini-batch size	16	32	64
	Learning rate 0.01		
Accuracy	0.4376	0.4095	0.3181
Top-5 accuracy	0.7206	0.6977	0.6115
Top-10 accuracy	0.8198	0.8014	0.733
Total time (s)	4230	4200	4224
Number of epochs	30	42	44
	Learning rate 0.05		
Accuracy	0.4777	**0.4832**	0.3943
Top-5 accuracy	0.7640	**0.7648**	0.688
Top-10 accuracy	**0.8529**	0.8524	0.7928
Total time (s)	4207	4207	4269
Number of epochs	30	42	45
	Learning rate 0.1		
Accuracy	0.4503	0.4788	0.4011
Top-5 accuracy	0.7453	0.7623	0.6987
Top-10 accuracy	0.8386	0.8514	0.8025
Total time (s)	4320	4242	4275
Number of epochs	30	42	45

Table 6
Accuracy, top-5 accuracy, top-10 accuracy, total time and number of epochs achieved by SGD+Armijo and Deep-LISA in training ResNet18 on CIFAR100.

Methods	SGD+Armijo	Deep-LISA 5
Accuracy	0.4289	**0.4718**
Top-5 accuracy	0.7162	**0.7517**
Top-10 accuracy	0.8119	**0.8401**
Total time (s)	4266	4240
Number of epochs	32	37

algorithm provides comparable results to the ones of SGD and SGD+Armijo with the optimal hyperparameters setting. We stress that the performance of Deep-LISA is not so dependent on the tuning of the hyperparameters.

ResNet18 for CIFAR100. This paragraph is devoted to present the results obtained for the ResNet18 on the CIFAR100 dataset. Similarly to the analysis performed for the CIFAR10 dataset, we firstly investigate the best hyperparameters setting for the SGD algorithm. In Table 5 we report the values of the accuracy, the top-5 accuracy and the top-10 accuracy on the test set provided by SGD for different configurations of mini-batch size and learning rate. We recall that the top-k accuracy score is a metric commonly used in multi-class problems when the number of classes is very large. In particular this metric computes the number of times where the correct label is among the top k labels predicted (ranked by predicted scores). Moreover, in Table 5, the total computational time and the number of epochs performed are also presented. We remark that a time budget of approximately 4200 seconds was also taken into account for these numerical experiments.

The results of Table 5 confirm that the performance of SGD can largely vary depending of the hyperparameters configurations: the best results can be achieved by setting, $\forall k$, $N_k = 32$ and $\alpha_k = 0.05$. In Table 6 we show the results for the SGD+Armijo algorithm and the Deep-LISA method both equipped by the hyperparameters setting found before as the optimal for the CIFAR10 dataset. From these results we can conclude that Deep-LISA outperforms SGD+Armijo in terms of accuracies on the test set. Analogous considerations can be deduced by Fig. 4 where the values of the accuracy, the top-5 accuracy, the top-10 accuracy and the loss values $f_{\mathcal{N}_k}(x^{(k)})$ provided by the considered methods are reported with respect to the computational time. The Deep-LISA scheme show promising performance with respect to SGD and SGD+Armijo without needing an expensive phase of hyperparameters tuning. Indeed we stress that, for the second dataset, the hyperparameters setting of Deep-LISA is the same adopted for the first one.

5. Conclusions

In this paper we analyse a stochastic gradient method designed for minimizing a finite-sum optimization problem. The considered theoretical algorithm is an SGD-like scheme where the learning rate is adjusted by means of a line search strategy and the mini-batch size properly increases along the iterations according to an a priori rate. The practical implementation of the theoretical method has been studied in order to make it suitable for deep-learning applications. Indeed the growth of the mini-batch size can be monitored before running the iterations by ensuring adherence to the memory constraints imposed by the hardware accelerators. Moreover the mini-batch size is increased to prevent inefficient use of the main data loaders employed to efficiently handle large training

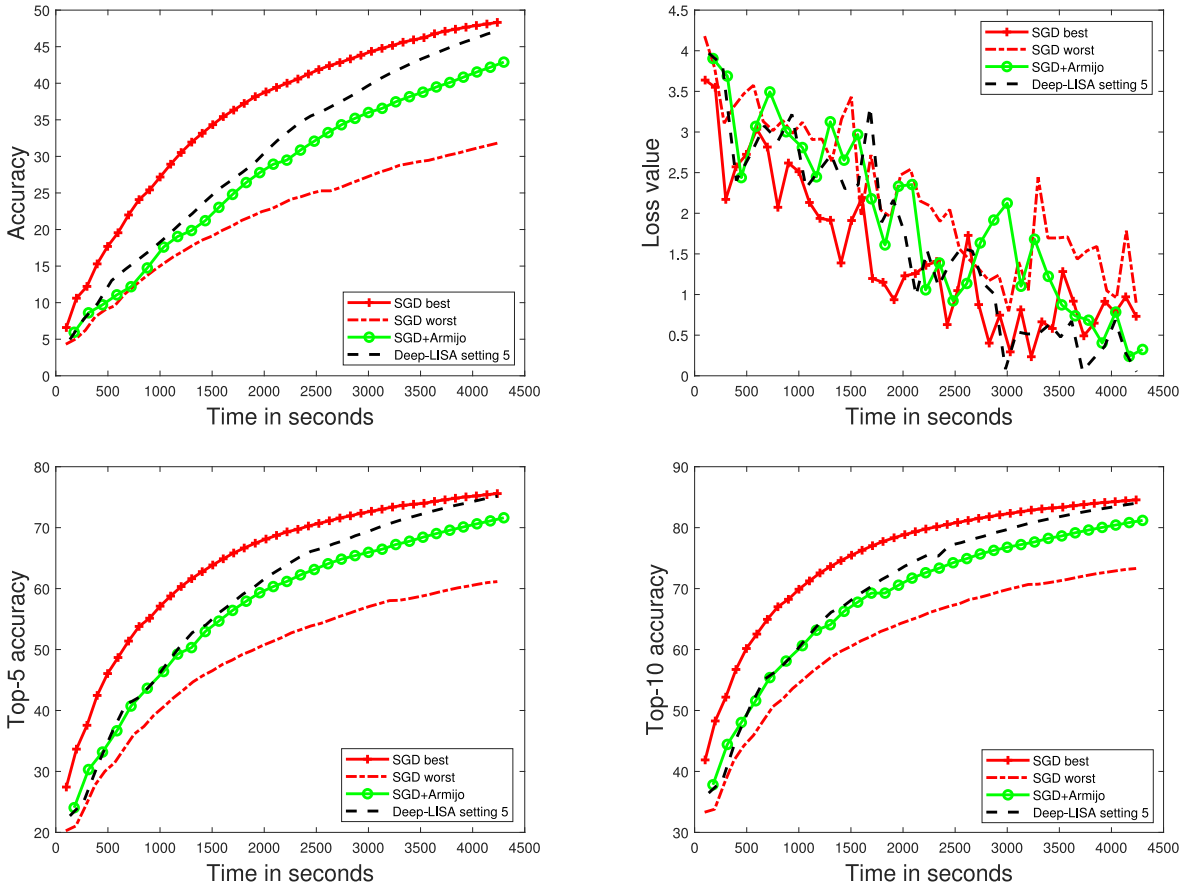

Fig. 4. Resnet18 on CIFAR100: accuracy on the test set (first row, left panel), loss values $f_{\mathcal{N}_k}(x^{(k)})$ (first row, right panel), top-5 accuracy on the test set (second row, left panel) and top-10 accuracy on the test set (second row, right panel) with respect to the execution time (in seconds) obtained by standard SGD (with the optimal and the worst settings), SGD+Armijo, Deep-LISA (with settings 5).

sets in the Pyhton language. Numerical results on multi-class image classification problems show a promising performance of the considered approach when compared to similar state of the art methods.

Data availability

We provided the link of the employed data.

Acknowledgements

This work was supported by "Gruppo Nazionale per il Calcolo Scientifico (GNCS–INdAM)" (Progetti 2023). G. Franchini was partially funded by the European Union–FSE–REACT–E, PON Research and Innovation 2014–2020 DM1062/2021. L. Zanni work was partly funded by the Partenariato Esteso PE00000013 – "FAIR", funded by the European Commission under the NextGeneration EU programme, PNRR – M4C2 – Investimento 1.3. F. Porta and L. Zanni were partially supported by the Italian MUR through the PRIN 2022 Project "Numerical Optimization with Adaptive Accuracy and Applications to Machine Learning", project code: 2022N3ZNAX (CUP E53D23007700006), under the National Recovery and Resilience Plan (PNRR), Italy, Mission 04 Component 2 Investment 1.1 funded by the European Commission – NextGeneration EU programme. F. Porta and L. Zanni were partially supported by the Italian MUR through the PRIN 2022 PNRR Project "Advanced optimization METhods for automated central veIn Sign detection in multiple sclerosis from magneTic resonAnce imaging (AMETISTA)", project code: P2022J9SNP (CUP E53D23017980001), under the National Recovery and Resilience Plan (PNRR), Italy, Mission 04 Component 2 Investment 1.1 funded by the European Commission – NextGeneration EU programme.
V. Ruggiero and I. Trombini were partially funded by European Union – NextGenerationEU through the Italian Ministry of University and Research as part of the PNRR – Mission 4 Component 2, Investment 1.3 (MUR Directorial Decree no. 341 of 03/15/2022, FAIR "Future" Partnership Artificial Intelligence Research, Proposal Code PE00000013 – CUP J33C22002830006).

References

[1] L. Bottou, F. Curtis, J. Nocedal, Optimization methods for large-scale machine learning, SIAM Rev. 60 (2018) 223–311.
[2] F. Curtis, K. Scheinberg, Optimization methods for supervised machine learning: from linear models to deep learning, in: Leading Developments from INFORMS Communities. INFORMS, 2017, pp. 89–114.
[3] H. Robbins, S. Monro, A stochastic approximation method, Ann. Math. Stat. (1951) 400–407.
[4] M. Friedlander, M. Schmidt, Hybrid deterministic-stochastic methods for data fitting, SIAM J. Sci. Comput. 34 (2012) A1380–A1405.
[5] H. Richard, R. Byrd, G. Chin, J. Nocedal, Y. Wu, Sample size selection in optimization methods for machine learning, Math. Program. 134 (2012) 127–155.
[6] C. Cartis, K. Katya Scheinberg, Global convergence rate analysis of unconstrained optimization methods based on probabilistic models, Math. Program. (2015) 1–39.
[7] F. Hashemi, S. Ghosh, R. Pasupathy, On adaptive sampling rules for stochastic recursions, in: Simulation Conference (WSC), 2014 Winter, 2014, pp. 3959–3970.
[8] R. Bollapragada, R. Byrd, J. Nocedal, Adaptive sampling strategies for stochastic optimization, SIAM J. Optim. 28 (2018) 3312–3343.
[9] G. Franchini, F. Porta, V. Ruggiero, I. Trombini, A line search based proximal stochastic gradient algorithm with dynamical variance reduction, J. Sci. Comput. 94 (2023) 23.
[10] G. Franchini, F. Porta, V. Ruggiero, I. Trombini, L. Zanni, Learning rate selection in stochastic gradient methods based on line search strategies, Appl. Math. Sci. Eng. 31 (2023) 2164000.
[11] G. Franchini, F. Porta, V. Ruggiero, I. Trombini, L. Zanni, Line search stochastic gradient algorithm with a-priori rule for monitoring the control of the variance, in: LNCS Conference Proceedings, NUMTA2023, 2023.
[12] C. Paquette, K. Scheinberg, A stochastic line search method with expected complexity analysis, SIAM J. Optim. 30 (2020) 349–376.
[13] S. Vaswani, A. Mishkin, I. Laradji, M. Schmidt, G. Gidel, S. Lacoste-Julien, Painless stochastic gradient; interpolation, line–search, and convergence rates, Adv. Neural Inf. Process. Syst. 32 (2018) 3312–3343.
[14] B. Polyak, Introduction to Optimization, Optimization Software, New York, 1987.
[15] D. Bertsekas, Convex Optimization Theory, Chapter 6 on Convex Optimization Algorithms, Athena Scientific, Belmont, Massachussets, 2009.
[16] M. Schmidt, N. Le Roux, Fast convergence of stochastic gradient descent under a strong growth condition, 2013, arXiv:1308.6370.
[17] C. Poon, J. Liang, C. Schoenlieb, Local convergence properties of SAGA/Prox-SVRG and acceleration, in: Proceedings of the 35th International Conference on Machine Learning, Vol. 80, 2018, pp. 4124–4132.
[18] N. Krejić, N. Krklec Jerinkić, Nonmonotone line search methods with variable sample size, Numer. Algorithms 68 (2015) 711–739.
[19] H. Karimi, J. Nutini, M. Schmidt, Linear convergence of gradient and proximal-gradient methods under the Polyak-Łojasiewicz condition, in: Machine Learning and Knowledge Discovery in Databases, 2016, pp. 795–811.
[20] J. Freund, Mathematical Statistics, Prentice-Hall Englewood Cliffs, N.J., 1971.
[21] A. Krizhevsky, V. Nair, G. Hinton, CIFAR-10 (Canadian Institute for Advanced Research). http://www.cs.toronto.edu/kriz/cifar.html.
[22] A. Krizhevsky, V. Nair, G. Hinton, CIFAR-100 (Canadian Institute for Advanced Research). http://www.cs.toronto.edu/kriz/cifar.html.
[23] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 770–778.
[24] N. Golmant, N. Vemuri, Z. Yao, V. Feinberg, A. Gholami, K. Rothauge, M. Mahoney, J. Gonzalez, On the computational inefficiency of large batch sizes for stochastic gradient descent, 2018, ArXiv. abs/1811.12941.