A Second look at Exponential and Cosine Step Sizes: Simplicity, Adaptivity, and Performance

Xiaoyu Li [*1] **Zhenxun Zhuang** [*2] **Francesco Orabona** [1 2 3]

Abstract

Stochastic Gradient Descent (SGD) is a popular tool in training large-scale machine learning models. Its performance, however, is highly variable, depending crucially on the choice of the step sizes. Accordingly, a variety of strategies for tuning the step sizes have been proposed, ranging from coordinate-wise approaches (a.k.a. "adaptive" step sizes) to sophisticated heuristics to change the step size in each iteration. In this paper, we study two step size schedules whose power has been repeatedly confirmed in practice: the exponential and the cosine step sizes. For the first time, we provide theoretical support for them proving convergence rates for smooth non-convex functions, with and without the Polyak-Łojasiewicz (PL) condition. Moreover, we show the surprising property that these two strategies are *adaptive* to the noise level in the stochastic gradients of PL functions. That is, contrary to polynomial step sizes, they achieve almost optimal performance without needing to know the noise level nor tuning their hyperparameters based on it. Finally, we conduct a fair and comprehensive empirical evaluation of real-world datasets with deep learning architectures. Results show that, even if only requiring at most two hyperparameters to tune, these two strategies best or match the performance of various finely-tuned state-of-the-art strategies.

1. Introduction

In the last 10 years, non-convex machine learning formulations have received more and more attention as they can typically better scale with the complexity of the predictors and the amount of training data compared with convex ones. One such example is the deep neural networks. Over the years, various algorithms have been proposed and employed to optimize non-convex machine learning problems, among which Stochastic Gradient Descent (SGD) (Robbins & Monro, 1951) has become the most important ingredient in Machine Learning pipelines. Practitioners prefer it over more sophisticated methods for its simplicity and speed. Yet, this generality comes with a cost: SGD is far from the robustness of, e.g., second-order methods that require little to no tweaking of knobs to work. In particular, the step size is still the most important parameter to tune in the SGD algorithm, carrying the actual weight of making SGD adaptive to different situations.

The importance of step sizes in SGD is testified by the numerous proposed strategies to tune step sizes (e.g., Duchi et al., 2010; McMahan & Streeter, 2010; Tieleman & Hinton, 2012; Zeiler, 2012; Kingma & Ba, 2015). However, for most of them, there is little or no theory that can really explain their empirical success. Moreover, SGD with appropriate step sizes is already optimal in all the possible situations, so it is unclear what kind of advantage we might show.

An interesting viewpoint is to go beyond worst-case analyses and show that these learning rates provide SGD with some form of *adaptivity* to the characteristics of the function. More specifically, an algorithm is considered adaptive (or *universal*) if it has the best theoretical performance w.r.t. to a quantity X without the need to know it (Nesterov, 2015). So, for example, it is possible to design optimization algorithms adaptive to scale (Orabona & Pál, 2015; Orabona & Pál, 2018), smoothness (Levy et al., 2018), noise (Levy et al., 2018; Li & Orabona, 2019), and strong convexity (Cutkosky & Orabona, 2018). On the other hand, as noted in Orabona (2019), it is remarkable that even if most of the proposed step size strategies for SGD are called "adaptive", for most of them their analyses do not show any provable advantage over plain SGD nor any form of adaptation to the intrinsic characteristics of the non-convex function.

In this paper, we look at the two simple to use and empirically successful step size decay strategies, the *exponential* and the *cosine step size* (with and without

[*]Equal contribution [1]Division of System Engineering, Boston University, Boston, MA, US [2]Department of Computer Science, Boston University, Boston, MA, US [3]Department of Electrical & Computer Engineering, Boston University, Boston, MA, US. Correspondence to: Xiaoyu Li <xiaoyuli@bu.edu>, Zhenxun Zhuang <zxzhuang@bu.edu>.

Proceedings of the 38^{th} International Conference on Machine Learning, PMLR 139, 2021.

restarts) (Loshchilov & Hutter, 2017; He et al., 2019). The exponential step size is simply an exponential decaying step size. It is less discussed in the optimization literature and it is also unclear who proposed it first, even if it has been known to practitioners for a long time and already included in many deep learning software libraries (e.g., Abadi et al., 2015; Paszke et al., 2019). The cosine step size, which anneals the step size following a cosine function, has exhibited great power in practice but it does not have any theoretical justification.

For both these step size decay strategies, we prove *for the first time* a convergence guarantee. Moreover, we show that they have (unsuspected!) adaptation properties. Moreover, we also empirically test them showing that they have the best empirical performance among various state-of-the-art strategies. Finally, our proofs reveal the hidden similarity between these two step sizes.

Specifically, the contributions of this paper are:

- In the case when the function satisfies the PL condition (Polyak, 1963; Łojasiewicz, 1963; Karimi et al., 2016), both exponential step size and cosine step size strategies *automatically adapt to the level of noise of the stochastic gradients*.
- Without the PL condition, we show that SGD with either exponential step sizes or cosine step sizes has an *(almost) optimal convergence rate* for smooth non-convex functions.
- We also conduct an empirical evaluation on deep learning architectures: Exponential and cosine step sizes have essentially matching or better empirical performance than polynomial step decay, stagewise step decay, Adam (Kingma & Ba, 2015), and stochastic line search (Vaswani et al., 2019b), while requiring at most two hyperparameters.

The rest of the paper is organized as follows: We first discuss the relevant literature (Section 2). In Section 3, we introduce the notation, setting, and precise assumptions. Then, in Section 4 we describe in detail the step sizes and the theoretical guarantees. We show our empirical results in Section 5. Finally, we conclude with a discussion of the results and future work.

2. Related Work

Adaptation in non-convex optimization Adaptation is a general concept and an algorithm can be adaptive to any characteristic of the optimization problem. The idea is formalized in (Nesterov, 2015) with the equivalent name of *universality*, but it goes back at least to the "self-confident" strategies in online convex optimization (Auer et al., 2002). Indeed, the famous AdaGrad algorithm (McMahan & Streeter, 2010; Duchi et al., 2010) uses exactly that method to design an algorithm *adaptive to the gradients*. Nowadays, "adaptive step size" tend to denote coordinate-wise ones, with no guarantee of adaptation to any particular property. There is an abundance of adaptive optimization algorithm in the convex setting (e.g., McMahan & Streeter, 2010; Duchi et al., 2010; Kingma & Ba, 2015; Reddi et al., 2018), while only a few in the more challenging non-convex setting (e.g., Chen et al., 2018). The first analysis to show adaptivity to noise of non-convex SGD with appropriate step sizes is in Li & Orabona (2019) and later in Ward et al. (2019; 2020) under stronger assumptions. Then, Li & Orabona (2020) studied the adaptivity to noise of AdaGrad plus momentum, with a high probability analysis.

Exponential step size To the best of our knowledge, the exponential step size has been incorporated in Tensorflow (Abadi et al., 2015) and PyTorch (Paszke et al., 2019), yet no convergence guarantee have ever been proved for it. The closest strategy is the *stagewise step decay*, which corresponds to the discrete version of the exponential step size we analyze. The stagewise step decay uses a piece-wise constant step size strategy, where the step size is cut by a factor in each "stage". This strategy is known with many different names: "stagewise step size" (Yuan et al., 2019), "step decay schedule" (Ge et al., 2019), "geometrically decaying schedule" (Davis et al., 2021), and "geometric step decay" (Davis et al., 2019). In this paper, we will call it stagewise step decay. The stagewise step decay approach was first introduced in (Goffin, 1977) and used in many *convex* optimization problem (e.g., Hazan & Kale, 2011; Aybat et al., 2019; Kulunchakov & Mairal, 2019; Ge et al., 2019). Interestingly, Ge et al. (2019) also shows promising empirical results on non-convex functions, but instead of using their proposed decay strategy, they use an exponentially decaying schedule, like the one we analyze here. The only use of the stagewise step decay for non-convex functions we know are for sharp functions (Davis et al., 2019) and weakly-quasi-convex functions (Yuan et al., 2019). However, they do not show any adaptation property and they still do not consider the exponential step size but its discrete version. As far as we know, we prove the first theoretical guarantee for the exponential step size.

Cosine step decay Cosine step decay was originally presented in Loshchilov & Hutter (2017) with two tunable parameters. Later, He et al. (2019) proposed a simplified version of it with one parameter. However, there is no theory for this strategy though it is popularly used in the practical world (Liu et al., 2018; Zhang et al., 2019b; Lawen et al., 2019; Zhang et al., 2019a; Ginsburg et al., 2019; Cubuk et al., 2019; Zhao et al., 2020; You et al., 2020; Chen et al., 2020; Grill et al., 2020). As far as we know, we prove the first theoretical guarantee for the cosine step decay and the

first ones to hypothesize and prove the adaptation properties of the cosine decay step size.

SGD on non-convex smooth functions The first paper to analyze SGD on smooth functions with generic step sizes is Ghadimi & Lan (2013). Their analysis show that the optimal step size strategy strongly depends on the level of noise, but they do not offer any automatic strategy to adapt to it.

SGD with the PL condition The PL condition was proposed by Polyak (1963) and Łojasiewicz (1963). It is the weakest assumption we know to prove linear rates on non-convex functions. For SGD, Karimi et al. (2016) proved the rate of $O\left(1/\mu^2 T\right)$ for polynomial step sizes assuming Lipschitz and smooth functions, where μ is the PL constant. Note that the Lipschitz assumption hides the dependency of convergence and step sizes from the noise. It turns out that the Lipschitz assumption is not necessary to achieve the same rate, see Theorem 5 in the Appendix. Considering functions with finite-sum structure, Reddi et al. (2016), Lei et al. (2017) and Li et al. (2020) proved improved rates for variance reduction methods. The convergence rate that we show for the exponential step size is new in the literature on minimization of PL functions. Independently and the same time[1] with us, Khaled & Richtárik (2020) obtained the same convergence result in the PL condition for SGD with a stepsize that is constant in the first half and then decreases polynomially.

3. Problem Set-up

Notation We denote vectors by bold letters, e.g., $\boldsymbol{x} \in \mathbb{R}^d$. We denote by $\mathbb{E}[\cdot]$ the expectation with respect to the underlying probability space and by $\mathbb{E}_t[\cdot]$ the conditional expectation with respect to the past. Any norm in this work is the ℓ_2 norm.

Setting and Assumptions We consider the unconstrained optimization problem $\min_{\boldsymbol{x}\in\mathbb{R}^d} f(\boldsymbol{x})$, where $f(\boldsymbol{x}) : \mathbb{R}^d \to \mathbb{R}$ is a function bounded from below and we denote its infimum by $f^\star$. Note that we do *not* require f to be convex *nor* to have a finite-sum structure.

We focus on SGD, where, after an initialization of the first iterate as any $\boldsymbol{x}_1 \in \mathbb{R}^d$, in each round $t = 1, 2, \ldots, T$ we receive $\boldsymbol{g}_t$, an unbiased estimate of the gradient of f at point $\boldsymbol{x}_t$, i.e., $\mathbb{E}_t \boldsymbol{g}_t = \nabla f(\boldsymbol{x}_t)$. We update $\boldsymbol{x}_t$ with a step size η_t, i.e., $\boldsymbol{x}_{t+1} = \boldsymbol{x}_t - \eta_t \boldsymbol{g}_t$.

We assume that

(A1) f is L-*smooth*, i.e., f is differentiable and its gradient $\nabla f(\cdot)$ is L-Lipschitz, namely: $\forall \boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^d$, $\|\nabla f(\boldsymbol{x}) - \nabla f(\boldsymbol{y})\| \leq L\|\boldsymbol{x} - \boldsymbol{y}\|$. This implies for $\forall \boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^d$ (Nesterov, 2004, Lemma 1.2.3)

$$|f(\boldsymbol{y}) - f(\boldsymbol{x}) - \langle \nabla f(\boldsymbol{x}), \boldsymbol{y} - \boldsymbol{x}\rangle| \leq \frac{L}{2}\|\boldsymbol{y} - \boldsymbol{x}\|^2 . \quad (1)$$

(A2) f satisfies the μ-*PL* condition, that is, for some $\mu > 0$, $\frac{1}{2}\|\nabla f(\boldsymbol{x})\|^2 \geq \mu\left(f(\boldsymbol{x}) - f^\star\right), \forall \boldsymbol{x}$.

(A3) For $t = 1, 2, \ldots, T$, we assume $\mathbb{E}_t[\|\boldsymbol{g}_t - \nabla f(\boldsymbol{x}_t)\|^2] \leq a\|\nabla f(\boldsymbol{x}_t)\|^2 + b$, where $a, b \geq 0$.

Discussion on the assumptions It is worth stressing that non-convex functions are not characterized by a particular property, but rather from the lack of a specific property: convexity. In this sense, trying to carry out any meaningful analyses on the entire class of non-convex functions is hopeless. So, the assumptions we use balance the trade-off of *approximately* model many interesting machine learning problems while allowing to restrict the class of non-convex functions on particular subsets where we can underline interesting behaviours.

More in detail, the smoothness assumption (**A1**) is considered "weak" and ubiquitous in analyses of optimization algorithms in the non-convex setting. In many neural networks, it is only approximately true because ReLUs activation functions are non-smooth. However, if the number of training points is large enough, it is a good approximation of the loss landscape.

On the other hand, the PL condition (**A2**) is often considered a "strong" condition. However, it was formally proved to hold locally in deep neural networks in Allen-Zhu et al. (2019). Furthermore, Kleinberg et al. (2018) empirically observed that the loss surface of neural networks has good one-point convexity properties, and thus locally satisfies the PL condition. Of course, in our theorems we only need it to hold along the optimization path and not over the entire space, as also pointed out in Karimi et al. (2016). So, while being strong, it actually models the cases we are interested in. Moreover, dictionary learning (Arora et al., 2015), phase retrieval (Chen & Candes, 2015), and matrix completion (Sun & Luo, 2016), all satisfy the one-point convexity locally (Zhu, 2018), and in turn they all satisfy the PL condition locally.

Our assumption on the noise (**A3**) is strictly weaker than the common assumption of assuming a bounded variance, i.e., $\mathbb{E}_t[\|\boldsymbol{g}_t - \nabla f(\boldsymbol{x}_t)\|^2] \leq \sigma^2$. Indeed, our assumption recovers the bounded variance case with $a = 0$ while also allowing for the variance to grow unboundedly far from the optimum when $a > 0$. This is indeed the case when the optimal solution has low training error and the stochastic gradients are generated by mini-batches. This relaxed assumption on the noise was first used by Bertsekas & Tsitsiklis (1996) in the analysis of the asymptotic convergence of SGD.

[1] The first version of Khaled & Richtárik (2020) was released on Feb. 9th 2020 on ArXiv while our very first version was available online on Feb. 12th 2020 on ArXiv as well.

Exponential and Cosine Step Size We will use the following definition for the exponential step size

$$\eta_t = \eta_0 \cdot \alpha^t \tag{2}$$

and for cosine step sizes

$$\eta_t = \frac{\eta_0}{2}\left(1 + \cos\frac{t\pi}{T}\right), \tag{3}$$

where $\eta_0 = (L(1+a))^{-1}$. For the exponential step sizes, we use $\alpha = (\beta/T)^{1/T} \leq 1$, a and L are defined in (**A1**, **A3**), and $\beta \geq 1$.

4. Convergence and Adaptivity of Cosine and Exponential Step Sizes

Here, we present the guarantees of the exponential step size and the cosine step size and their adaptivity property.

4.1. Noise and Step Sizes

For the stochastic optimization of smooth functions, the noise plays a crucial role in setting the optimal step sizes: *To achieve the best performance, we need two completely different step size decay schemes in the noisy and noiseless case*. In particular, if the PL condition holds, in the noise-free case a constant step size is used to get a linear rate (i.e., exponential convergence), while in the noisy case the best rate $O(1/T)$ is given by time-varying step sizes $O(1/(\mu t))$ (Karimi et al., 2016). Similarly, without the PL condition, we still need a constant step size in the noise-free case for the optimal rate whereas a $O(1/\sqrt{t})$ step size is required in the noisy case (Ghadimi & Lan, 2013). Using a constant step size in noisy cases is of course possible, but the best guarantee we know is converging towards a neighborhood of the critical point or the optimum, instead of the exact convergence let alone the adaptivity to the noise, as shown in Theorem 2.1 of (Ghadimi & Lan, 2013) and Theorem 4 of (Karimi et al., 2016). Moreover, if the noise decreases over the course of the optimization, we should change the step size as well. Unfortunately, noise levels are rarely known or measured. On the other hand, an optimization algorithm *adaptive to noise* would always get the best performance without changing its hyperparameters.

In the following, we will show that exponential and cosine step sizes achieve exactly this adaptation to noise. It is worth reminding the reader that *any* polynomial decay of the step size does not give us this adaptation. So, let's gain some intuition on why this should happen with these two step sizes. In the early stage of the optimization process, we can expect that the disturbance due to the noise is relatively small compared to how far we are from the optimal solution. Accordingly, at this phase, a near-constant step size should be used. More precisely, the proofs shows that to achieve a linear rate we need $\sum_{t=1}^{T}\eta_t = \Omega(T)$ or even $\sum_{t=1}^{T}\eta_t = \Omega(T/\ln T)$. This is exactly what happens with (2) and (3). On the other hand, when the iterate is close to the optimal solution, we have to decrease the step size to fight with the effects of the noise. In this stage, the exponential step size goes to 0 as $O(1/T)$, which is the optimal step size used in the noisy case. Meanwhile, the last ith cosine step size is $\eta_{T-i} = \frac{\eta_0}{2}(1-\cos\frac{i\pi}{T}) = \eta_0 \sin^2\frac{i\pi}{2T}$, which amounts $O(1/T^2)$ when i is much smaller than T.

Hence, the analysis shows that (2) and (3) are surprisingly similar, smoothly varying from the near-constant behavior at the start and decreasing with a similar pattern towards the end, and both will be adaptive to the noise level. Next, we formalize these intuitions in convergence rates.

4.2. Convergence Guarantees

We now prove the convergence guarantees for these two step sizes. First, we consider the case where the function is smooth and satisfies the PL condition.

Theorem 1 (SGD with exponential step size)**.** *Assume (**A1**, **A2**, **A3**). For a given $T \geq \max\{3, \beta\}$ and $\eta_0 = (L(1+a))^{-1}$, with step size (2), SGD guarantees*

$$\mathbb{E}f(\boldsymbol{x}_{T+1}) - f^\star \leq \frac{5LC(\beta)}{e^2\mu^2}\frac{\ln^2\frac{T}{\beta}}{T}b + C(\beta)\exp\left(-\frac{0.69\mu}{L+a}\left(\frac{T}{\ln\frac{T}{\beta}}\right)\right)\cdot(f(\boldsymbol{x}_1) - f^\star),$$

where $C(\beta) \triangleq \exp\left((2\mu\beta)/(L(1+a)\ln T/\beta)\right)$.

Choice of β Note that if $\beta = L(1+a)/\mu$, we get

$$\mathbb{E}f(\boldsymbol{x}_{T+1}) - f^\star \leq O\left(\exp\left(-\frac{\mu}{L+a}\left(\frac{T}{\ln\frac{\mu T}{L}}\right)\right) + \frac{b\ln^2\frac{\mu T}{L}}{\mu^2 T}\right).$$

In words, this means that we are basically free to choose β, but will pay an exponential factor in the mismatch between β and $\frac{L}{\mu}$, which is basically the condition number for PL functions. This has to be expected because it also happens in the easier case of stochastic optimization of strongly convex functions (Bach & Moulines, 2011).

Theorem 2 (SGD with cosine step size)**.** *Assume (**A1**, **A2**, **A3**). For a given T and $\eta_0 = (L(1+a))^{-1}$, with step size (3), SGD guarantees*

$$\mathbb{E}f(\boldsymbol{x}_{t+1}) - f^\star \leq \exp\left(-\frac{\mu(T-1)}{2L(1+a)}\right)(f(x_1) - f^\star) + \frac{\pi^4 b}{32(1+a)T^4}\left(\left(\frac{8T^2}{\mu}\right)^{4/3} + \left(\frac{6T^2}{\mu}\right)^{\frac{5}{3}}\right).$$

Adaptivity to Noise From the above theorems, we can see that both the exponential step size and the cosine step size have a provable advantage over polynomial ones: *adaptivity to the noise*. Indeed, when $b = 0$, namely there is only noise relative to the distance from the optimum, they both guarantee a linear rate. Meanwhile, if there is noise, using the *same step size without any tuning*, the exponential step size recovers the rate of $O\left(1/(\mu^2 T)\right)$ while the cosine step size achieves the rate of $O(1/(\mu^{\frac{5}{3}} T^{\frac{2}{3}}))$ (up to poly-logarithmic terms). In contrast, polynomial step sizes would require two different settings—decaying vs constant—in the noisy vs no-noise situation (Karimi et al., 2016). It is worth stressing that the rate in Theorem 1 is one of the first results in the literature on stochastic optimization of smooth PL functions (Khaled & Richtárik, 2020).

Optimality of the bounds As far as we know, it is unknown if the rate we obtain for the optimization of non-convex smooth functions under the PL condition is optimal or not. However, up to poly-logarithmic terms, Theorem 1 matches at the same time the best-known rates for the noisy and deterministic cases (Karimi et al., 2016) (see also Theorem 5 in the Appendix). We would remind the reader that this rate is not comparable with the one for strongly convex functions which is $O(1/(\mu T))$. Meanwhile, cosine step size achieves a rate slightly worse in T (but better in μ) under the same assumptions.

Cosine Step Size with Restarts The original cosine stepsize was proposed with a restarting strategy, yet it has been commonly used without restarting and achieves good results (e.g., Loshchilov & Hutter, 2017; Gastaldi, 2017; Zoph et al., 2018; He et al., 2019; Cubuk et al., 2019; Liu et al., 2018; Zhao et al., 2020; You et al., 2020; Chen et al., 2020; Grill et al., 2020). Indeed, the previous theorem has confirmed that the cosine stepsize alone is well worth studying theoretically. Yet for completeness, we cover the analysis in a restart scheme for SGD with cosine stepsize in the PL condition in Appendix A.2. We obtain the same convergence rate μ and T as that in the case of no restarts under the PL condition.

Convergence without the PL condition The PL condition tells us that all stationary points are optimal points (Karimi et al., 2016), which is not always true for the parameter space in deep learning (Jin et al., 2017). However, this condition might still hold locally, for a considerable area around the local minimum. Indeed, as we said, this is exactly what was proven for deep neural networks (Allen-Zhu et al., 2019). The previous theorems tell us that once we reach the area where the geometry of the objective function satisfies the PL condition, we can get to the optimal point with an almost linear rate, depending on the noise. Nevertheless, we still have to be able to reach that region. Hence, in the following, we discuss the case where the PL condition is not satisfied and show for both step sizes that they are still able to move to a critical point at the optimal speed.

Theorem 3. *Assume **(A1)**, **(A3)** and $c > 1$. SGD with step sizes (2) with $\eta_0 = (cL(1+a))^{-1}$ guarantees*

$$\mathbb{E}\|\nabla f(\tilde{\boldsymbol{x}}_T)\|^2 \leq \frac{3Lc(a+1)\ln\frac{T}{\beta}}{T-\beta} \cdot (f(\boldsymbol{x}_1) - f^\star) + \frac{bT}{c(a+1)(T-\beta)},$$

where $\tilde{\boldsymbol{x}}_T$ is a random iterate drawn from $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_T$ with $\mathbb{P}[\tilde{\boldsymbol{x}}_T = \boldsymbol{x}_t] = \frac{\eta_t}{\sum_{i=1}^T \eta_i}$.

Theorem 4. *Assume **(A1)**, **(A3)** and $c > 1$. SGD with step sizes (3) with $\eta_0 = (cL(1+a))^{-1}$ guarantees*

$$\mathbb{E}\|\nabla f(\tilde{\boldsymbol{x}}_T)\|^2 \leq \frac{4Lc(a+1)}{T-1} \cdot (f(\boldsymbol{x}_1) - f^\star) + \frac{21bT}{4\pi^4 cL(a+1)(T-1)},$$

where $\tilde{\boldsymbol{x}}_T$ is a random iterate drawn from $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_T$ with $\mathbb{P}[\tilde{\boldsymbol{x}}_T = \boldsymbol{x}_t] = \frac{\eta_t}{\sum_{i=1}^T \eta_i}$.

If $b \neq 0$ in **(A3)**, setting $c \propto \sqrt{T}$ and $\beta = O(1)$ would give the $\tilde{O}(1/\sqrt{T})$ rate[2] and $O(1/\sqrt{T})$ for the exponential and cosine step size respectively. Note that the optimal rate in this setting is $O(1/\sqrt{T})$. On the other hand, if $b = 0$, setting $c = O(1)$ and $\beta = O(1)$ yields a $\tilde{O}(1/T)$ rate and $O(1/T)$ for the exponential and cosine step size respectively. It is worth noting that the condition $b = 0$ holds in many practical scenarios (Vaswani et al., 2019a). Note that both guarantees are optimal up to poly-logarithmic terms (Arjevani et al., 2019).

In the following, we present the main elements of the proofs of these theorems, leaving the technical details in the Appendix. The proofs also show the mathematical similarities between these two step sizes.

Proofs of the Theorems Given that the space is limited, we defer the proofs of Theorem 3 and Theorem 4 to the Appendix.

We first introduce some technical lemmas whose proofs are in the Appendix.

Lemma 1. *Assume **(A1)**, **(A3)**, and $\eta_t \leq \frac{1}{L(1+a)}$. SGD guarantees*

$$\mathbb{E}f(\boldsymbol{x}_{t+1}) - \mathbb{E}f(\boldsymbol{x}_t) \leq -\frac{\eta_t}{2}\mathbb{E}\|\nabla f(\boldsymbol{x}_t)\|^2 + \frac{L\eta_t^2 b}{2}\,. \quad (4)$$

Lemma 2. *Assume $X_k, A_k, B_k \geq 0, k = 1, ...,$ and $X_{k+1} \leq A_k X_k + B_k$, then we have*

$$X_{k+1} \leq \prod_{i=1}^{k} A_i X_1 + \sum_{i=1}^{k} \prod_{j=i+1}^{k} A_j B_i\,.$$

[2]The $\tilde{O}$ notations hides poly-logarithmic terms.

Lemma 3. *For $\forall T \geq 1$, we have $\sum_{t=1}^{T} \cos \frac{t\pi}{T} = -1$.*

Lemma 4. *For $T \geq 3$, $\alpha \geq 0.69$ and $\frac{\alpha^{T+1}}{(1-\alpha)} \leq \frac{2\beta}{\ln \frac{T}{\beta}}$.*

Lemma 5. $1 - x \leq \ln\left(\frac{1}{x}\right), \forall x > 0$.

Lemma 6. *Let $a, b \geq 0$. Then*

$$\sum_{t=0}^{T} \exp(-bt) t^a \leq 2\exp(-a)\left(\frac{a}{b}\right)^a + \frac{\Gamma(a+1)}{b^{a+1}}\,.$$

We can now prove both Theorem 1 and Theorem 2.

Proof of Theorem 1 and Theorem 2. Denote $\mathbb{E}f(\boldsymbol{x}_t) - f^\star$ by Δ_t. From Lemma 1 and the PL condition, we get

$$\Delta_{t+1} \leq (1 - \mu\eta_t)\Delta_t + \frac{L}{2}\eta_t^2 b^2\,.$$

By Lemma 2 and $1 - x \leq \exp(-x)$, we have

$$\begin{aligned}\Delta_{T+1} &\leq \prod_{t=1}^{T}(1-\mu\eta_t)\Delta_1 + \frac{L}{2}\sum_{t=1}^{T}\prod_{i=t+1}^{T}(1-\mu\eta_i)\eta_t^2 b \\ &\leq \exp\left(-\mu\sum_{t=1}^{T}\eta_t\right)\Delta_1 + \frac{Lb}{2}\sum_{t=1}^{T}\exp\left(-\mu\sum_{i=t+1}^{T}\eta_i\right)\eta_t^2.\end{aligned}$$

We then show that both the exponential step size and the cosine step size satisfy $\sum_{t=1}^{T}\eta_t = \Omega(T)$, which guarantees a linear rate in the noiseless case.

For the cosine step size (3), we observe that

$$\sum_{t=1}^{T}\eta_t = \frac{\eta_0 T}{2} + \frac{\eta_0}{2}\sum_{t=1}^{T}\cos\frac{t\pi}{T} = \frac{\eta_0(T-1)}{2},$$

where in the last equality we used Lemma 3.

Also, for the exponential step size (2), we can show

$$\begin{aligned}\sum_{t=1}^{T}\eta_t = \eta_0\frac{\alpha - \alpha^{T+1}}{1-\alpha} &\geq \frac{\eta_0\alpha}{1-\alpha} - \frac{2\eta_0\beta}{\ln\frac{T}{\beta}} \\ &\geq T\cdot\frac{0.69\eta_0}{\ln\frac{T}{\beta}} - \frac{2\eta_0\beta}{\ln\frac{T}{\beta}},\end{aligned}$$

where we used Lemma 4 in the first inequality and Lemma 5 in the second inequality.

Next, we upper bound $\sum_{t=1}^{T}\exp\left(-\mu\sum_{i=t+1}^{T}\eta_i\right)\eta_t^2$ for these two kinds of step sizes respectively.

For the exponential step size, by Lemma 4, we obtain

$$\begin{aligned}&\sum_{t=1}^{T}\exp\left(-\mu\sum_{i=t+1}^{T}\eta_i\right)\eta_t^2 \\ &= \eta_0^2\sum_{t=1}^{T}\exp\left(-\mu\eta_0\frac{\alpha^{t+1}-\alpha^{T+1}}{1-\alpha}\right)\alpha^{2t} \\ &\leq \eta_0^2 C(\beta)\sum_{t=1}^{T}\exp\left(-\frac{\mu\eta_0\alpha^{t+1}}{1-\alpha}\right)\alpha^{2t} \\ &\leq \eta_0^2 C(\beta)\sum_{t=1}^{T}\left(\frac{e}{2}\frac{\mu\alpha^{t+1}}{L(1+a)(1-\alpha)}\right)^{-2}\alpha^{2t} \\ &\leq \frac{4L^2(1+a)^2}{e^2\mu^2}\sum_{t=1}^{T}\frac{1}{\alpha^2}\ln^2\left(\frac{1}{\alpha}\right) \leq \frac{10L^2(1+a)^2\ln^2\frac{T}{\beta}}{e^2\mu^2 T},\end{aligned}$$

where in the second inequality we used $\exp(-x) \leq \left(\frac{\gamma}{ex}\right)^\gamma, \forall x > 0, \gamma > 0$.

For the cosine step size, using the fact that $\sin x \geq \frac{2}{\pi}x$ for $0 \leq x \leq \frac{\pi}{2}$, we can lower bound $\sum_{i=t+1}^{T}\eta_i$ by

$$\begin{aligned}\sum_{i=t+1}^{T}\eta_i &= \frac{\eta_0}{2}\sum_{i=t+1}^{T}\left(1+\cos\frac{i\pi}{T}\right) \\ &= \frac{\eta_0}{2}\sum_{i=0}^{T-t-1}\sin^2\frac{i\pi}{2T} \geq \frac{\eta_0}{2T^2}\sum_{i=0}^{T-t-1} i^2 \\ &\geq \frac{\eta_0(T-t-1)^3}{6T^2}\,.\end{aligned}$$

Then, we proceed

$$\begin{aligned}&\sum_{t=1}^{T}\exp\left(-\mu\sum_{i=t+1}^{T}\eta_i\right)\eta_t^2 \\ &\leq \frac{\eta_0^2}{4}\sum_{t=1}^{T}\left(1+\cos\frac{t\pi}{T}\right)^2\exp\left(-\frac{\mu\eta_0(T-t-1)^3}{6T^2}\right) \\ &= \frac{\eta_0^2}{4}\sum_{t=1}^{T-1}\left(1-\cos\frac{t\pi}{T}\right)^2\exp\left(-\frac{\eta_0\mu(t-1)^3}{6T^2}\right) \\ &= \eta_0^2\sum_{t=1}^{T-1}\sin^4\frac{t\pi}{2T}\exp\left(-\frac{\eta_0\mu(t-1)^3}{6T^2}\right) \\ &\leq \frac{\eta_0^2\pi^4}{16T^4}\sum_{t=0}^{T-1}t^4\exp\left(-\frac{\eta_0\mu t^3}{6T^2}\right) \\ &\leq \frac{\eta_0\pi^4}{16T^4}\left(2\exp\left(-\frac{4}{3}\right)\left(\frac{8T^2}{\mu}\right)^{4/3} + \left(\frac{6T^2}{\mu}\right)^{\frac{5}{3}}\right),\end{aligned}$$

where in the third line we used $\cos(\pi - x) = -\cos(x)$, in the forth line we used $1 - \cos(2x) = 2\sin^2(x)$, and in the last inequality we applied Lemma 6.

Putting things together, we get the stated bounds. □

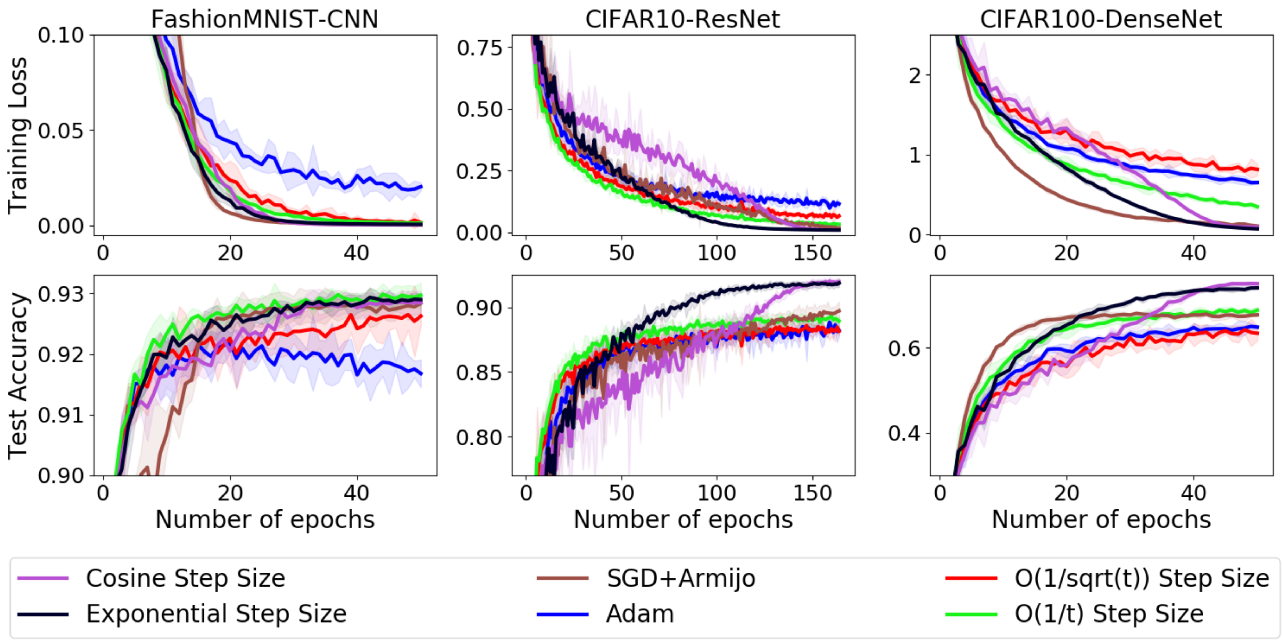

Figure 1. Training loss (top plots) and test accuracy (bottom plots) curves on employing different step size schedules to do image classification using a simple CNN for FashionMNIST (left), a 20-layer ResNet for CIFAR-10 (middle), and a 100-layer DenseNet on CIFAR-100 (right). *(The shading of each curve represents the 95% confidence interval computed across five independent runs from random initial starting points.)*

5. Empirical Results

The empirical performance of the cosine step size is already well-known in the applied world and does not require additional validation. However, both the exponential and the cosine step size are often missing as baselines in recent empirical evaluations. Hence, the main aim of this section is to provide a comparison of the exponential and cosine step sizes to other popular state-of-the-art step sizes methods. All experiments are done in PyTorch (Paszke et al., 2019) and the codes can be found at https://github.com/zhenxun-zhuang/SGD-Exponential-Cosine-Stepsize.

We performed experiments using deep neural networks to do image classification tasks on various datasets with different network architectures. Additionally, Appendix A.3.3 features an experiment on a Natural Language Processing (NLP) task, where the exponential and cosine step size strategies obtain better results than Adam (Kingma & Ba, 2015), the de-facto optimization method in NLP. Finally, in Appendix A.3.1, we include a synthetic experiment where those assumptions we need in analysis hold and show in detail the noise adaptation of both step sizes as predicted by the theory.

All models and experiments were carefully chosen to be easily reproducible.

Datasets We consider the image classification task on FashionMNIST and CIFAR-10/100 datasets. For all datasets, we select 10% training images as the validation set. Data augmentation and normalization are described in the Appendix.

Models For FashionMNIST, we use a CNN model consisting of two alternating stages of 5×5 convolutional filters and 2×2 max-pooling followed by one fully connected layer of 1024 units. To reduce overfitting, 50% dropout noise is used during training. For the CIFAR-10 dataset, we employ the 20-layer Residual Network model (He et al., 2016); and for CIFAR-100, we utilize the DenseNet-BC model (Huang et al., 2017) with 100 layers and a growth rate of 12. The loss is cross-entropy. The codes for implementing the latter two models can be found here[3] and here[4] respectively.

Training During the validation stage, we tune each method using the grid search (full details in the Appendix) to select the hyperparameters that work best according to their respective performance on the validation set. At the testing stage, the best performing hyperparameters from the validation stage are employed to train the model over all training images. The testing stage is repeated with random seeds for 5 times to eliminate the influence of stochasticity.

We use Nesterov momentum (Nesterov, 1983) of 0.9 without

[3] https://github.com/akamaster/pytorch_resnet_cifar10

[4] https://github.com/bearpaw/pytorch-classification

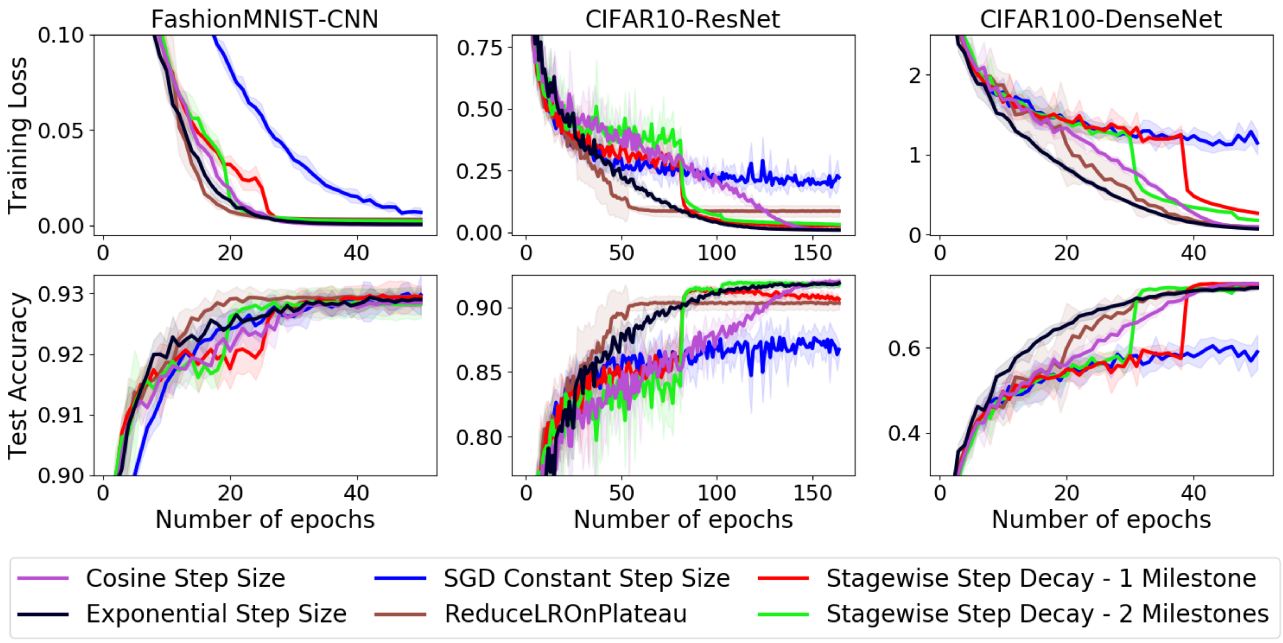

Figure 2. Training loss (top plots) and test accuracy (bottom plots) curves comparing the exponential and cosine step sizes with stagewise step decay for image classification using a simple CNN for FashionMNIST (left), a 20-layer ResNet for CIFAR-10 (middle), and a 100-layer DenseNet on CIFAR-100 (right). *(The shading of each curve represents the 95% confidence interval computed across five independent runs from random initial starting points.)*

dampening (if having this option), weight-decay of 0.0001 (FashionMNIST and CIFAR-10) and 0.0005 (CIFAR100), and use a batch size of 128. Regarding the employment of Nesterov momentum, we follow the setting of Ge et al. (2019). The use of momentum is essential to have a fair and realistic comparison in that the majority of practitioners would use it when using SGD.

Optimization methods We consider SGD with the following step size decay schedules:

$$\begin{aligned} &\eta_t = \eta_0 \cdot \alpha^t; \quad \eta_t = \eta_0(1+\alpha\sqrt{t})^{-1}; \\ &\eta_t = \eta_0(1+\alpha t)^{-1}; \quad \eta_t = \eta_0/2\left(1+\cos\left(t\pi/T\right)\right), \end{aligned} \tag{5}$$

where t is the iteration number (instead of the number of epochs). We also compare with Adam (Kingma & Ba, 2015), SGD+Armijo (Vaswani et al., 2019b), PyTorch's ReduceLROnPlateau scheduler[5] and stagewise step decay. In the following, we will call the place of decreasing the step size in stagewise step decay a **milestone**. (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., Aybat et al., 2019).)

Results and discussions The exact loss and accuracy values are reported in Table 1. To avoid overcrowding the figures, we compare the algorithms in groups of baselines. The comparison of performance between step size schemes listed in (5), Adam, and SGD+Armijo are shown in Figure 1. As can be seen, the *only* two methods that perform well on *all* 3 datasets are cosine and exponential step size. In particular, cosine step size performs the best across datasets both in training loss and test accuracy, with the exponential step size following closely.

On the other hand, as we noted above, stagewise step decay is a very popular decay schedule in deep learning. Thus, our second group of baselines in Figure 2 is composed by the stagewise step decay, ReduceLROnPlateau, and SGD with constant stepsize. The results show that exponential and cosine step sizes can still match or exceed the best of them with a fraction of their needed time to find the best hyperparameters. Indeed, we need 4 hyperparameters for two milestones, 3 for one milestone, and at least 4 for ReduceLROnPlateau. In contrast, the cosine step size requires only 1 hyperparameter and the exponential one needs 2.

Note that we do not pretend that our benchmark of the stagewise step decay is exhaustive. Indeed, there are many unexplored (potentially infinite!) possible hyperparameter settings. For example, it is reasonable to expect that adding even more milestones at the appropriate times could lead to better performance. However, this would result in a linear growth of the number of hyperparameters leading to an exponential increase in the number of possible location combinations.

[5] https://pytorch.org/docs/stable/optim.html

Table 1. Average final training loss and test accuracy achieved by each method when optimizing respective models on each dataset. The $\pm$ shows 95% confidence intervals of the mean loss/accuracy value over 5 runs starting from different random seeds.

Methods	FashionMNIST		CIFAR10		CIFAR100	
	Training loss	Test accuracy	Training loss	Test accuracy	Training loss	Test accuracy
SGD Constant Step Size	0.0068 ± 0.0023	0.9297 ± 0.0033	0.2226 ± 0.0169	0.8674 ± 0.0048	1.1467 ± 0.1437	0.5896 ± 0.0404
$O(1/t)$ Step Size	0.0013 ± 0.0004	$\mathbf{0.9297 \pm 0.0021}$	0.0331 ± 0.0028	0.8894 ± 0.0040	0.3489 ± 0.0263	0.6874 ± 0.0076
$O(1/\sqrt{t})$ Step Size	0.0016 ± 0.0005	0.9262 ± 0.0014	0.0672 ± 0.0086	0.8814 ± 0.0034	0.8147 ± 0.0717	0.6336 ± 0.0169
Adam	0.0203 ± 0.0021	0.9168 ± 0.0023	0.1161 ± 0.0111	0.8823 ± 0.0041	0.6513 ± 0.0154	0.6478 ± 0.0054
SGD+Armijo	$\mathbf{0.0003 \pm 0.0000}$	0.9284 ± 0.0016	0.0185 ± 0.0043	0.8973 ± 0.0071	0.1063 ± 0.0153	0.6768 ± 0.0044
ReduceLROnPlateau	0.0031 ± 0.0009	0.9294 ± 0.0015	0.0867 ± 0.0230	0.9033 ± 0.0049	0.0927 ± 0.0085	0.7435 ± 0.0076
Stagewise - 1 Milestone	0.0007 ± 0.0002	0.9294 ± 0.0018	0.0269 ± 0.0017	0.9062 ± 0.0020	0.2673 ± 0.0084	0.7459 ± 0.0030
Stagewise - 2 Milestones	0.0023 ± 0.0005	0.9283 ± 0.0024	0.0322 ± 0.0008	0.9174 ± 0.0020	0.1783 ± 0.0030	0.7487 ± 0.0025
Exponential Step Size	0.0006 ± 0.0001	0.9290 ± 0.0009	$\mathbf{0.0098 \pm 0.0010}$	0.9188 ± 0.0033	$\mathbf{0.0714 \pm 0.0041}$	0.7398 ± 0.0037
Cosine Step Size	0.0004 ± 0.0000	0.9285 ± 0.0019	0.0106 ± 0.0008	$\mathbf{0.9199 \pm 0.0029}$	0.0949 ± 0.0053	$\mathbf{0.7497 \pm 0.0044}$

This in turn causes the rapid growth of tuning time in selecting a good set of milestones in practice. Worse still, even the intuition that one should decrease the step size once the test loss curve stops decreasing is not always correct. Indeed, we observed in experiments (see Figure 3) that doing this will, after the initial drop of the curve in response to the step size decrease, make the test loss curve gradually go up again.

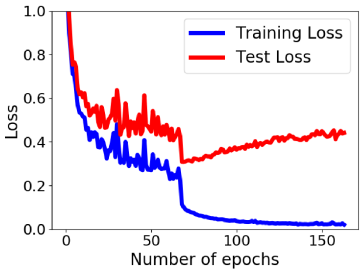

Figure 3. Plot showing that decreasing the step size too soon would lead to overfitting (ResNet20 on CIFAR10).

6. Conclusion

We have analyzed theoretically and empirically the exponential and cosine step size, two successful step size decay schedules for the stochastic optimization of non-convex functions. We have shown that, up to poly-logarithmic terms, both step sizes guarantee convergence with the best-known rates for smooth non-convex functions. Moreover, in the case of functions satisfying the PL condition, we have also proved that they are both adaptive to the level of noise. Furthermore, we have validated our theoretical findings on both synthetic and real-world tasks, showing that these two step sizes consistently match or outperform other strategies, while at the same time requiring only 1 (cosine) / 2 (exponential) hyperparameters to tune.

In future work, we plan to extend our theoretical results. For example, high probability bounds are easy to be obtained from our results.

Acknowledgements

This material is based upon work supported by the National Science Foundation under grants no. 1925930 “Collaborative Research: TRIPODS Institute for Optimization and Learning”, no. 1908111 “AF: Small: Collaborative Research: New Representations for Learning Algorithms and Secure Computation”, and no. 2022446 “Foundations of Data Science Institute”.

References

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., and Zheng, X. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015.

Allen-Zhu, Z., Li, Y., and Song, Z. A convergence theory for deep learning via over-parameterization. In *International Conference on Machine Learning*, pp. 242–252. PMLR, 2019.

Arjevani, Y., Carmon, Y., Duchi, J. C., Foster, D. J., Srebro, N., and Woodworth, B. Lower bounds for non-convex stochastic optimization. *arXiv preprint arXiv:1912.02365*, 2019.

Arora, S., Ge, R., Ma, T., and Moitra, A. Simple, efficient, and neural algorithms for sparse coding. In P, G., Hazan, E., and Kale, S. (eds.), *Proc. of The 28th Conference on Learning Theory*, volume 40 of *Proc. of Machine Learning Research*, pp. 113–149, Paris, France, 03–06 Jul 2015. PMLR.

Auer, P., Cesa-Bianchi, N., and Gentile, C. Adaptive and self-confident on-line learning algorithms. *J. Comput. Syst. Sci.*, 64(1):48–75, 2002.

Aybat, N. S., Fallah, A., Gurbuzbalaban, M., and Ozdaglar, A. A universally optimal multistage accelerated stochastic gradient method. In *Advances in Neural Information Processing Systems*, pp. 8523–8534, 2019.

Bach, F. and Moulines, E. Non-asymptotic analysis of stochastic approximation algorithms for machine learning. In Shawe-Taylor, J., Zemel, R. S., Bartlett, P. L., Pereira, F., and Weinberger, K. Q. (eds.), *Advances in Neural Information Processing Systems 24*, pp. 451–459. Curran Associates, Inc., 2011.

Berrada, L., Zisserman, A., and Kumar, M. Deep Frank-Wolfe for neural network optimization. *International Conference on Learning Representations*, 2019.

Bertsekas, D. P. and Tsitsiklis, J. N. *Neuro-Dynamic Programming*. Athena Scientific, 1996.

Bowman, S., Angeli, G., Potts, C., and Manning, C. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, 2015.

Chen, J., Zhou, D., Tang, Y., Yang, Z., and Gu, Q. Closing the generalization gap of adaptive gradient methods in training deep neural networks. *arXiv preprint arXiv:1806.06763*, 2018.

Chen, T., Kornblith, S., Norouzi, M., and Hinton, G. A simple framework for contrastive learning of visual representations. In Daumé, III, H. and Singh, A. (eds.), *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 1597–1607. PMLR, 13–18 Jul 2020.

Chen, Y. and Candes, E. Solving random quadratic systems of equations is nearly as easy as solving linear systems. In Cortes, C., Lawrence, N. D., Lee, D. D., Sugiyama, M., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 28*, pp. 739–747. Curran Associates, Inc., 2015.

Conneau, A., Kiela, D., Schwenk, H., Barrault, L., and Bordes, A. Supervised learning of universal sentence representations from natural language inference data. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 670–680, Copenhagen, Denmark, September 2017. Association for Computational Linguistics.

Cubuk, E. D., Zoph, B., Mane, D., Vasudevan, V., and Le, Q. V. Autoaugment: Learning augmentation strategies from data. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 113–123, 2019.

Cutkosky, A. and Orabona, F. Black-box reductions for parameter-free online learning in Banach spaces. In *Proc. of the Conference on Learning Theory (COLT)*, 2018.

Davis, D., Drusvyatskiy, D., and Charisopoulos, V. Stochastic algorithms with geometric step decay converge linearly on sharp functions. *arXiv preprint arXiv:1907.09547*, 2019.

Davis, D., Drusvyatskiy, D., Xiao, L., and Zhang, J. From low probability to high confidence in stochastic convex optimization. *Journal of Machine Learning Research*, 22 (49):1–38, 2021.

Duchi, J., Hazan, E., and Singer, Y. Adaptive subgradient methods for online learning and stochastic optimization. In *COLT*, 2010.

Gastaldi, X. Shake-shake regularization of 3-branch residual networks. In *Proc. of the International Conference on Learning Representations*, 2017.

Ge, R., Kakade, S. M., Kidambi, R., and Netrapalli, P. The step decay schedule: A near optimal, geometrically decaying learning rate procedure for least squares. In *Advances in Neural Information Processing Systems*, pp. 14951–14962, 2019.

Ghadimi, S. and Lan, G. Stochastic first- and zeroth-order methods for nonconvex stochastic programming. *SIAM Journal on Optimization*, 23(4):2341–2368, 2013.

Ginsburg, B., Castonguay, P., Hrinchuk, O., Kuchaiev, O., Lavrukhin, V., Leary, R., Li, J., Nguyen, H., Zhang, Y., and Cohen, J. M. Stochastic gradient methods with layer-wise adaptive moments for training of deep networks. *arXiv preprint arXiv:1905.11286*, 2019.

Goffin, J.-L. On convergence rates of subgradient optimization methods. *Mathematical programming*, 13(1): 329–347, 1977.

Grill, J., Strub, F., Altché, F., Tallec, C., Richemond, P., Buchatskaya, E., Doersch, C., Avila P., B., Guo, Z., Gheshlaghi Azar, M., Piot, B., kavukcuoglu, K., Munos, R., and Valko, M. Bootstrap your own latent - a new approach to self-supervised learning. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M. F., and Lin, H. (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 21271–21284. Curran Associates, Inc., 2020.

Hazan, E. and Kale, S. Beyond the regret minimization barrier: an optimal algorithm for stochastic strongly-convex optimization. In Kakade, S. M. and von Luxburg, U. (eds.), *Proc. of the 24th Annual Conference on Learning Theory*, volume 19 of *Proc. of Machine Learning Research*, pp. 421–436, Budapest, Hungary, 09–11 Jun 2011. PMLR.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proc. of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

He, T., Zhang, Z., Zhang, H., Zhang, Z., Xie, J., and Li, M. Bag of tricks for image classification with convolutional neural networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

Huang, G., Liu, Z., Van Der Maaten, L., and Weinberger, K. Q. Densely connected convolutional networks. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pp. 4700–4708, 2017.

Jin, C., Ge, R., Netrapalli, P., Kakade, S., and Jordan, M. I. How to escape saddle points efficiently. In *Proc. of the 34th International Conference on Machine Learning*, volume 70, pp. 1724–1732. PMLR, 2017.

Karimi, H., Nutini, J., and Schmidt, M. Linear convergence of gradient and proximal-gradient methods under the Polyak-Łojasiewicz condition. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pp. 795–811. Springer, 2016.

Khaled, A. and Richtárik, P. Better theory for SGD in the nonconvex world. *arXiv preprint arXiv:2002.03329*, 2020.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, 2015.

Kleinberg, B., Li, Y., and Yuan, Y. An alternative view: When does SGD escape local minima? In Dy, J. and Krause, A. (eds.), *Proc. of the 35th International Conference on Machine Learning*, volume 80 of *Proc. of Machine Learning Research*, pp. 2698–2707, Stockholmsmässan, Stockholm Sweden, 10–15 Jul 2018. PMLR.

Kulunchakov, A. and Mairal, J. A generic acceleration framework for stochastic composite optimization. In *Advances in Neural Information Processing Systems*, pp. 12556–12567, 2019.

Lawen, H., Ben-Cohen, A., Protter, M., Friedman, I., and Zelnik-Manor, L. Attention network robustification for person ReID. *arXiv preprint arXiv:1910.07038*, 2019.

Lee, C., Xie, S., Gallagher, P., Zhang, Z., and Tu, Z. Deeply-supervised nets. In *Proc. of the Eighteenth International Conference on Artificial Intelligence and Statistics*, volume 38, pp. 562–570. PMLR, 2015.

Lei, L., Ju, C., Chen, J., and Jordan, M. I. Non-convex finite-sum optimization via SCSG methods. In *Advances in Neural Information Processing Systems 30*, pp. 2348–2358. Curran Associates, Inc., 2017.

Levy, K. Y., Yurtsever, A., and Cevher, V. Online adaptive methods, universality and acceleration. In *Advances in Neural Information Processing Systems*, pp. 6500–6509, 2018.

Li, X. and Orabona, F. On the convergence of stochastic gradient descent with adaptive stepsizes. In *Proc. of the 22nd International Conference on Artificial Intelligence and Statistics, AISTATS*, 2019.

Li, X. and Orabona, F. A high probability analysis of adaptive SGD with momentum. In *ICML 2020 Workshop on Beyond First Order Methods in ML Systems*, 2020.

Li, Z., Bao, H., Zhang, X., and Richtárik, P. Page: A simple and optimal probabilistic gradient estimator for nonconvex optimization. *arXiv preprint arXiv:2008.10898*, 2020.

Liu, H., Simonyan, K., and Yang, Y. Darts: Differentiable architecture search. In *International Conference on Learning Representations*, 2018.

Łojasiewicz, S. A topological property of real analytic subsets (in french). *Coll. du CNRS, Les équations aux dérivées partielles*, pp. 87–89, 1963.

Loshchilov, I. and Hutter, F. SGDR: Stochastic gradient descent with warm restarts. In *International Conference on Learning Representations*, 2017.

McMahan, H. B. and Streeter, M. J. Adaptive bound optimization for online convex optimization. In *COLT*, 2010.

Nesterov, Y. A method for unconstrained convex minimization problem with the rate of convergence $O(1/k^2)$. In *Doklady AN SSSR (translated as Soviet. Math. Docl.)*, volume 269, pp. 543–547, 1983.

Nesterov, Y. *Introductory lectures on convex optimization: A basic course*, volume 87. Springer, 2004.

Nesterov, Y. Universal gradient methods for convex optimization problems. *Mathematical Programming*, 152 (1-2):381–404, 2015.

Orabona, F. A modern introduction to online learning. *arXiv preprint arXiv:1912.13213*, 2019.

Orabona, F. and Pál, D. Scale-free algorithms for online linear optimization. In *International Conference on Algorithmic Learning Theory*, pp. 287–301. Springer, 2015.

Orabona, F. and Pál, D. Scale-free online learning. *Theoretical Computer Science*, 716:50–69, 2018. Special Issue on ALT 2015.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pp. 8024–8035. Curran Associates, Inc., 2019.

Pennington, J., Socher, R., and Manning, C. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1532–1543, 2014.

Polyak, B. T. Gradient methods for minimizing functionals. *Zhurnal Vychislitel'noi Matematiki i Matematicheskoi Fiziki*, 3(4):643–653, 1963.

Reddi, S. J., Hefny, A., Sra, S., Poczos, B., and Smola, A. Stochastic variance reduction for nonconvex optimization. In *International conference on machine learning*, pp. 314–323, 2016.

Reddi, S. J., Kale, S., and Kumar, S. On the convergence of Adam and beyond. In *International Conference on Learning Representations*, 2018.

Robbins, H. and Monro, S. A stochastic approximation method. *Annals of Mathematical Statistics*, 22:400–407, 1951.

Sun, R. and Luo, Z. Guaranteed matrix completion via nonconvex factorization. *IEEE Transactions on Information Theory*, 62(11):6535–6579, 2016.

Tieleman, T. and Hinton, G. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural Networks for Machine Learning*, 2012.

Vaswani, S., Bach, F., and Schmidt, M. Fast and faster convergence of SGD for over-parameterized models and an accelerated Perceptron. In Chaudhuri, K. and Sugiyama, M. (eds.), *Proc. of the 22nd International Conference on Artificial Intelligence and Statistics*, volume 89 of *Proc. of Machine Learning Research*, pp. 1195–1204. PMLR, 16–18 Apr 2019a.

Vaswani, S., Mishkin, A., Laradji, I., Schmidt, M., Gidel, G., and Lacoste-Julien, S. Painless stochastic gradient: Interpolation, line-search, and convergence rates. In *Advances in Neural Information Processing Systems*, pp. 3727–3740, 2019b.

Ward, R., Wu, X., and Bottou, L. AdaGrad stepsizes: Sharp convergence over nonconvex landscapes. In *International Conference on Machine Learning*, pp. 6677–6686. PMLR, 2019.

Ward, R., Wu, X., and Bottou, L. AdaGrad stepsizes: Sharp convergence over nonconvex landscapes. *Journal of Machine Learning Research*, 21(219):1–30, 2020.

You, J., Leskovec, J., He, K., and Xie, S. Graph structure of neural networks. In *International Conference on Machine Learning*, pp. 10881–10891. PMLR, 2020.

Yuan, Z., Yan, Y., Jin, R., and Yang, T. Stagewise training accelerates convergence of testing error over sgd. In *Advances in Neural Information Processing Systems*, pp. 2604–2614, 2019.

Zeiler, M. D. ADADELTA: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012.

Zhang, X., Wang, Q., Zhang, J., and Zhong, Z. Adversarial autoaugment. In *International Conference on Learning Representations*, 2019a.

Zhang, Z., Wu, Y., and Wang, G. Bpgrad: Towards global optimality in deep learning via branch and pruning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3301–3309, 2018.

Zhang, Z., He, T., Zhang, H., Zhang, Z., Xie, J., and Li, M. Bag of freebies for training object detection neural networks. *arXiv preprint arXiv:1902.04103*, 2019b.

Zhao, H., Jia, J., and Koltun, V. Exploring self-attention for image recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

Zhou, Z., Mertikopoulos, P., Bambos, N., Boyd, S., and Glynn, P. W. Stochastic mirror descent in variationally coherent optimization problems. In *Advances in Neural Information Processing Systems*, pp. 7043–7052, 2017.

Zhu, Z. Natasha 2: Faster non-convex optimization than SGD. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 31*, pp. 2675–2686. Curran Associates, Inc., 2018.

Zoph, B., Vasudevan, V., Shlens, J., and Le, Q. V. Learning transferable architectures for scalable image recognition. In *Proc. of the IEEE conference on computer vision and pattern recognition*, pp. 8697–8710, 2018.