Training Neural Networks for and by Interpolation

Leonard Berrada [1] **Andrew Zisserman** [2] **M. Pawan Kumar** [2]

Abstract

In modern supervised learning, many deep neural networks are able to interpolate the data: the empirical loss can be driven to near zero on all samples simultaneously. In this work, we explicitly exploit this interpolation property for the design of a new optimization algorithm for deep learning, which we term Adaptive Learning-rates for Interpolation with Gradients (ALI-G). ALI-G retains the two main advantages of Stochastic Gradient Descent (SGD), which are (i) a low computational cost per iteration and (ii) good generalization performance in practice. At each iteration, ALI-G exploits the interpolation property to compute an adaptive learning-rate in closed form. In addition, ALI-G clips the learning-rate to a maximal value, which we prove to be helpful for non-convex problems. Crucially, in contrast to the learning-rate of SGD, the maximal learning-rate of ALI-G does not require a decay schedule, which makes it considerably easier to tune. We provide convergence guarantees of ALI-G in various stochastic settings. Notably, we tackle the realistic case where the interpolation property is satisfied up to some tolerance. We provide experiments on a variety of architectures and tasks: (i) learning a differentiable neural computer; (ii) training a wide residual network on the SVHN data set; (iii) training a Bi-LSTM on the SNLI data set; and (iv) training wide residual networks and densely connected networks on the CIFAR data sets. ALI-G produces state-of-the-art results among adaptive methods, and even yields comparable performance with SGD, which requires manually tuned learning-rate schedules. Furthermore, ALI-G is simple to implement in any standard deep learning framework and can be used as a drop-in replacement in existing code.

1. Introduction

Training a deep neural network is a challenging optimization problem: it involves minimizing the average of many high-dimensional non-convex functions. In practice, the main algorithms of choice are Stochastic Gradient Descent (SGD) (Robbins & Monro, 1951) and adaptive gradient methods such as AdaGrad (Duchi et al., 2011) or Adam (Kingma & Ba, 2015). It has been observed that SGD tends to provide better generalization performance than adaptive gradient methods (Wilson et al., 2017). However, the downside of SGD is that it requires the manual design of a learning-rate schedule, which is widely regarded as an onerous and time consuming task. In this work, we alleviate this issue with the design of an adaptive learning-rate algorithm that needs minimal tuning for good performance. Indeed, we postulate that by using the same descent direction as SGD while automatically adapting its learning-rate, the resulting algorithm can offer similar generalization performance while requiring considerably less tuning.

In this work, we build on the following two ideas. First, an adaptive learning-rate can be computed for the non-stochastic gradient direction when the minimum value of the objective function is known (Polyak, 1969; Shor, 1985; Brännlund, 1995; Nedić & Bertsekas, 2001a;b). And second, one such minimum value is usually approximately known for interpolating models: for instance, it is close to zero for a model trained with the cross-entropy loss. By carefully combining these two ideas, we create a stochastic algorithm that (i) provably converges fast in convex or Restricted Secant Inequality (RSI) settings, and (ii) obtains state-of-the-art empirical results with neural networks. We refer to this algorithm as Adaptive Learning-rates for Interpolation with Gradients (ALI-G).

Procedurally, ALI-G is close to many existing algorithms, such as Deep Frank-Wolfe (Berrada et al., 2019), APROX (Asi & Duchi, 2019) and L_4 (Rolinek & Martius, 2018). And yet uniquely, thanks to its careful design and analysis, the learning-rate of ALI-G effectively requires a single hyper-parameter that does not need to be decayed. Since ALI-G is easy to implement in any deep learning framework, we believe that it can prove to be a practical and reliable optimization tool for deep learning.

Contributions. We summarize the contributions of this work as follows:

- We design an adaptive learning-rate algorithm that uses a single hyper-parameter and does need any decaying sched-

[1]DeepMind, London, United Kingdom. Work performed while at University of Oxford. [2]Department of Engineering Science, University of Oxford, Oxford, United Kingdom. Correspondence to: Leonard Berrada <lberrada@google.com>.

Proceedings of the 37^{th} International Conference on Machine Learning, Vienna, Austria, PMLR 119, 2020.

ule. In contrast, the closely related APROX (Asi & Duchi, 2019) and L_4 (Rolinek & Martius, 2018) use respectively two and four hyper-parameters for their learning-rate.
- We provide convergence rates of ALI-G in various stochastic convex settings. Importantly, our theoretical results take into account the error in the estimate of the minimum objective value. To the best of our knowledge, our work is the first to establish convergence rates for interpolation with approximate estimates.
- We prove that using a maximal learning-rate helps with convergence for a class of non-convex problems.
- We demonstrate state-of-the-art results for ALI-G on learning a differentiable neural computer; training variants of residual networks on the SVHN and CIFAR data sets; and training a Bi-LSTM on the Stanford Natural Language Inference data set.

2. The Algorithm

2.1. Problem Setting

Loss Function. We consider a supervised learning task where the model is parameterized by $\boldsymbol{w} \in \mathbb{R}^p$. Usually, the objective function can be expressed as an expectation over $z \in \mathcal{Z}$, a random variable indexing the samples of the training set:

$$f(\boldsymbol{w}) \triangleq \mathbb{E}_{z\in\mathcal{Z}}[\ell_z(\boldsymbol{w})], \tag{1}$$

where each ℓ_z is the loss function associated with the sample z. We assume that each ℓ_z is non-negative, which is the case for the large majority of loss functions used in machine learning. For instance, suppose that the model is a deep neural network with weights $\boldsymbol{w}$ performing classification. Then for each sample z, $\ell_z(\boldsymbol{w})$ can represent the cross-entropy loss, which is always non-negative. Other non-negative loss functions include the structured or multi-class hinge loss, and the ℓ_1 or ℓ_2 loss functions for regression.

Regularization. It is often desirable to employ a regularization function ϕ in order to promote generalization. In this work, we incorporate such regularization as a constraint on the feasible domain: $\Omega = \{\boldsymbol{w} \in \mathbb{R}^p : \phi(\boldsymbol{w}) \leq r\}$ for some value of r. In the deep learning setting, this will allow us to assume that the objective function can be driven close to zero without unrealistic assumptions about the regularization. Our framework can handle any constraint set Ω on which Euclidean projections are computationally efficient. This includes the feasible set induced by ℓ_2 regularization: $\Omega = \left\{\boldsymbol{w} \in \mathbb{R}^p : \|\boldsymbol{w}\|_2^2 \leq r\right\}$, for which the projection is given by a simple rescaling of $\boldsymbol{w}$. Finally, note that if we do not wish to use any regularization, we define $\Omega = \mathbb{R}^p$ and the corresponding projection is the identity.

Problem Formulation. The learning task can be expressed as the problem $(\mathcal{P})$ of finding a feasible vector of parameters $\mathbf{w}_\star \in \Omega$ that minimizes f:

$$\mathbf{w}_\star \in \operatorname*{argmin}_{\boldsymbol{w}\in\Omega} f(\boldsymbol{w}). \tag{$\mathcal{P}$}$$

Also note that $f_\star$ refers to the minimum value of f over Ω: $f_\star \triangleq \min_{\boldsymbol{w}\in\Omega} f(\boldsymbol{w})$.

Interpolation. We say that the problem $(\mathcal{P})$ satisfies the interpolation assumption if there exist a solution $\mathbf{w}_\star$ that simultaneously minimizes all individual loss functions:

$$\forall z \in \mathcal{Z},\ \ell_z(\mathbf{w}_\star) = 0. \tag{2}$$

The condition (2) can be equivalently expressed as $f_\star = 0$. We also point out that in some cases, it can be more realistic to relax (2) to $\forall z \in \mathcal{Z},\ \ell_z(\mathbf{w}_\star) \leq \varepsilon$ for a small positive ε.

2.2. The Polyak Step-Size

Before outlining the ALI-G algorithm, we begin with a brief description of the Polyak step-size, from which ALI-G draws some fundamental ideas.

Setting. We assume that $f_\star$ is known and we use non-stochastic updates: at each iteration, the full objective f and its derivative are evaluated. We denote by $\nabla f(\boldsymbol{w})$ the first-order derivative of f at $\boldsymbol{w}$ (e.g. $\nabla f(\boldsymbol{w})$ can be a sub-gradient or the gradient). In addition, $\|\cdot\|$ is the standard Euclidean norm in $\mathbb{R}^p$, and $\Pi_\Omega(\boldsymbol{w})$ is the Euclidean projection of the vector $\boldsymbol{w} \in \mathbb{R}^p$ on the set Ω.

Polyak Step-Size. At time-step t, using the Polyak step-size (Polyak, 1969; Shor, 1985; Brännlund, 1995; Nedić & Bertsekas, 2001a;b) yields the following update:

$$\boldsymbol{w}_{t+1} = \Pi_\Omega\left(\boldsymbol{w}_t - \gamma_t \nabla f(\boldsymbol{w}_t)\right), \text{ where } \gamma_t \triangleq \frac{f(\boldsymbol{w}_t) - f_\star}{\|\nabla f(\boldsymbol{w}_t)\|^2}, \tag{3}$$

where we loosely define $\frac{0}{0} = 0$ for simplicity purposes.

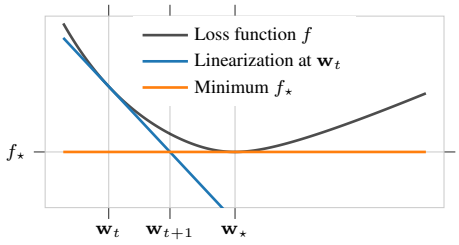

Figure 1. Illustration of the Polyak step-size in 1D. In this case, and further assuming that $f_\star = 0$, the algorithm coincides with the Newton-Raphson method for finding roots of a function.

Interpretation. It can be shown that $\boldsymbol{w}_{t+1}$ lies on the intersection between the linearization of f at $\boldsymbol{w}_t$ and the horizontal plane $z = f_\star$ (see Figure 1, more details in

Proposition 1 in the supplementary material). Note that since $f_\star$ is the minimum of f, the Polyak step-size γ_t is necessarily non-negative.

Limitations. Equation (3) has two major short-comings that prevent its applicability in a machine learning setting. First, each update requires a full evaluation of f and its derivative. Stochastic extensions have been proposed in (Nedić & Bertsekas, 2001a;b), but they still require frequent evaluations of f. This is expensive in the large data setting, and even computationally infeasible when using massive data augmentation. Second, when applying this method to the non-convex setting of deep neural networks, the method sometimes fails to converge.

Therefore we would like to design an extension of the Polyak step-size that (i) is inexpensive to compute in a stochastic setting (e.g. with a computational cost that is independent of the total number of training samples), and (ii) converges in practice when used with deep neural networks. The next section introduces the ALI-G algorithm, which achieves these two goals in the interpolation setting.

2.3. The ALI-G Algorithm

We now present the ALI-G algorithm. For this, we suppose that we are in an interpolation setting: the model is assumed to be able to drive the loss function to near zero on all samples simultaneously.

Algorithm. The main steps of the ALI-G algorithm are provided in Algorithm 1. ALI-G iterates over three operations until convergence. First, it computes a stochastic approximation of the learning objective and its derivative (line 3). Second, it computes a step-size decay parameter γ_t based on the stochastic information (line 4). Third, it updates the parameters by moving in the negative derivative direction by an amount specified by the step-size and projecting the resulting vector on to the feasible region (line 5).

Algorithm 1 *The ALI-G algorithm*

Require: maximal learning-rate η, initial feasible $\boldsymbol{w}_0 \in \Omega$, small constant $\delta > 0$

1: $t = 0$
2: **while** not converged **do**
3: Get $\ell_{z_t}(\boldsymbol{w}_t)$, $\nabla \ell_{z_t}(\boldsymbol{w}_t)$ with z_t drawn i.i.d.
4: $\gamma_t = \min\left\{ \frac{\ell_{z_t}(\boldsymbol{w}_t)}{\|\nabla \ell_{z_t}(\boldsymbol{w}_t)\|^2 + \delta}, \eta \right\}$
5: $\boldsymbol{w}_{t+1} = \Pi_\Omega\left(\boldsymbol{w}_t - \gamma_t \nabla \ell_{z_t}(\boldsymbol{w}_t)\right)$
6: $t = t + 1$
7: **end while**

Comparison with the Polyak Step-Size. There are three main differences to the update in equation (3). First, each update only uses the loss ℓ_{z_t} and its derivative rather than the full objective f and its derivative. Second, the learning-rate γ_t is clipped to η, the maximal learning-rate hyper-parameter. We emphasize that η remains constant throughout the iterations, therefore it is a single hyper-parameter and does not need a schedule like SGD learning-rate. Third, the minimum $f_\star$ has been replaced by the lower-bound of 0. All these modifications will be justified in the next section.

The ALI-G$^\infty$ Variant. When ALI-G uses no maximal learning-rate, we refer to the algorithm as ALI-G$^\infty$, since it is equivalent to use an infinite maximal learning-rate. Note that ALI-G$^\infty$ requires no hyper-parameter for its step-size.

Momentum. In some of our experiments, we accelerate ALI-G with Nesterov momentum. The update step at line 5 of algorithm 1 is then replaced by (i) a velocity update $\boldsymbol{v}_t = \mu \boldsymbol{v}_{t-1} - \gamma_t \nabla \ell_{z_t}(\boldsymbol{w}_t)$ and (ii) a parameter update $\boldsymbol{w}_{t+1} = \Pi_\Omega\left(\boldsymbol{w}_t + \mu \boldsymbol{v}_t\right)$.

3. Justification and Analysis

3.1. Stochasticity

By definition, the interpolation setting gives $f_\star = 0$, which we used in ALI-G to simplify the formula of the learning-rate γ_t. More subtly, the interpolation property also allows the updates to rely on the stochastic estimate $\ell_{z_t}(\boldsymbol{w}_t)$ rather than the exact but expensive $f(\boldsymbol{w}_t)$. Intuitively, this is possible because in the interpolation setting, we know the minimum of the loss function for each individual training sample. Recall that ALI-G$^\infty$ is the variant of ALI-G that uses no maximal learning-rate. The following result formalizes the convergence guarantee of ALI-G$^\infty$ in the stochastic convex setting.

Theorem 1 (Convex and Lipschitz)**.** *We assume that Ω is a convex set, and that for every $z \in \mathcal{Z}$, ℓ_z is convex and C-Lipschitz. Let $\mathbf{w}_\star$ be a solution of ($\mathcal{P}$), and assume that the interpolation property is approximately satisfied: $\forall z \in \mathcal{Z}$, $\ell_z(\mathbf{w}_\star) \leq \varepsilon$, for some interpolation tolerance $\varepsilon \geq 0$. Then ALI-G$^\infty$ applied to f satisfies:*

$$\mathbb{E}\left[f\left(\frac{1}{T+1} \sum_{t=0}^{T} \boldsymbol{w}_t \right) \right] \leq \frac{\|\boldsymbol{w}_0 - \mathbf{w}_\star\| \sqrt{C^2 + \delta}}{\sqrt{T+1}} + \varepsilon \sqrt{\left(\frac{C^2}{\delta} + 1 \right)}. \tag{4}$$

In other words, by assuming interpolation, ALI-G provably converges while requiring only $\ell_{z_t}(\boldsymbol{w}_t)$ and $\nabla \ell_{z_t}(\boldsymbol{w}_t)$ (stochastic estimation per sample) to compute its learning-rate. In contrast, the Polyak step-size would require $f(\boldsymbol{w}_t)$ and $\nabla f(\boldsymbol{w}_t)$ to compute the learning-rate (deterministic computation over all training samples). This is because the Polyak step-size exploits the knowledge of $f_\star$ only, which is weaker information than knowing the minimum of all

individual loss functions ℓ_z (as ALI-G does in the interpolation setting). This difference induces a major computational advantage of ALI-G over the usual Polyak step-size.

We emphasize that in Theorem 1, our careful analysis explicitly shows the dependency of the convergence result on the interpolation tolerance ε. It is reassuring to note that convergence is exact when the interpolation property is exactly satisfied ($\varepsilon = 0$).

In the supplementary material, we also establish convergence rates of $\mathcal{O}(1/T)$ for smooth convex functions, and $\mathcal{O}(\exp(-\alpha T/8\beta))$ for α-strongly convex and β-smooth functions. Similar results can be proved when using a maximal learning-rate η: the convergence speed then remains unchanged provided that η is large enough, and it is lowered when η is small. We refer the interested reader to the supplementary for the formal results and their proofs.

Interpolation and Gradient Variance. In the literature, most convergence results of SGD depend on the variance of the gradient, which we denote by υ here. The reader may have noticed that our convergence results depends only the interpolation tolerance ε rather than υ. We briefly compare how these two quantities help convergence in their own distinct ways. The gradient variance υ globally characterizes how much the gradient direction can differ across individual samples z, at any point $\boldsymbol{w}$ of the parameter space. In particular, a low value for υ implies that the loss functions ℓ_z agree in the steepest descent direction at any point of the trajectory $\boldsymbol{w}_0, ..., \boldsymbol{w}_T$. In contrast, the interpolation tolerance ε locally characterizes the behavior of all loss functions near a global minimum $\mathbf{w}_\star$ only. More specifically, a low value for ε ensures that all loss functions ℓ_z agree in a common minimizer $\mathbf{w}_\star$. Thus these two mechanisms are distinct ways of ensuring convergence of SGD. Importantly, a low interpolation tolerance ε does not necessarily imply a low gradient variance υ and vice-versa.

3.2. Maximal Learning-Rate

Non-Convexity. The Polyak step-size may fail to converge when the objective is non-convex, as figure 2 illustrates: in this (non-convex) setting, gradient descent with Polyak step-size oscillates between two symmetrical points because its step-size is too large. A similar behavior can be observed on the non-convex problem of training deep neural networks.

In order to analyze the convergence of ALI-G in a non-convex setting, we introduce the Restricted Secant Inequality (RSI) (Zhang & Yin, 2013):

Definition 1. *Let $\phi : \mathbb{R}^p \to \mathbb{R}$ be a lower-bounded differentiable function achieving its minimum at $\mathbf{w}_\star$. We say that*

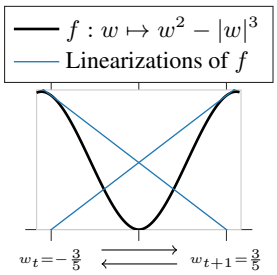

Figure 2. A simple example where the Polyak step-size oscillates due to non-convexity. On this problem, ALI-G converges whenever its maximal learning-rate is lower than 10.

ϕ satisfies the RSI if there exists $\alpha > 0$ such that:

$$\forall \boldsymbol{w} \in \mathbb{R}^p,\ \nabla\phi(\mathbf{w})^\top(\boldsymbol{w} - \mathbf{w}_\star) \geq \alpha\|\boldsymbol{w} - \mathbf{w}_\star\|^2. \quad (5)$$

The RSI does not require convexity and is a weaker assumption in the sense that all strongly convex functions satisfy the RSI (Zhang & Yin, 2013). In particular, the example in figure 2 does satisfy the RSI (proof in the supplementary material). In other words, the example above shows that the Polyak step-size can fail to converge under the RSI assumption. In contrast, we prove that with an appropriate maximal learning-rate, ALI-G converges (exponentially fast) on all interpolating problems that satisfy the RSI:

Theorem 2. *We assume that $\Omega = \mathbb{R}^p$, and that for every $z \in \mathcal{Z}$, ℓ_z is β-smooth and satisfies the RSI with constant μ. Let $\mathbf{w}_\star$ be a solution of ($\mathcal{P}$) such that $\forall z \in \mathcal{Z}$, $\ell_z(\mathbf{w}_\star) = 0$. Further assume that $\frac{1}{2\beta} \leq \eta \leq \frac{2\mu}{\beta^2}$. Then if we apply ALI-G with a maximal learning-rate of η to f, we have:*

$$f(\mathbf{w}_{T+1}) - f_\star \leq \frac{\beta}{2}\exp\left(\frac{-(2\mu - \eta\beta^2)T}{2\beta}\right)\|\mathbf{w}_0 - \mathbf{w}_\star\|^2. \quad (6)$$

Note that the above theorem assumes perfect interpolation, that is, the tolerance $\epsilon = 0$. Nonetheless, it demonstrates the importance of a maximal learning rate, which does not need a manual decaying schedule. It is currently an open question whether a similar result to theorem 2 can be proved with some interpolation tolerance $\varepsilon > 0$ on the value of all $\ell_z(\mathbf{w}_\star)$.

Proximal Interpretation. Interestingly, using a maximal learning-rate can be seen as a natural extension of SGD when using a non-negative loss function:

Proposition 1. *[Proximal Interpretation] Suppose that $\Omega = \mathbb{R}^p$ and let $\delta = 0$. We consider the update performed by SGD: $\boldsymbol{w}_{t+1}^{SGD} = \boldsymbol{w}_t - \eta_t\nabla\ell_{z_t}(\boldsymbol{w}_t)$; and the update performed by ALI-G: $\boldsymbol{w}_{t+1}^{ALI\text{-}G} = \boldsymbol{w}_t - \gamma_t\nabla\ell_{z_t}(\boldsymbol{w}_t)$, where*

$\gamma_t = \min\left\{\frac{\ell_{z_t}(\boldsymbol{w}_t)}{\|\nabla \ell_{z_t}(\boldsymbol{w}_t)\|^2}, \eta\right\}$. *Then we have:*

$$\begin{aligned}
\boldsymbol{w}_{t+1}^{SGD} &= \operatorname*{argmin}_{\boldsymbol{w}\in\mathbb{R}^p}\Big\{\tfrac{1}{2\eta_t}\|\boldsymbol{w}-\boldsymbol{w}_t\|^2 + \\
&\qquad \ell_{z_t}(\boldsymbol{w}_t) + \nabla \ell_{z_t}(\boldsymbol{w}_t)^\top(\boldsymbol{w}-\boldsymbol{w}_t)\Big\}, \\
\boldsymbol{w}_{t+1}^{ALI\text{-}G} &= \operatorname*{argmin}_{\boldsymbol{w}\in\mathbb{R}^p}\Big\{\tfrac{1}{2\eta}\|\boldsymbol{w}-\boldsymbol{w}_t\|^2 + \\
&\qquad \max\left\{\ell_{z_t}(\boldsymbol{w}_t) + \nabla \ell_{z_t}(\boldsymbol{w}_t)^\top(\boldsymbol{w}-\boldsymbol{w}_t), 0\right\}\Big\}.
\end{aligned} \tag{7}$$

In other words, at each iteration, ALI-G solves a proximal problem in closed form in a similar way to SGD. In both cases, the loss function ℓ_{z_t} is locally approximated by a first-order Taylor expansion at $\boldsymbol{w}_t$. The difference is that ALI-G also exploits the fact that ℓ_{z_t} is non-negative. This allows ALI-G to use a constant value for η in the interpolation setting, while the learning-rate η_t of SGD needs to be manually decayed.

4. Related Work

Interpolation in Deep Learning. As mentioned in the introduction, recent works have successfully exploited the interpolation assumption to prove convergence of SGD in the context of deep learning (Ma et al., 2018; Vaswani et al., 2019a; Zhou et al., 2019). Such works are complementary to ours in the sense that they provide a convergence analysis of an existing algorithm for deep learning. In a different line of work, (Liu & Belkin, 2019) propose to exploit interpolation to prove convergence of a new acceleration method for deep learning. However, their experiments suggest that the method still requires the use of a hand-designed learning-rate schedule.

Adaptive Gradient Methods. Similarly to ALI-G, most adaptive gradient methods also rely on tuning a single hyper-parameter, thereby providing a more pragmatic alternative to SGD that needs a specification of the full learning-rate schedule. While the most popular ones are Adagrad (Duchi et al., 2011), RMSPROP (Tieleman & Hinton, 2012), Adam (Kingma & Ba, 2015) and AMSGrad (Reddi et al., 2018), there have been many other variants (Zeiler, 2012; Orabona & Pál, 2015; Défossez & Bach, 2017; Levy, 2017; Mukkamala & Hein, 2017; Zheng & Kwok, 2017; Bernstein et al., 2018; Chen & Gu, 2018; Shazeer & Stern, 2018; Zaheer et al., 2018; Chen et al., 2019; Loshchilov & Hutter, 2019; Luo et al., 2019). However, as pointed out in (Wilson et al., 2017), adaptive gradient methods tend to give poor generalization in supervised learning. In our experiments, the results provided by ALI-G are significantly better than those obtained by the most popular adaptive gradient methods. Recently, Liu et al. (2019) have proposed to "rectify" Adam with a learning-rate warmup, which partly bridges the gap in generalization performance between Adam and SGD. However, their method still requires a learning-rate schedule, and thus remains difficult to tune on new tasks.

Adaptive Learning-Rate Algorithms. Vaswani et al. (2019b) show that one can use line search in a stochastic setting for interpolating models while guaranteeing convergence. This work is complementary to ours, as it provides convergence results with weaker assumptions on the loss function, but is less practically useful as it requires up to four hyper-parameters, instead of one for ALI-G. Less closely related methods, included second-order ones, adaptively compute the learning-rate without using the minimum (Schaul et al., 2013; Martens & Grosse, 2015; Tan et al., 2016; Zhang et al., 2017; Baydin et al., 2018; Wu et al., 2018; Li & Orabona, 2019; Henriques et al., 2019), but do not demonstrate competitive generalization performance against SGD with a well-tuned hand-designed schedule.

L_4 Algorithm. The L_4 algorithm (Rolinek & Martius, 2018) also uses a modified version of the Polyak step-size. However, the L_4 algorithm computes an online estimate of $f_\star$ rather than relying on a fixed value. This requires three hyper-parameters, which are in practice sensitive to noise and crucial for empirical convergence of the method. In addition, L_4 does not come with convergence guarantees. In contrast, by utilizing the interpolation property and a maximal learning-rate, our method is able to (i) provide reliable and accurate minimization with only a single hyper-parameter, and (ii) offer guarantees of convergence in the stochastic convex setting.

Frank-Wolfe Methods. The proximal interpretation in Proposition 1 allows us to draw additional parallels to existing methods. In particular, the formula of the learning-rate γ_t may remind the reader of the Frank-Wolfe algorithm (Frank & Wolfe, 1956) in some of its variants (Locatello et al., 2017), or other dual methods (Lacoste-Julien & Jaggi, 2013; Shalev-Shwartz & Zhang, 2016). This is because such methods solve in closed form the dual of problem (7), and problems in the form of (7) naturally appear in dual coordinate ascent methods (Shalev-Shwartz & Zhang, 2016).

When no regularization is used, ALI-G and Deep Frank-Wolfe (DFW) (Berrada et al., 2019) are procedurally identical algorithms. This is because in such a setting, one iteration of DFW also amounts to solving (7) in closed-form – more generally, DFW is designed to train deep neural networks by solving proximal linear support vector machine problems approximately. However, we point out the two fundamental advantages of ALI-G over DFW: (i) ALI-G can handle arbitrary (lower-bounded) loss functions, while DFW can only use convex piece-wise linear loss functions;

and (ii) as seen previously, ALI-G provides convergence guarantees in the convex setting.

SGD with Polyak's Learning-Rate. (Oberman & Prazeres, 2019) extend the Polyak step-size to rely on a stochastic estimation of the gradient $\nabla \ell_{z_t}(\boldsymbol{w}_t)$ only, instead of the expensive deterministic gradient $\nabla f(\boldsymbol{w}_t)$. However, they still require to evaluate $f(\boldsymbol{w}_t)$, the objective function over the entire training data set, in order to compute its learning-rate, which makes the method impractical. In addition, since they do not do exploit the interpolation setting nor the fact that regularization can be expressed as a constraint, they also require the knowledge of the optimal objective function value $f_\star$. We also refer the interested reader to the recent analysis of (Loizou et al., 2020), which appeared after this work and provides a set of improved theoretical results.

APROX **Algorithm.** (Asi & Duchi, 2019) have recently introduced the APROX algorithm, a family of proximal stochastic optimization algorithms for convex problems. Notably, the APROX "truncated model" version is similar to ALI-G. However, there are four clear advantages of our work over (Asi & Duchi, 2019) in the interpolation setting, in particular for training neural networks. First, our work is the first to empirically demonstrate the applicability and usefulness of the algorithm on varied modern deep learning tasks – most of our experiments use several orders of magnitude more data and model parameters than the small-scale convex problems of (Asi & Duchi, 2019). Second, our analysis and insights allow us to make more aggressive choices of learning rate than (Asi & Duchi, 2019). Indeed, (Asi & Duchi, 2019) assume that the maximal learning-rate is exponentially decaying, even in the interpolating convex setting. In contrast, by avoiding the need for an exponential decay, the learning-rate of ALI-G requires only one hyper-parameters instead of two for APROX. Third, our analysis takes into account the interpolation tolerance $\varepsilon \geq 0$ rather than unrealistically assuming the perfect case $\varepsilon = 0$ (that would require infinite weights when using the cross-entropy loss for instance). Fourth, our analysis proves fast convergence in function space rather than iterate space.

5. Experiments

We empirically compare ALI-G to the optimization algorithms most commonly used in deep learning. Our experiments span a variety of architectures and tasks: (i) learning a differentiable neural computer; (ii) training wide residual networks on SVHN; (iii) training a Bi-LSTM on the Stanford Natural Language Inference data set; and (iv) training wide residual networks and densely connected networks on the CIFAR data sets. Note that the tasks of training wide residual networks on SVHN and CIFAR-100 are part of the DeepOBS benchmark (Schneider et al., 2019), which aims at standardizing baselines for deep learning optimizers. In particular, these tasks are among the most difficult ones of the benchmark because the SGD baseline benefits from a manual schedule for the learning rate. Despite this, our set of experiments demonstrate that ALI-G obtains competitive performance with SGD. In addition, ALI-G significantly outperforms adaptive gradient methods.

The code to reproduce our results is publicly available[1]. In the TensorFlow (Abadi et al., 2015) experiment, we use the official and publicly available implementation of L_4[2]. In the PyTorch (Paszke et al., 2017) experiments, we use our implementation of L_4, which we unit-test against the official TensorFlow implementation. In addition, we employ the official implementation of DFW[3] and we re-use their code for the experiments on SNLI and CIFAR. All experiments are performed either on a 12-core CPU (differentiable neural computer), on a single GPU (SVHN, SNLI, CIFAR) or on up to 4 GPUs (ImageNet). We emphasize that all methods approximately have the same cost per iteration. Consequently, faster convergence in terms of number of iterations or epochs translates into faster convergence in terms of wall-clock time.

5.1. Differentiable Neural Computers

Setting. The Differentiable Neural Computer (DNC) (Graves et al., 2016) is a recurrent neural network that aims at performing computing tasks by learning from examples rather than by executing an explicit program. In this case, the DNC learns to repeatedly copy a fixed size string given as input. Although this learning task is relatively simple, the complex architecture of the DNC makes it an interesting benchmark problem for optimization algorithms.

Methods. We use the official and publicly available implementation of DNC[4]. We vary the initial learning rate as powers of ten between 10^{-4} and 10^4 for each method except for L_4Adam and L_4Mom. For L_4Adam and L_4Mom, since the main hyper-parameter α is designed to lie in $(0, 1)$, we vary it between 0.05 and 0.095 with a step of 0.1. The gradient norm is clipped for all methods except for ALI-G, L_4Adam and L_4Mom (as recommended by (Rolinek & Martius, 2018)).

Results. We present the results in Figure 3. ALI-G provides accurate optimization for any η within $[10^{-1}, 10^6]$, and is among the best performing methods by reaching an objective function of 4.10^{-8}. On this task, RMSProp, L_4Adam and L_4Mom also provide accurate and robust opti-

[1] https://github.com/oval-group/ali-g

[2] https://github.com/martius-lab/14-optimizer

[3] https://github.com/oval-group/dfw

[4] https://github.com/deepmind/dnc

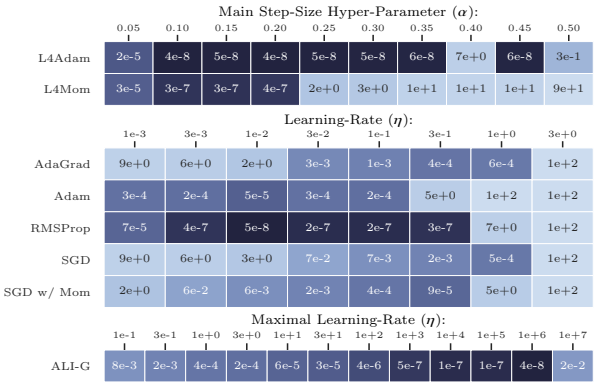

Figure 3. Final objective function when training a Differentiable Neural Computer for 10*k steps (lower is better). The intensity of each cell is log-proportional to the value of the objective function (darker is better). ALI-G obtains good performance for a very large range of* η *(*$10^{-1} \leq \eta \leq 10^{6}$*).*

mization. In contrast to ALI-G and the L_4 methods, the most commonly used algorithms such as SGD, SGD with momentum and Adam are very sensitive to their main learning-rate hyper-parameter. Note that the difference between well-performing methods is not significant here because these reach the numerical precision limit of single-precision float numbers.

5.2. Wide Residual Networks on SVHN

Setting. The SVHN data set contains 73k training samples, 26k testing samples and 531k additional easier samples. From the 73k difficult training examples, we select 6k samples for validation; we use all remaining (both difficult and easy) examples for training, for a total of 598k samples. We train a wide residual network 16-4 following (Zagoruyko & Komodakis, 2016).

Method. For SGD, we use the manual schedule for the learning rate of (Zagoruyko & Komodakis, 2016). For L_4Adam and L_4Mom, we cross-validate the main learning-rate hyper-parameter α to be in $\{0.0015, 0.015, 0.15\}$ (0.15 is the value recommended by (Rolinek & Martius, 2018)). For other methods, the learning rate hyper-parameter is tuned as a power of 10. The ℓ_2 regularization is cross-validated in $\{0.0001, 0.0005\}$ for all methods but ALI-G. For ALI-G, the regularization is expressed as a constraint on the ℓ_2-norm of the parameters, and its maximal value is set to 50. SGD, ALI-G and BPGrad use a Nesterov momentum of 0.9. All methods use a dropout rate of 0.4 and a fixed budget of 160 epochs, following (Zagoruyko & Komodakis, 2016).

Test Accuracy on SVHN (%)			
Adagrad	98.0	Adam	97.9
AMSGrad	97.9	BPGrad	98.1
DFW	98.1	L_4Adam	**98.2**
L_4Mom	19.6	ALI-G	98.1
SGD	98.3	SGD†	98.4

Table 1. In red, SGD benefits from a hand-designed schedule for its learning-rate. In black, adaptive methods, including ALI-G, have a single hyper-parameter for their learning-rate. $SGD^\dagger$ *refers to the performance reported by (Zagoruyko & Komodakis, 2016).*

Results. The results are presented in Table 1. On this relatively easy task, most methods achieve about 98% test accuracy. Despite the cross-validation, L_4Mom does not converge on this task. Even though SGD benefits from a hand-designed schedule, ALI-G and other adaptive methods obtain close performance to it.

5.3. Bi-LSTM on SNLI

Setting. We train a Bi-LSTM of 47M parameters on the Stanford Natural Language Inference (SNLI) data set (Bowman et al., 2015). The SNLI data set consists in 570k pairs of sentences, with each pair labeled as entailment, neutral or contradiction. This large scale data set is commonly used as a pre-training corpus for transfer learning to many other natural language tasks where labeled data is scarcer (Conneau et al., 2017) – much like ImageNet is used for pre-training in computer vision. We follow the protocol of (Berrada et al., 2019); we also re-use their results for the baselines.

Method. For L_4Adam and L_4Mom, the main hyper-parameter α is cross-validated in $\{0.015, 0.15\}$ – compared to the recommended value of 0.15, this helped convergence and considerably improved performance. The SGD algorithm benefits from a hand-designed schedule, where the learning-rate is decreased by 5 when the validation accuracy does not improve. Other methods use adaptive learning-rates and do not require such schedule. The value of the main hyper-parameter η is cross-validated as a power of ten for the ALI-G algorithm and for previously reported adaptive methods. Following the implementation by (Conneau et al., 2017), no ℓ_2 regularization is used. The algorithms are evaluated with the Cross-Entropy (CE) loss and the multi-class hinge loss (SVM), except for DFW which is designed for use with an SVM loss only. For all optimization algorithms, the model is trained for 20 epochs, following (Conneau et al., 2017).

Results. We present the results in Table 2. ALI-G$^\infty$ is the only method that requires no hyper-parameter for its learning-rate. Despite this, and the fact that SGD employs

Test Accuracy on SNLI (%)					
	CE	SVM		CE	SVM
Adagrad*	83.8	84.6	Adam*	84.5	85.0
AMSGrad*	84.2	85.1	BPGrad*	83.6	84.2
DFW*	-	**85.2**	L_4Adam	83.3	82.5
L_4Mom	83.7	83.2	ALI-G$^\infty$	84.6	84.7
ALI-G	**84.8**	**85.2**			
SGD*	84.7	85.2	SGD†	84.5	-

Table 2. In red, SGD benefits from a hand-designed schedule for its learning-rate. In black, adaptive methods have a single hyper-parameter for their learning-rate. In blue, ALI-G$^\infty$ does not have any hyper-parameter for its learning-rate. With an SVM loss, DFW and ALI-G are procedurally identical algorithms – but in contrast to DFW, ALI-G can also employ the CE loss. Methods in the format X^* re-use results from (Berrada et al., 2019). SGD† is the result from (Conneau et al., 2017).

a learning-rate schedule that has been hand designed for good validation performance, ALI-G$^\infty$ is still able to obtain results that are competitive with SGD. Moreover, ALI-G, which requires a single hyper-parameter for the learning-rate, outperforms all other methods for both the SVM and the CE loss functions.

5.4. Wide Residual Networks and Densely Connected Networks on CIFAR

Setting. We follow the methodology of (Berrada et al., 2019), and we reproduce their results. We test two architectures: a Wide Residual Network (WRN) 40-4 (Zagoruyko & Komodakis, 2016) and a bottleneck DenseNet (DN) 40-40 (Huang et al., 2017). We use 45k samples for training and 5k for validation. The images are centered and normalized per channel. We apply standard data augmentation with random horizontal flipping and random crops. AMSGrad was selected in (Berrada et al., 2019) because it was the best adaptive method on similar tasks, outperforming in particular Adam and Adagrad. In addition to the baselines from (Berrada et al., 2019), we also provide the performance of L_4Adam, L_4Mom, AdamW (Loshchilov & Hutter, 2019) and Yogi (Zaheer et al., 2018).

Method. All optimization methods employ the cross-entropy loss, except for the DFW algorithm, which is designed to use an SVM loss. For DN and WRN respectively, SGD uses the manual learning rate schedules from (Huang et al., 2017) and (Zagoruyko & Komodakis, 2016). Following (Berrada et al., 2019), the batch-size is cross-validated in $\{64, 128, 256\}$ for the DN architecture, and $\{128, 256, 512\}$ for the WRN architecture. For L_4Adam and L_4Mom, the learning-rate hyper-parameter α is cross-validated in $\{0.015, 0.15\}$. For AMSGrad, AdamW, Yogi, DFW and ALI-G, the learning-rate hyper-parameter η is cross-validated as a power of 10 (in practice $\eta \in \{0.1, 1\}$ for ALI-G). SGD, DFW and ALI-G use a Nesterov momentum of 0.9. Following (Berrada et al., 2019), for all methods but ALI-G and AdamW, the ℓ_2 regularization is cross-validated in $\{0.0001, 0.0005\}$ on the WRN architecture, and is set to 0.0001 for the DN architecture. For AdamW, the weight-decay is cross-validated as a power of 10. For ALI-G, ℓ_2 regularization is expressed as a constraint on the norm on the vector of parameters; its maximal value is set to 100 for the WRN models, 80 for DN on CIFAR-10 and 75 for DN on CIFAR-100. For all optimization algorithms, the WRN model is trained for 200 epochs and the DN model for 300 epochs, following respectively (Zagoruyko & Komodakis, 2016) and (Huang et al., 2017).

Results. We present the results in Table 3. In this setting again, ALI-G obtains competitive performance with manually decayed SGD. ALI-G largely outperforms AMSGrad, AdamW and Yogi.

Test Accuracy on CIFAR (%)				
	CIFAR-10		CIFAR-100	
	WRN	DN	WRN	DN
AMSGrad	90.8	91.7	68.7	69.4
AdamW	92.1	92.6	69.6	69.5
Yogi	91.2	92.1	68.7	69.6
DFW	94.2	94.6	**76.0**	73.2
L_4Adam	90.5	90.8	61.7	60.5
L_4Mom	91.6	91.9	61.4	62.6
ALI-G	**95.2**	**95.0**	75.8	**76.3**
SGD	95.3	95.1	77.8	76.3
SGD†	95.4	-	78.8	-

Table 3. In red, SGD benefits from a hand-designed schedule for its learning-rate. In black, adaptive methods, including ALI-G, have a single hyper-parameter for their learning-rate. SGD† refers to the result from (Zagoruyko & Komodakis, 2016). Each reported result is an average over three independent runs; the standard deviations are reported in Appendix (they are at most 0.3 for ALI-G and SGD).

5.5. Comparing Training Performance on CIFAR-100

In this section, we empirically assess the performance of ALI-G and its competitors in terms of training objective on CIFAR-100. In order to have comparable objective functions, the ℓ_2 regularization is deactivated. The learning-rate is selected as a power of ten for best final objective value, and the batch-size is set to its default value. For clarity, we only display the performance of SGD, Adam, Adagrad and ALI-G (DFW does not support the cross-entropy loss). The L_4 methods diverge in this setting. Here SGD uses a

constant learning-rate to emphasize the need for adaptivity. Therefore all methods use one hyper-parameter for their learning-rate. All methods use a fixed budget of 200 epochs for WRN-CIFAR-100 and 300 epochs for DN-CIFAR-100. As can be seen, ALI-G provides better training performance than the baseline algorithms on all tasks.

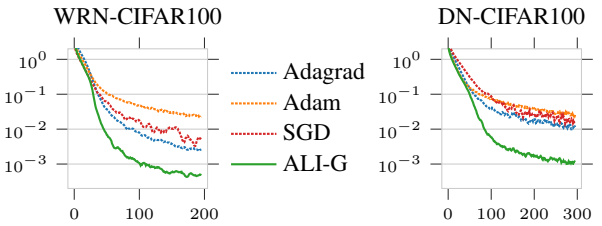

Figure 4. Objective function over the epochs on CIFAR-100 (smoothed with a moving average over 5 epochs). ALI-G reaches a value that is an order of magnitude better than the baselines.

5.6. Training at Large Scale

We demonstrate the scalability of ALI-G by training a ResNet-18 (He et al., 2016) on the ImageNet data set. In order to satisfy the interpolation assumption, we employ a loss function tailored for top-5 classification (Lapin et al., 2016), and we do not use data augmentation. Our focus here is on the training objective and accuracy. ALI-G uses the following training setup: a batch-size of 1024 split over 4 GPUs, a ℓ_2 maximal norm of 400 for $\mathbf{w}$, a maximal learning-rate of 10 and no momentum. SGD uses the state-of-the-art hyper-parameters and learning-rate schedule from (He et al., 2016). As can be seen in figure 5, ALI-G reaches 99% top-5 accuracy in 12 epochs (faster than SGD), and minimizes the objective function as well as SGD with its custom schedule.

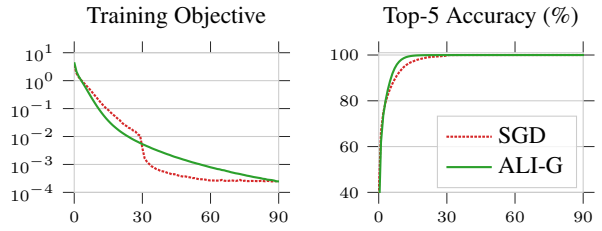

Figure 5. Training a ResNet-18 on ImageNet. The final performance of ALI-G is as good as that of SGD, even though SGD benefits from a custom learning-rate schedule. In addition, ALI-G reaches a high training accuracy faster than SGD.

6. Discussion

We have introduced ALI-G, an optimization algorithm that automatically adapts the learning-rate in the interpolation setting. ALI-G provides convergence guarantees in the stochastic setting, including for a class of non-convex problems. By using the same descent direction as SGD, it offers comparable generalization performance while requiring significantly less tuning. In future work, it would be interesting to extend ALI-G to the non-interpolating setting by adapting the minimum $f_\star$ online while requiring few hyper-parameters.

Acknowledgements

This work was supported by the EPSRC grants AIMS CDT EP/L015987/1, Seebibyte EP/M013774/1, EP/P020658/1 and TU/B/000048, and by YouGov. We also thank the Nvidia Corporation for the GPU donation.

References

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G., 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., Man, 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., Vi, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., and Zheng, X. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*, 2015. Software available from tensorflow.org.

Asi, H. and Duchi, J. C. Stochastic (approximate) proximal point methods: Convergence, optimality, and adaptivity. *SIAM Journal on Optimization*, 2019.

Baydin, A. G., Cornish, R., Rubio, D. M., Schmidt, M., and Wood, F. Online learning rate adaptation with hypergradient descent. *International Conference on Learning Representations*, 2018.

Bernstein, J., Wang, Y.-X., Azizzadenesheli, K., and Anandkumar, A. signsgd: Compressed optimisation for non-convex problems. *International Conference on Machine Learning*, 2018.

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

Bowman, S. R., Angeli, G., Potts, C., and Manning, C. D. A large annotated corpus for learning natural language inference. *Conference on Empirical Methods in Natural Language Processing*, 2015.

Brännlund, U. A generalized subgradient method with relaxation step. *Mathematical Programming*, 1995.

Chen, J. and Gu, Q. Padam: Closing the generalization gap of adaptive gradient methods in training deep neural networks. *arXiv preprint*, 2018.

Chen, X., Liu, S., Sun, R., and Hong, M. On the convergence of a class of adam-type algorithms for non-convex optimization. *International Conference on Learning Representations*, 2019.

Conneau, A., Kiela, D., Schwenk, H., Barrault, L., and Bordes, A. Supervised learning of universal sentence representations from natural language inference data. *Conference on Empirical Methods in Natural Language Processing*, 2017.

Défossez, A. and Bach, F. Adabatch: Efficient gradient aggregation rules for sequential and parallel stochastic gradient methods. *arXiv preprint*, 2017.

Duchi, J., Hazan, E., and Singer, Y. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 2011.

Frank, M. and Wolfe, P. An algorithm for quadratic programming. *Naval Research Logistics Quarterly*, 1956.

Graves, A., Wayne, G., Reynolds, M., Harley, T., Danihelka, I., Grabska-Barwińska, A., Colmenarejo, S. G., Grefenstette, E., Ramalho, T., Agapiou, J., Badia, A. P., Hermann, K. M., Zwols, Y., Ostrovski, G., Cain, A., King, H., Summerfield, C., Blunsom, P., Kavukcuoglu, K., and Hassabis, D. Hybrid computing using a neural network with dynamic external memory. *Nature*, 2016.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. *Conference on Computer Vision and Pattern Recognition*, 2016.

Henriques, J. F., Ehrhardt, S., Albanie, S., and Vedaldi, A. Small steps and giant leaps: Minimal newton solvers for deep learning. *International Conference on Computer Vision*, 2019.

Huang, G., Liu, Z., Weinberger, K. Q., and van der Maaten, L. Densely connected convolutional networks. *Conference on Computer Vision and Pattern Recognition*, 2017.

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

Lacoste-Julien, S. and Jaggi, M. Block-coordinate Frank-Wolfe optimization for structural SVMs. *International Conference on Machine Learning*, 2013.

Lapin, M., Hein, M., and Schiele, B. Loss functions for top-k error: Analysis and insights. *Conference on Computer Vision and Pattern Recognition*, 2016.

Levy, K. Online to offline conversions, universality and adaptive minibatch sizes. *Neural Information Processing Systems*, 2017.

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

Liu, C. and Belkin, M. Accelerating sgd with momentum for over-parameterized learning. *International Conference on Learning Representations*, 2019.

Liu, L., Jiang, H., He, P., Chen, W., Liu, X., Gao, J., and Han, J. On the variance of the adaptive learning rate and beyond. *arXiv preprint*, 2019.

Locatello, F., Khanna, R., Tschannen, M., and Jaggi, M. A unified optimization view on generalized matching pursuit and frank-wolfe. *International Conference on Artificial Intelligence and Statistics*, 2017.

Loizou, N., Vaswani, S., Laradji, I., and Lacoste-Julien, S. Stochastic polyak step-size for sgd: An adaptive learning rate for fast convergence. *arXiv preprint*, 2020.

Loshchilov, I. and Hutter, F. Fixing weight decay regularization in adam. *International Conference on Learning Representations*, 2019.

Luo, L., Xiong, Y., Liu, Y., and Sun, X. Adaptive gradient methods with dynamic bound of learning rate. *International Conference on Learning Representations*, 2019.

Ma, S., Bassily, R., and Belkin, M. The power of interpolation: Understanding the effectiveness of sgd in modern over-parametrized learning. *International Conference on Machine Learning*, 2018.

Martens, J. and Grosse, R. Optimizing neural networks with Kronecker-factored approximate curvature. *International Conference on Machine Learning*, 2015.

Mukkamala, M. C. and Hein, M. Variants of rmsprop and adagrad with logarithmic regret bounds. *International Conference on Machine Learning*, 2017.

Nedić, A. and Bertsekas, D. Convergence rate of incremental subgradient algorithms. *Stochastic optimization: algorithms and applications*, 2001a.

Nedić, A. and Bertsekas, D. Incremental subgradient methods for nondifferentiable optimization. *SIAM Journal on Optimization*, 2001b.

Oberman, A. M. and Prazeres, M. Stochastic gradient descent with polyak's learning rate. *arXiv preprint*, 2019.

Orabona, F. and Pál, D. Scale-free algorithms for online linear optimization. *International Conference on Algorithmic Learning Theory*, 2015.

Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., and Lerer, A. Automatic differentiation in pytorch. *NIPS Autodiff Workshop*, 2017.

Polyak, B. T. Minimization of unsmooth functionals. *USSR Computational Mathematics and Mathematical Physics*, 1969.

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

Robbins, H. and Monro, S. A stochastic approximation method. *The annals of mathematical statistics*, 1951.

Rolinek, M. and Martius, G. L4: Practical loss-based stepsize adaptation for deep learning. *Neural Information Processing Systems*, 2018.

Schaul, T., Zhang, S., and LeCun, Y. No more pesky learning rates. *International Conference on Machine Learning*, 2013.

Schneider, F., Balles, L., and Hennig, P. DeepOBS: A deep learning optimizer benchmark suite. *International Conference on Learning Representations*, 2019.

Shalev-Shwartz, S. and Zhang, T. Accelerated proximal stochastic dual coordinate ascent for regularized loss minimization. *Mathematical Programming*, 2016.

Shazeer, N. and Stern, M. Adafactor: Adaptive learning rates with sublinear memory cost. *International Conference on Machine Learning*, 2018.

Shor, N. Z. *Minimization methods for non-differentiable functions*. Springer Series in Computational Mathematics, 1985.

Tan, C., Ma, S., Dai, Y.-H., and Qian, Y. Barzilai-borwein step size for stochastic gradient descent. *Neural Information Processing Systems*, 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. *International Conference on Artificial Intelligence and Statistics*, 2019a.

Vaswani, S., Mishkin, A., Laradji, I., Schmidt, M., Gidel, G., and Lacoste-Julien, S. Painless stochastic gradient: Interpolation, line-search, and convergence rates. *arXiv preprint*, 2019b.

Wilson, A. C., Roelofs, R., Stern, M., Srebro, N., and Recht, B. The marginal value of adaptive gradient methods in machine learning. *Neural Information Processing Systems*, 2017.

Wu, X., Ward, R., and Bottou, L. WNGrad: Learn the learning rate in gradient descent. *arXiv preprint*, 2018.

Zagoruyko, S. and Komodakis, N. Wide residual networks. *British Machine Vision Conference*, 2016.

Zaheer, M., Reddi, S., Sachan, D., Kale, S., and Kumar, S. Adaptive methods for nonconvex optimization. *Neural Information Processing Systems*, 2018.

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

Zhang, H. and Yin, W. Gradient methods for convex minimization: better rates under weaker conditions. *arXiv preprint*, 2013.

Zhang, Z., Wu, Y., and Wang, G. Bpgrad: Towards global optimality in deep learning via branch and pruning. *Conference on Computer Vision and Pattern Recognition*, 2017.

Zheng, S. and Kwok, J. T. Follow the moving leader in deep learning. *International Conference on Machine Learning*, 2017.

Zhou, Y., Yang, J., Zhang, H., Liang, Y., and Tarokh, V. Sgd converges to global minimum in deep learning via star-convex path. *International Conference on Learning Representations*, 2019.