LOCALLY OPTIMAL DESCENT FOR DYNAMIC STEPSIZE SCHEDULING

Anonymous authors
Paper under double-blind review

ABSTRACT

We introduce a novel dynamic learning-rate scheduling scheme grounded in theory with the goal of simplifying the manual and time-consuming tuning of schedules in practice. Our approach is based on estimating the locally-optimal stepsize, guaranteeing maximal descent in the direction of the stochastic gradient of the current step. We first establish theoretical convergence bounds for our method within the context of smooth non-convex stochastic optimization, matching state-of-the-art bounds while only assuming knowledge of the smoothness parameter. We then present a practical implementation of our algorithm and conduct systematic experiments across diverse datasets and optimization algorithms, comparing our scheme with existing state-of-the-art learning-rate schedulers. Our findings indicate that our method needs minimal tuning when compared to existing approaches, removing the need for auxiliary manual schedules and warm-up phases and achieving comparable performance with drastically reduced parameter tuning.

1 INTRODUCTION

Stochastic gradient-based optimization methods such as SGD and Adam (Kingma & Ba, 2014) are the main workhorse behind modern machine learning. Such methods sequentially apply stochastic gradient steps to update the trained model and their performance crucially depends on the choice of a learning rate sequence, or schedule, used throughout this process to determine the magnitude of the sequential updates. All in all, effectively tuning the learning rate schedule is widely considered a tedious task requiring extensive, sometimes prohibitive, hyper-parameter search, resulting in a significant excess of engineering time and compute resources usage in ML training.

A prominent approach to address this issue gave rise to a plethora of *adaptive optimization methods* (most notably Duchi *et al.*, 2011 and Kingma & Ba, 2014), where the learning rate parameter is automatically tuned during the optimization process based on previously received stochastic gradients. In some important applications these methods provide superior convergence performance, while their theoretical guarantees match the state-of-the-art in the stochastic convex and (smooth) non-convex optimization settings (Li & Orabona, 2019; Ward *et al.*, 2020; Attia & Koren, 2023). However, despite the adaptivity incorporated into these methods, auxiliary learning rate schedules are often still required to actually attain their optimal performance (e.g., Loshchilov & Hutter, 2016), and the nuisance of laborious and extensive manual tuning still remain relevant for these methods as well. Furthermore, and perhaps more fundamentally, commonly used schedules appear rather arbitrary and manually tailored to the specific task at hand, and it is desirable to have a more principled, theory-grounded and general-purpose approach to schedule tuning.

In this paper, we introduce a novel dynamic learning-rate scheduling scheme, we call *GLyDER*,[1] with the goal of automatizing and simplifying the manual and time-consuming tuning of schedules in practice. GLyDER is principled on an update rule that uses a locally-optimal learning rate, that is, a step-size picked so as to optimize (a bound on) the achievable single-step (greedy) improvement in function value. This apparatus can be used as an add-on on top of any first-order optimization method that provides (stochastic) step directions for which locally-adapted learning rates are desired. Resulting algorithms from this scheme do not require external learning-rate schedules and are shown to achieve comparable performance, across several tasks and datasets, to state-of-the-art optimizers that rely on carefully tuned learning rate schedules.

The basic approach behind GLyDER is simple and lies on the classical "descent lemma" (e.g., Bauschke & Combettes, 2011; Beck, 2017), that quantifies a worst-case bound on the achievable single-step improvement in a given direction, which in turn dictates a choice of a learning rate that optimizes the bound. In GLyDER, the idea is to approximate this optimal learning rate given that the direction of progress is the *current stochastic gradient* (possibly mini-batched, or otherwise modified), rather than the true non-stochastic gradient as in classical uses of the descent lemma. This approximation requires methods for effectively estimating the true gradient's norm and its inner product with the stochastic gradient, based solely on additional samples of stochastic gradients. We give rigorous methods to

[1] GLyDER stands for Greedy Local Descent optimizER.

accomplish these tasks, along theoretical analysis and more practical variants used for our actual implementation of GLyDER which is used in our experiments.

To summarize, our main contributions in this paper are as follows:

- We present a new dynamic scheme for learning-rate scheduling, based on estimating the locally-optimal step-size *in the direction of the current-step stochastic gradient*; the ideas underlying this scheme and its precise derivation are described in Section 2. In contrast to common adaptive stochastic optimization methods, our scheme is able to adapt the instantaneous learning rate to the *current, local* conditions, rather than to *past* observations.
- We prove theoretical convergence bounds for our method in the smooth, non-convex stochastic optimization setting, while only requiring knowledge of the smoothness parameter. Our bounds match the state-of-the-art performance known for perfectly-tuned SGD in the smooth, non-convex setting (Ghadimi & Lan, 2013), and are discussed in Section 2.
- We propose two practical implementation variants of of the method and experiment with them extensively across several datasets and optimizers. Our findings, described in Section 4, demonstrate that our method achieves performance comparable to state-of-the-art learning rate schedulers. Notably, our method requires having to tune only a single learning rate parameter, in contrast to other schemes that often require extensive parameter tuning, manually crafted schedules and warm-up phases.

1.1 Related work

Adaptive stochastic optimization. A closely related and extremely influential line of work, with similar motivation to ours, is that on so-called adaptive optimization methods. This body of research has originated in the online learning literature and has lead to an abundance of practical and effective optimizers, some of which are used extensively in modern training pipelines (Duchi *et al.*, 2011; Kingma & Ba, 2014; Loshchilov & Hutter, 2017; Gupta *et al.*, 2018; Shazeer & Stern, 2018; Anil *et al.*, 2020; Ward *et al.*, 2020). Our study complements this family of algorithms in two important aspects: (i) our approach is based on optimizing the instantaneous learning rate to the local conditions in the *current* step, which is very different from the purpose of adaptive optimizers that adapt the learning rate based on *past* gradients; (ii) as already noted above, existing adaptive optimizers often still require auxiliary, manually-tuned learning rate schedules, and our method can be used in conjunction so as to provide them so as to provide them an appropriate schedule.

Stochastic line-search methods. Several works (Schaul *et al.*, 2013; Wu *et al.*, 2018; Rolinek & Martius, 2018; Vaswani *et al.*, 2019; Paquette & Scheinberg, 2020) consider local line search for every iteration in the stochastic setting. Most notably, Schaul *et al.* (2013) is perhaps the most closely related to our work, as they also consider a locally optimal learning rate based on a descent lemma for smooth functions. There are a few major differences: (1) Our work provide convergence analysis with exact rates that are optimal under certain assumptions, while their work only show an asymptotic convergence proof; (2) Our work propose an improved estimation of the optimal learning rate, and in Subsection 3.2 we show empirically the advantage of our estimator; and (3) We also propose a model independent method to estimate the smoothness parameter, based on Liu *et al.* (2023), while Schaul *et al.* (2013) rely on the structure of the model to estimate the smoothness. In a follow-up to Schaul *et al.* (2013), Wu *et al.* (2018) generalized their work to momentum SGD under the assumption of a convex objective. They also study cases in which the locally optimal learning rate have short-horizon bias and may not perform well on longer optimization tasks. In our work we empirically test our method on both short and long tasks (such as large text datasets) and show that it performs well across tasks.

Parameter-free optimization. There is a vast literature focusing on parameter-free algorithms, that can optimize a large class of functions with essentially no hyperparameter tuning. Many such works (Orabona & Pál, 2016; Cutkosky & Orabona, 2018; Foster *et al.*, 2017; Luo & Schapire, 2015; Jacobsen & Cutkosky, 2022) use online learning techniques to construct algorithms that achieve near-optimal convergence rate in the convex setting, these works are mostly of theoretical nature. Several works proposed more practical parameter-free algorithm with an empirical study of their methods (Orabona, 2014; Orabona & Tommasi, 2017; Kempka *et al.*, 2019; Chen *et al.*, 2022). Recently Ivgi *et al.* (2023) and Defazio & Mishchenko (2023) have proposed parameter-free algorithms which are similar in flavor and focus on estimating the distance of the weights from the solution. While Defazio & Mishchenko (2023) focus on the deterministic case, Ivgi *et al.* (2023) prove convergence rate with high probability in the stochastic case with bounded noise. We note that all the above works consider the convex setting, while our work consider smooth and not-necessarily convex objectives.

Meta-learning techniques. Several works consider different approaches to "learn the learning rate" in a dynamic method, while requiring either a few or no parameters to tune (Baydin *et al.*, 2017; Zhang *et al.*, 2019; Zhuang *et al.*, 2020). Baydin *et al.* (2017) dynamically update the learning rate by calculating the gradient w.r.t the learning rate itself. Zhang *et al.* (2019) proposes a method to "lookahead" several steps using a a different and presumably faster optimizer, and using this information to calculate an improved descent direction. Zhuang *et al.* (2020) proposes a method of predict the next gradient based on previous gradients, and take a step only if the observed gradient is close to the prediction.

2 Dynamic Step-size Based on Locally Optimal Step Size

In the following section we will first show how we derived the GLyDER stepsize scheduler which is based on finding an optimal stepsize at each iteration given a stochastic descent direction. We will next show how to efficiently estimate this optimal stepsize using several stochastic gradients, present the GLyDER stepsize algorithm and show a convergence result for it.

Problem formulation. Throughout, we assume the following optimization setup for our technical derivations and theoretical analyses. We consider an objective function $f : \mathbb{R}^d \to \mathbb{R}$ which is L-smooth, We assume that at each point $\mathbf{x}$ we have access to the gradient via a noisy oracle $\mathbf{g} = \nabla f(\mathbf{x}) + \boldsymbol{\xi}$ for some noise vector $\boldsymbol{\xi}$. Our assumptions on the noise are $\mathbb{E}[\boldsymbol{\xi} \mid \mathbf{x}] = 0$ and $\mathbb{E}\left[\|\boldsymbol{\xi}\|^2 \mid \mathbf{x}\right] \leq \sigma$, i.e., it has zero mean and bounded variance conditioned on the current iteration. This oracle is standard in many machine learning applications where at each iteration a batch of data is sampled, and the gradient of the loss is calculated with respect to this batch.

2.1 Deriving the GLyDER Stepsize

Our starting point is the following descent lemma for L-smooth functions, which is often used to give optimization guarantees (see Bauschke & Combettes (2011); Beck (2017)):

Theorem 2.1 (The descent lemma)**.** *Let $\mathbf{x}'$ be defined by $\mathbf{x}' = \mathbf{x} - \mathbf{d}$ for some arbitrary vector $\mathbf{d}$. Then for any L-smooth function f we have*

$$f(\mathbf{x}) - f(\mathbf{x}') \geq \langle \nabla f(\mathbf{x}), \mathbf{d} \rangle - \frac{L}{2}\|\mathbf{d}\|^2. \tag{1}$$

From the descent lemma we derive the optimal step size in terms of the best lower bound for the change in function value when moving in an arbitrary direction $\mathbf{d}$, by considering a learning rate that maximizes the r.h.s of Eq. (1). This provides the following corollary, where by optimal we mean that it maximizes the change in function value when only adjusting the learning rate and keeping the descent direction constant.

Corollary 2.2. *Given a starting point $\mathbf{x}$ and an arbitrary direction $\mathbf{d}$, the optimal step size in the direction $\mathbf{d}$ (according to the bound in Thm. 2.1) is: $\mathbf{x}' = \mathbf{x} - \frac{\langle \nabla f(\mathbf{x}), \mathbf{d} \rangle}{L\|\mathbf{d}\|^2}\mathbf{d}$, and the change in the function value is lower bounded by: $f(\mathbf{x}) - f(\mathbf{x}') \geq \frac{\langle \nabla f(\mathbf{x}), \mathbf{d} \rangle^2}{2L\|\mathbf{d}\|^2}$.*

Consider running SGD for t iterations, and at each iteration sampling a stochastic gradient $\mathbf{g}_t$. Corollary 2.2 states that given $\mathbf{g}_t$, the optimal learning rate for this descent direction would be

$$\eta_t = \frac{\langle \nabla f(\mathbf{x}_t), \mathbf{g}_t \rangle}{L\|\mathbf{g}_t\|^2}. \tag{2}$$

Thus, given a sequence of stochastic gradients $\mathbf{g}_1, \ldots, \mathbf{g}_T$, we can define a sequence of learning rates $\eta_1, \ldots, \eta_T$ which are guaranteed to be locally optimal in the sense that at each step $t \in [T]$ we choose the optimal learning rate when taking a step in the direction $\mathbf{g}_t$. Also, note that in the noiseless case, i.e. when $\mathbf{g}_t = \nabla f(\mathbf{x}_t)$, then the learning rate is constant and equal to $\frac{1}{L}$ which is known to give the optimal convergence rate for gradient descent using the descent lemma (see e.g. Bubeck *et al.* (2015)).

Our goal is to find a stationary point of f, that is, given $\epsilon > 0$ finding $\mathbf{x} \in \mathbb{R}^d$ with $\|\nabla f(\mathbf{x})\|^2 \leq \epsilon$. Note that it is not possible to provide stronger guarantees such as converging to a global or local minimum without further assumptions on the function (such as convexity, PL condition Karimi *et al.* (2016) etc.) In the following theorem we analyze the convergence rate of the learning rate scheduler from Eq. (2):

Theorem 2.3. *Let $f : \mathbb{R}^d \to \mathbb{R}$ be an L-smooth function. Suppose we run SGD for T iterations in the following way: We start at some arbitrary point $\mathbf{x}_0$ with $|f(\mathbf{x}_0) - f(\mathbf{x}^*)| \leq R$ where $\mathbf{x}^*$ is a global minimum of f. At each iteration $t \in [T]$*

we sample $\mathbf{g}_t = \nabla f(\mathbf{x}_t) + \boldsymbol{\xi}_t$ *where* $\mathbb{E}[\boldsymbol{\xi}_t] = 0$ *and* $\mathbb{E}\left[\|\boldsymbol{\xi}_t\|^2\right] \leq \sigma^2$ *for* $\sigma > 0$. *We define* $\mathbf{x}_t = \mathbf{x}_{t-1} - \frac{1}{L} \cdot \frac{\langle \nabla f(\mathbf{x}_t), \mathbf{g}_t \rangle}{\|\mathbf{g}_t\|^2} \mathbf{g}_t$, *then:*

$$\min_{t=1,\ldots,T} \mathbb{E}\left[\|\nabla f(\mathbf{x}_t)\|^2\right] \leq \frac{2LR}{T} + \sigma\sqrt{\frac{2LR}{T}}$$

The full proof can be found in Appendix A.1. The intuition behind the proof is pretty straightforward and requires iterative use of the descent lemma. The theorem shows that our learning rate scheduler have a couple of very useful properties:

- The convergence rate to reach a gradient of squared norm smaller than ϵ is $O\left(1/\epsilon^2\right)$. Due to previous lower bounds Drori & Shamir (2020), this is known to be the optimal convergence rate in a noisy regime for smooth functions that are not necessarily convex.
- The learning rate doesn't require knowledge of either σ or R, and is adaptive to both parameters.
- The convergence rate interpolates between the noisy regime ($\sigma > 0$) and the noiseless regime ($\sigma = 0$).

To practically implement the learning rate described in Thm. 2.3 we need to approximate the inner product $\langle \nabla f(\mathbf{x}_t), \mathbf{g}_t \rangle$ at each iteration. This approximation can be done by sampling a fresh mini-batch of stochastic gradients $\mathbf{h}_t^1, \ldots, \mathbf{h}_t^n$ at each iteration and estimating the inner product:

$$\langle \nabla f(\mathbf{x}_t), \mathbf{g}_t \rangle \approx \left\langle \frac{1}{n} \sum_{i=1}^{n} \mathbf{h}_i^t, \mathbf{g}_t \right\rangle .$$

However, this approximation has one main caveat: We cannot bound the term $\langle \nabla f(\mathbf{x}_t), \mathbf{g}_t \rangle$ away from zero, hence it is not possible to use concentration bounds such as Hoefdding's inequality to achieve a good relative approximation where n is constant for all iterations that depends only on the problem's parameter. To overcome this caveat, in the next subsection we provide a different analysis which estimate the squared norm of the gradient, instead of its inner product with the stochastic gradient.

2.2 Approximation of the GLyDER Stepsize

We begin this section with finding the locally optimal learning rate in expectation, which relies on a descent lemma in expectation. Recall that we consider an L-smooth function $f(\mathbf{x})$, and we are given at $\mathbf{x}$ a stochastic gradient $\mathbf{g} := \nabla f(\mathbf{x}) + \boldsymbol{\xi}$ where the noise is with zero mean, and $\mathbb{E}[\|\boldsymbol{\xi}\|^2 \mid \mathbf{x}] \leq \sigma^2$. Applying the smoothness condition for a descent direction $\eta(\nabla f(\mathbf{x}) + \boldsymbol{\xi})$ in expectation over the noise $\boldsymbol{\xi}$ we get:

$$\begin{aligned}\mathbb{E}\left[f(\mathbf{x} - \eta(\nabla f(\mathbf{x}) + \boldsymbol{\xi})) - f(\mathbf{x}) \mid \mathbf{x}\right] &\leq \mathbb{E}\left[-\eta\langle \nabla f(\mathbf{x}), \nabla f(\mathbf{x}) + \boldsymbol{\xi}\rangle + \eta^2 L \frac{\|\nabla f(\mathbf{x}) + \boldsymbol{\xi}\|^2}{2} \mid \mathbf{x}\right] \\ &= -\eta\|\nabla f(\mathbf{x})\|^2 + \eta^2 L \frac{\|\nabla f(\mathbf{x})\|^2 + \sigma^2}{2}\end{aligned}$$

minimizing the r.h.s over η yields that the optimal descent rate in expectation is $\frac{1}{L} \cdot \frac{\|\nabla f(\mathbf{x})\|^2}{\|\nabla f(\mathbf{x})\|^2 + \sigma^2}$. As in the previous section, we can show that this learning rate scheduler achieves the optimal convergence rate:

Theorem 2.4. *Let* $f : \mathbb{R}^d \to \mathbb{R}$ *be an* L*-smooth function. Suppose we run SGD for* T *iterations in the following way: We start at some arbitrary point* $\mathbf{x}_0$ *with* $|f(\mathbf{x}_0) - f(\mathbf{x}^*)| \leq R$ *where* $\mathbf{x}^*$ *is a global minimum of* f. *At each iteration* $t \in [T]$ *we sample* $\mathbf{g}_t = \nabla f(\mathbf{x}_t) + \boldsymbol{\xi}_t$ *where* $\mathbb{E}[\boldsymbol{\xi}_t] = 0$ *and* $\mathbb{E}\left[\|\boldsymbol{\xi}_t\|^2\right] \leq \sigma^2$ *for* $\sigma > 0$. *We define* $\mathbf{x}_t = \mathbf{x}_{t-1} - \frac{1}{L} \cdot \frac{\|\nabla f(\mathbf{x}_t)\|^2}{\|\nabla f(\mathbf{x}_t)\|^2 + \sigma^2} \mathbf{g}_t$, *then:*

$$\min_{t=1,\ldots,T} \mathbb{E}\left[\|\nabla f(\mathbf{x}_t)\|^2\right] \leq \frac{2LR}{T} + \sigma\sqrt{\frac{2LR}{T}} .$$

The full proof can be found in Appendix A.2. Note that contrary to Thm. 2.3, here we do need to know σ to define the learning rate, although we still don't need to know R. Note that also here, if we consider the noiseless case, then we get a constant learning rate of $\frac{1}{L}$.

The advantage of defining the locally optimal learning rate in this way, is that it can be efficiently estimated by sampling new gradients. Namely, by sampling stochastic gradients we can derive an unbiased estimator of the squared norm of the gradient. This is described by the following procedure:

Suppose that we sample i.i.d stochastic gradients $\mathbf{h}_t^1, \ldots, \mathbf{h}_t^n$ at $\mathbf{x}_t$, where $\mathbf{h}_t^i = \nabla f(\mathbf{x}_t) + \boldsymbol{\xi}_t^i$ with $\mathbb{E}[\boldsymbol{\xi}_i^t] = 0$ and bounded variance. Then $\frac{1}{n(n-1)} \sum_{i \neq j} \langle \mathbf{h}_t^i, \mathbf{h}_t^j \rangle$ is an unbiased estimator of the norm of the gradient. This is because:

$$\begin{aligned}
\mathbb{E}\left[\frac{1}{n(n-1)} \sum_{i \neq j} \langle \mathbf{h}_t^i, \mathbf{h}_t^j \rangle\right] &= \frac{1}{n(n-1)} \mathbb{E}\left[\sum_{i \neq j} \langle \nabla f(\mathbf{x}_t) + \boldsymbol{\xi}_t^i, \nabla f(\mathbf{x}_t) + \boldsymbol{\xi}_t^j \rangle\right] \\
&= \frac{1}{n(n-1)} \mathbb{E}\left[\sum_{i \neq j} \langle \nabla f(\mathbf{x}_t), \nabla f(\mathbf{x}_t) \rangle + \langle \nabla f(\mathbf{x}_t), \boldsymbol{\xi}_t^i \rangle + \langle \nabla f(\mathbf{x}_t), \boldsymbol{\xi}_t^j \rangle + \langle \boldsymbol{\xi}_t^i, \boldsymbol{\xi}_t^j \rangle\right] \\
&= \frac{1}{n(n-1)} \sum_{i \neq j} \langle \nabla f(\mathbf{x}_t), \nabla f(\mathbf{x}_t) \rangle = \|\nabla f(\mathbf{x}_t)\|^2 , \qquad (3)
\end{aligned}$$

where we used that the noise is sampled i.i.d with zero mean. Note that to estimate the denominator of the learning rate, we can either estimate $\|\nabla f(\mathbf{x}_t)\|^2$ and σ^2 separately, or we could just use $\frac{1}{n^2} \sum_{i,j=1}^n \langle \mathbf{h}_i^t, \mathbf{h}_j^t \rangle$ which is an unbiased estimator of $\|\nabla f(\mathbf{x}_t)\|^2 + \sigma^2 / n$.

One of the advantages of this estimator is that it aligns with the goal of the algorithm, to minimize the squared norm of the gradient. In more details, suppose we want to achieve a squared gradient norm smaller than some $\epsilon > 0$, then either the norm of the gradient is larger than ϵ, and we can estimate it using a mini-batch of size $O\left(1/\epsilon^2\right)$, or it is smaller than ϵ, which means our algorithm converged to an ϵ-stationary point.

Since this estimation reduces the variance of the noise by a factor of n (i.e. σ^2/n instead of σ^2), it is natural to also sample a mini-batch of size n to determine the descent direction and reduce the variance by the same factor. The reason for sampling two different sets of stochastic gradients is only for theoretical reasons, to avoid the dependence between the learning rate calculation and the descent direction. In our experimental results we used the same set of gradients for both tasks.

2.3 The GLyDER stepsize Algorithm

We now present in algorithm 1 the GLyDER stepsize algorithm and provide convergence guarantees for it. The input for the algorithm is a starting point $\mathbf{x}_0$, the smoothness of the objective function L and the number of stochastic gradient n that are sampled at each iteration. The algorithm at each iteration calculates unbiased estimators $\mu_t \sim \|\nabla f(\mathbf{x}_t)\|^2$ and $\gamma_t \sim \|\nabla f(\mathbf{x}_t)\|^2 + \sigma^2/n$, where $\nabla f(\mathbf{x})$ is the gradient (i.e. non-stochastic) of f at $\mathbf{x}$. Finally, the algorithm performs an update in the direction of the sum of the stochastic gradients, and with a step size $\frac{1}{L} \cdot \frac{\mu_t}{\gamma_t}$.

Algorithm 1: Theoretical GLyDER learning rate

Input: $\mathbf{x}_0$, n, L
for $t = 1, 2, \ldots, T$ **do**
  Sample stochastic gradients $\mathbf{g}_t^1, \ldots, \mathbf{g}_t^n, \mathbf{h}_t^1, \ldots, \mathbf{h}_t^n$
  Set:
    $\mu_t := \frac{1}{n(n-1)} \sum_{i \neq j} \langle \mathbf{h}_t^i, \mathbf{h}_t^j \rangle$
    $\gamma_t := \frac{1}{n^2} \sum_{i,j=1}^m \langle \mathbf{h}_t^i, \mathbf{h}_t^j \rangle$
    $\mathbf{g}_t = \sum_{i=1}^n \mathbf{g}_t^i$
  Update $\mathbf{x}_t = \mathbf{x}_{t-1} - \frac{1}{L} \cdot \frac{\mu_t}{\gamma_t} \mathbf{g}_t$

The following theorem shows the convergence of algorithm 1:

Theorem 2.5. *Let $f : \mathbb{R}^d \to \mathbb{R}$ be an L-smooth and M-Lipschitz function and let $\epsilon > 0$. Suppose $\mathbf{x}_0$ is such that $|f(\mathbf{x}_0) - f(\mathbf{x}^*)| \leq R$ where $\mathbf{x}^*$ is a global minimum of f. Also assume that the norm of noise vectors are globally bounded by $G > 0$. Suppose that we run algorithm 1 with $n \geq 20\sigma^2/\epsilon$, then there exists a universal constant $c > 0$ such that w.p $> 1 - dT \exp\left(-\frac{\epsilon^2 nc}{\sigma^2 M^2 G^2}\right)$, we get:*

$$\min_{t=1,\ldots,T} \mathbb{E}\left[\|\nabla f(\mathbf{x}_t)\|^2\right] \leq \frac{2LR}{T} + \frac{\sigma}{\sqrt{n}} \sqrt{\frac{2LR}{T}} ,$$

where the high probability is w.r.t the noise of the stochastic gradients $\mathbf{h}_t^i$'s, and the expectations is w.r.t the stochastic gradients $\mathbf{g}_t^i$'s. In particular, by choosing $n = \Omega\left(\sigma^2 M^2 G^2 \log(dT)/\epsilon^2\right)$, then w.p $> \Omega(1)$ the oracle complexity to achieve $\min_{t=1,\ldots,T} \mathbb{E}\left[\|\nabla f(\mathbf{x}_t)\|^2\right] \leq \epsilon$ is $O\left(\sigma^2 M^2 G^2 L^2 R^2 \log(dT)/\epsilon^3\right)$.

The full proof can be found in Appendix A.3. The reason that we use two sets of stochastic gradients is to avoid the dependence for the theoretical analysis. In Section 2.2 we provide the practical algorithm which uses the same set of gradients for both calculations, and our experiments are done on this practical algorithm. Also note that the algorithm converges to a gradient with squared norm smaller than ϵ in $O\left(1/\epsilon^3\right)$ oracle calls. This is not the optimal convergence rate which we achieved in Thm. 2.3 and Thm. 2.4. The reason is that here we don't assume to known σ, and estimating it up to an error of $O(\epsilon)$ requires $\Omega\left(1/\epsilon^2\right)$ oracle calls. We are not aware of another method which achieves the optimal learning rate without knowledge of σ, and with only logarithmic dependence on the input dimension.

3 Practical Implementation

In the previous section we provided theoretical justification for GLyDER. In this section we discuss several practical considerations and heuristics which improve its empirical performance.

3.1 Smoothness Estimation

Our learning rate scheduler requires knowing the smoothness of the function. In practice, not only that in general there is no prior knowledge of this term, the smoothness may also not be bounded. This is in fact the case for modern neural network which are known to be non-smooth and non-convex. One naive method to approximate the smoothness is to assume it is globally bounded, and to run a hyperparameter sweep with a goal of finding an upper bound which will work for the entire optimization process.

A different approach is to use an adaptive method to approximate the smoothness at every iteration. Here we provide two methods for such an approximation, which are inspired by similar methods from Liu *et al.* (2023). For this subsection we assume a standard supervised setting: We have a loss function $\ell(\cdot,\cdot)$, a model $N(\boldsymbol{\theta},\mathbf{x})$ (e.g a neural network) parameterized by $\boldsymbol{\theta}$ with input data $\mathbf{x}$. For a batch of samples $(\mathbf{x}_1,y_1),\ldots,(\mathbf{x}_m,y_m)$ and parameters $\boldsymbol{\theta}$ we can define the empirical loss as $f(\boldsymbol{\theta}) := \frac{1}{m}\sum_{i=1}^{m}\ell_{\text{ce}}(N(\boldsymbol{\theta},\mathbf{x}_i),y_i)$[2]. In this setting sampling a stochastic gradient means sampling a batch of inputs.

Option 1: projection to a 1-dimensional function (1-d projection. For a function multivariate $f:\mathbb{R}^d\to\mathbb{R}$, if we pick at some point $\boldsymbol{\theta}_t$ a descent direction $\mathbf{g}_t$ (i.e. by sampling a batch of inputs), finding the best learning η_t for this direction boils down to a one-dimensional function. Thus, the smoothness of our function is defined by the second derivative of this 1-dimensional function. That is, we can calculate: $L_t := \mathbf{g}_t^\top\nabla^2 f(\boldsymbol{\theta}_t)\mathbf{g}_t \in \mathbb{R}$ as the smoothness bound for iteration t. The second derivative is calculated w.r.t the same batch of inputs that was used to calculate the gradient. Note that although this term contains the Hessian of f, it requires only calculating the projection of the Hessian in the direction $\mathbf{g}_t$. This projection can be calculated in time $O(d)$, which is similar (up to constant factors) to the computational cost of calculating the gradient. In theory, the smoothness of the function could significantly change across this line, and line search could be used to gain a better bound. For practical considerations we do not perform line search, and take L_t as the local smoothness bound.

Option 2: A simplified version of the Gauss-Newton-Bartlett (GNB) estimator. In Liu *et al.* (2023) (Section 2.3, Option 2) the authors consider a multi-class classification problem. They consider the cross-entropy loss, where N is some model (e.g. a neural network) parameterized by $\boldsymbol{\theta}$ with input data $\mathbf{x}$, and its output $N(\boldsymbol{\theta},\mathbf{x})\in\mathbb{R}^V$ are the logits, where V is the number of possible classes, and $y\in\{1,\ldots,V\}$ is the output class. This represents the loss function for a multi-class classification problem with V classes on a single sample.

For a batch of samples $(\mathbf{x}_1,y_1),\ldots,(\mathbf{x}_m,y_m)$ and parameters $\boldsymbol{\theta}$ we can define the empirical loss as $L(\boldsymbol{\theta}) := \frac{1}{m}\sum_{i=1}^{m}\ell_{\text{ce}}(N(\boldsymbol{\theta},\mathbf{x}_i),y_i)$. We consider the estimator: $L_t := \max\{\nabla_{\boldsymbol{\theta}}L(\boldsymbol{\theta})\odot\nabla_{\boldsymbol{\theta}}L(\boldsymbol{\theta})\}$, where $\odot$ is the Hadamard-product, and the max function is over the coordinates [3].

We note that both options are easy to calculate. The advantage of the first method is that the reasoning behind it is clearer and more straightforward, although it does require calculating a projected second derivative, which takes approximately the same time as calculating the gradient. The second option is faster to calculate, as it requires almost no extra calculations, but is viable in theory only for the cross-entropy loss. This method can be considered as an easily calculated heuristic.

3.2 Exponential Averaging and Norm Estimation

Our learning rate scheduler depends on estimating the norm of the real gradient using stochastic gradients. By only assuming that the noise of the stochastic gradients are independent and with zero mean, we constructed an unbiased estimator in Eq. (3). But having an unbiased estimator is usually not good enough, we would also like our estimator to have low-variance. In the following theorem we calculate the variance of this estimator in the general case and under the an additional assumption that the noise is Gaussian.

[2]Only in this subsection, we slightly abuse notation and consider the parameters of the objective function as $\boldsymbol{\theta}$ instead of $\mathbf{x}$, since

[3]We note that in Liu *et al.* (2023), the authors consider a more complicated estimator. Namely, given $t=N(\boldsymbol{\theta},\mathbf{x}_i)\in\mathbb{R}^V$ they consider the categorical distribution $\text{Cat}(t)$ (a categorical distribution over the vector of logits t), they sample a label $\hat{y}_i$ from this distribution and calculate the empirical loss using these labels. We use the simpler estimator, which is also easier to calculate, without sampling labels from the distribution $\text{Cat}(t)$. In our empirical evaluation this estimator seems to work well in most tasks, the reason might be that our goal is much simpler – finding the largest value in the diagonal of the Hessian, instead of estimating the entire diagonal in Liu *et al.* (2023)

Theorem 3.1. *Let $n, d \in \mathbb{N}$ with $n, d > 1$. Assume for each $i \in [n]$ we are given independent noisy estimate of the gradient, That is, $\mathbf{g}_i = \nabla + \boldsymbol{\xi}_i$ for some $\nabla \in \mathbb{R}^d$, where $\mathbb{E}[\boldsymbol{\xi}_i] = 0$, $\mathbb{E}[\|\boldsymbol{\xi}_i\|^2] = \sigma^2$, and $\mathbf{g}_i, \nabla, \boldsymbol{\xi}_i \in \mathbb{R}^d$. Define $\mu = \frac{1}{n(n-1)} \sum_{i \neq j} \langle \mathbf{g}_i, \mathbf{g}_j \rangle$, then we have $\mathbb{E}[\mu] = \|\nabla\|^2$ and $Var(\mu) \leq \frac{4\|\nabla\|^2\sigma^2}{n} + \frac{\sigma^4}{n(n-1)}$. In particular, for $\boldsymbol{\xi}_i \sim \mathcal{N}\left(0, \frac{\sigma^2}{d} I_d\right)$ we have $Var(\mu) = \frac{4\|\nabla\|\sigma^2}{nd} + \frac{\sigma^4}{dn(n-1)}$.*

The proof can be found in Appendix B. Note that without further assumptions on the noise we can only upper bound the variance, as it will depend on the inner products between the noise vectors $\boldsymbol{\xi}_i$'s, and on the inner product between the noise vectors, and δ (which represent the gradient). For a Gaussian distribution, which is spherically symmetric and each coordinate is distributed i.i.d these inner products can be calculated and we can give an exact expression of the variance. This expression is smaller than our upper bound by a factor of d (the input dimension).

It is interesting to note that the variance exhibits two phases during training depending on the norm of the gradient: (1) When $\|\nabla f(\mathbf{x})\|^2$ is large, the variance decreases by a factor of $O\left(1/n\right)$; (2) When $\|\nabla f(\mathbf{x})\|^2$ is very close to zero, the variance decreases by a larger factor of $O\left(1/n^2\right)$, but the effect of the noise σ is also larger. The variance reduction in the second phase is achieved thanks to the fact that although we only receive n stochastic gradients, using all the inner products yields $O(n^2)$ (unbiased) estimators of the norm. Although these estimators are not independent, they still allow for variance reduction, at least when the norm is small.

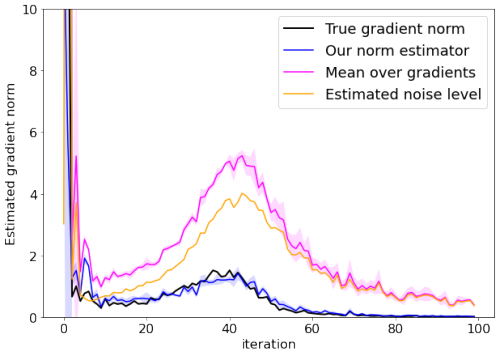


Figure 1: Comparison of our norm estimator (blue) with the norm over the mean of the stochastic gradients (magenta), and the full gradient (black). The estimated noise level (orange) is the subtraction between the two.

If figure 1 we plot our norm estimator, compared to the norm of the mean over the gradients (which is used as the norm estimator in Schaul *et al.* (2013)). We also plot the true gradient, i.e. the gradient w.r.t the entire dataset. Note that our estimator closely follow the real gradient, while using the mean estimator is biased. The experiment is done on the CIFAR100 dataset, where each stochastic gradient is drawn independently 5 times, showing the standard deviation of the estimators. For full experimental details see Appendix D.

Our bounds show that the noise has a very significant effect on the performance of our estimator. A method to reduce the variance which is commonly used in modern optimization algorithms (e.g. Adam Kingma & Ba (2014)) is to add exponential averaging. Namely, consider the newly estimated learning rate η_t at iteration t which is calculated in Thm. 2.5. For $\beta \in (0, 1)$ use exponential averaging to get the learning rates: $\hat{\eta}_t = (1 - \beta)\eta_t + \beta\hat{\eta}_{t-1}$, where $\hat{\eta}_t$ is the learning rate after exponential averaging. In all of our experiments, we used $\beta = 0.999$ as the coefficient for exponential averaging, which achieved good performance throughout all the tested datasets, algorithms and models. We note that also other optimization techniques uses exponential averaging (e.g Adam Kingma & Ba (2014)), with a constant coefficient that achieves good performance across different tasks, thus although there is an added parameter, in practice we do not tune it in any of our experiments.

3.3 A Practical Implementation of GLyDER

To summarize all the practical considerations discussed in this section, in algorithm 2 we present the full version of our learning rate scheduler. Note that we use the same set of stochastic gradients to calculate both the descent direction and learning rate at each iteration, in contrary to the more theoretical algorithm 1.

There are two more minor considerations which we take into account when implementing our learning rate scheduler, which we discuss more extensively in Appendix C. In a nutshell: (1) We use an efficient implementation of the unbiased norm estimator, requiring only $O(n)$ operations instead of $O(n^2)$ when considering all the inner products. (2) We describe how to use parallel computational processors such as TPUs (Jouppi *et al.*, 2017) to simulate several stochastic gradients without the need of re-sampling.

Finally, we can use GLyDER as a stepsize scheduler wrapper around any optimization algorithm, beyond vanilla SGD. We provide the full algorithm and additional details in Appendix C.1. In short, the only change is from the original algorith is when estimating the smoothness using option 1 from Subsection 3.1 we project onto the descent direction provided by the optimizer, instead of onto the direction if the gradient.

4 EXPERIMENTS

Algorithm 2: Practical GLyDER learning rate scheduler

Input: $\mathbf{x}_0,\ n,\ \eta_0, \beta$
for $t = 1, 2, \ldots, T$ **do**
  Sample stochastic gradients $\mathbf{g}_t^1, \ldots, \mathbf{g}_t^n$
  Set:
    $\mu_t := \left\|\sum_{i=1}^n \mathbf{g}_t^i\right\|^2 - \sum_{i=1}^n \left\|\mathbf{g}_t^i\right\|^2$
    $\gamma_t := \left\|\sum_{i=1}^n \mathbf{g}_t^i\right\|^2$
    $\mathbf{g}_t = \sum_{i=1}^n \mathbf{g}_t^i$
  if $\gamma_t = 0$ *or* $\mu_t \leq 0$ **then**
    Set $\frac{\mu_t}{\gamma_t} := 0$
  Estimate L_t using either option 1 or 2 from Subsection 3.1
  Set: $\eta_t := (1-\beta)\eta_{t-1} + \beta \cdot \frac{1}{L_t} \cdot \frac{\mu_t}{\gamma_t}$
  Update $\mathbf{x}_t = \mathbf{x}_{t-1} - \eta_t \mathbf{g}_t$

We compared our "GLyDER" stepsize scheduler to several standard manually tuned schedulers on a range of different machine learning tasks. In our experiments we varied both the models and the training algorithms to illustrate the effectiveness of our scheduler across many different scenarios. Our experiments are done using the init2winit framework (Gilmer *et al.*, 2021) which is based on JAX (Bradbury *et al.*, 2018).

4.1 EXPERIMENTAL DETAILS

Methodology and training algorithms. We compared our GLyDER stepsize scheduler to three standard schedulers: constant, cosine and reversed "squashed" square root (rsqrt) which is defined as $\eta_t = \eta_0 \cdot \frac{\sqrt{s}}{\sqrt{t+s}}$ where η_0 is the initial learning rate and s is the number of "squash" steps. We also compared our scheduler using three different training algorithms: SGD, momentum SGD and Adam (Kingma & Ba, 2014).

On each scheduler, dataset and training algorithm we performed a hyper-parameter search over a large parameter space. After choosing the best-performing hyperparameters we conducted five new experiments with different random seeds, and report the mean and standard deviation of the results. For the full hyper-parameter range of the scheduler and optimizers see Appendix D. Training was done using a TPU Jouppi *et al.* (2017) containing 8 chips. In practice, for the GLyDER scheduler it means that at algorithm 2 we received at each iteration $n = 8$ stochastic gradients which were used to estimate the GLyDER stepsize, while those gradients were also used to calculate the descent direction.

Datasets. We conducted the experiments on datasets from the fields of vision, NLP and recommendation systems:

1. CIFAR10 and CIFAR100 (Krizhevsky *et al.*, 2009), evaluated by error percentage on the test set.
2. Imagenet (Russakovsky *et al.*, 2015), evaluated by test error on the test set. We trained on the entire dataset (containing $\sim 1M$ samples), and all the images are scaled to size 224×224.
3. WikiText-2 (Merity *et al.*, 2016) which contains over $2M$ words extracted from Wikipedia, and is a standard language modeling benchmark. We evaluated on a left out validation set, and report the validation error, as well as the perplexity (in the appendix).
4. Criteo 1TB[4] which contains feature values and click feedback for millions of display ads. This is a recommendation task for click-through rate prediction. We report the area under the curve (AUC) metric.

For each dataset we used the default model from the init2winit framework. Notably, we compared the GLyDER scheduler across different models solving different types of tasks, including: Wide-ResNet (Zagoruyko & Komodakis, 2016), ResNet50 (He *et al.*, 2016), DLRM (Naumov *et al.*, 2019) and LSTM (Wiseman & Rush, 2016). For full details about the trained models see Appendix D.

4.2 RESULTS

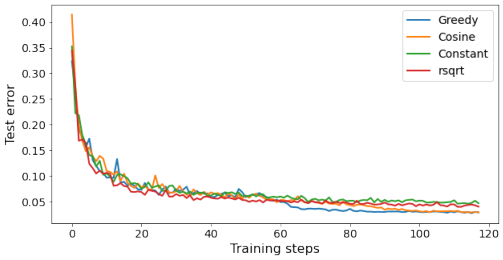
In Table 1 we compare the performance of the GLyDER scheduler using both options from Subsection 3.1 to estimate the smoothness, i.e. either using the smoothness of the projection to a 1-dimensional function, or the variation of the Gauss-Newton-Bartlett (GNB) estimator. All the experiments in Table 1 are done with the momentum SGD optimizer. From the results it can be seen that the GLyDER stepsize scheduler, with either one of the smoothness estimator, matches the best performing tuned scheduler up to an error of less than 1% across all the datasets. We performed similar experiments on vanilla SGD and Adam, showing that the GLyDER stepsize also generalize across different algorithms, for the full results see Appendix E.

In figure 2 we compare the GLyDER scheduler with 1-dimensional projection to the other schedulers for the CIFAR10 and Imagenet datasets. The figure depicts an interesting advantage of the GLyDER stepsize, although it achieves similar

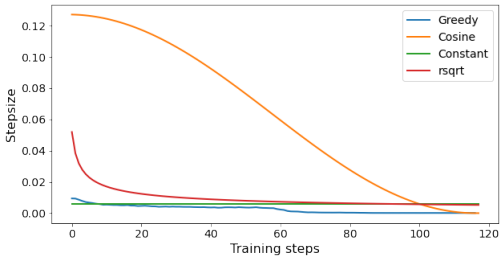
[4]https://www.kaggle.com/c/criteo-display-ad-challenge

	GLyDER + 1– d proj	**GLyDER + GNB**	**Constant**	**Cosine**	**rsqrt**
CIFAR10 ↓	3.0% ± 0.1	4.6% ± 0.2	4.7% ± 0.07	3.0% ± 0.1	4.3% ± 0.1
CIFAR100 ↓	19.7% ± 0.3	22.8% ± 1.3	22.6% ± 0.8	19.2% ± 0.2	21.3% ± 0.7
Imagenet ↓	24.6% ± 0.1	25.6% ± 1.1	32.3 ± 0.1	23.7% ± 0.08	28.7% ± 0.1
WikiText-2 ↓	78.3% ± 1.2	76.7% ± 0.03	75.8% ± 0.0	75.9% ± 0.04	75.9% ± 0.0
Criteo ↑	0.78 ± 0.003	0.69 ± 0.006	0.78 ± 0.001	0.75 ± 0.004	0.7 ± 0.003

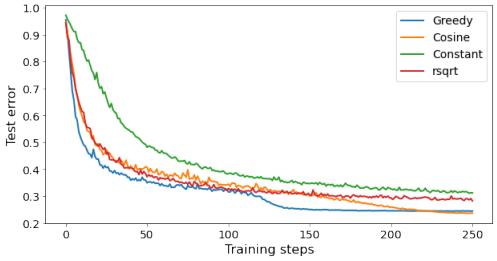
Table 1: Comparison of the GLyDER stepsize scheduler with the two options to estimate the smoothness trained using momentum SGD. For all the datasets except Criteo we report the top-1 error percentage on the test set (i.e. smaller is better), and for Criteo we report the AUC metric (larger is better).



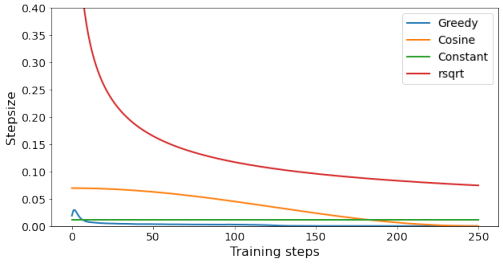
(a) CIFAR10 – error rate



(b) CIFAR10 – stepsizes



(c) Imagenet – error rate



(d) Imagenet – stepsizes

Figure 2: Comparison of the error rate and stepsize for the CIFAR10 and Imagenet datasets between the GLyDER scheduler and other manually tuned schedulers.

performance to the other best performing schedulers, it converges much faster to that solution, around two thirds of the number of steps it takes to the other schedulers. We emphasize that it is possible to run the other schedulers for less training steps, but it reduces their performance, specifically the cosine scheduler which matches the performance of the GLyDER scheduler requires a certain number of steps to reach its optimal solution. Also, note the although the GLyDER stepsize is very small, it is certainly not constant. In fact, it exhibits a sudden drop in the stepsize during its run. It is evident that the GLyDER schedulers is adaptive to the task, and is effected by the gradients which can be seen as an advantage, and it would be interesting to further study this sudden stepsize drop behavior in future works.

5 Discussion and Future Work

We introduced GLyDER, a stepsize scheduler that determines optimal step sizes in the presence of stochastic descent directions. It attains optimal rate in the smooth (not necessarily convex) setting. We also introduce heuristic techniques to enhance its performance. Experimental results show that GLyDER performs on par with manually fine-tuned schedulers.

There are several future research directions which we think could be interesting to study. First, to gain theoretical understanding on how GLyDER performs beyond SGD, e.g. when adding momentum, or utilizing per-parameter stepsizes. Second, extending the theoretical understanding of smoothness estimation, not confined solely to the scope of the GLyDER scheduler, presents a promising area for investigation.. Finally, it would be interesting to further test and improve the Greed scheduler to other types of tasks, such as fine-tuning, self-supervised learning and different network architectures.

Reproducibility statement

All the experiments were performed using the JAX-based (Bradbury *et al.*, 2018) publicly available init2winit framework (Gilmer *et al.*, 2021), using the default framework parameters except for the hyper-parameter search for each data-set and algorithm combination, as noted in Section 4 and detailed in Appendix D. The implementation of algorithm 2 is a straight-forward optimizer-module addition to init2winit and will be made available upon publication. Can we upload?

References

Abadi, Martín, Barham, Paul, Chen, Jianmin, Chen, Zhifeng, Davis, Andy, Dean, Jeffrey, Devin, Matthieu, Ghemawat, Sanjay, Irving, Geoffrey, Isard, Michael, *et al.* 2016. {TensorFlow}: a system for {Large-Scale} machine learning. *Pages 265–283 of: 12th USENIX symposium on operating systems design and implementation (OSDI 16)*.

Anil, Rohan, Gupta, Vineet, Koren, Tomer, Regan, Kevin, & Singer, Yoram. 2020. Scalable second order optimization for deep learning. *arXiv preprint arXiv:2002.09018*.

Attia, Amit, & Koren, Tomer. 2023. SGD with AdaGrad Stepsizes: Full Adaptivity with High Probability to Unknown Parameters, Unbounded Gradients and Affine Variance. *Pages 1147–1171 of: Proceedings of the 40th International Conference on Machine Learning*. Proceedings of Machine Learning Research, vol. 202. PMLR.

Bauschke, HH, & Combettes, PL. 2011. Convex Analysis and Monotone Operator Theory in Hilbert Spaces, 2011. *CMS books in mathematics). DOI*, **10**, 978–1.

Baydin, Atilim Gunes, Cornish, Robert, Rubio, David Martinez, Schmidt, Mark, & Wood, Frank. 2017. Online learning rate adaptation with hypergradient descent. *arXiv preprint arXiv:1703.04782*.

Beck, Amir. 2017. *First-order methods in optimization*. SIAM.

Bradbury, James, Frostig, Roy, Hawkins, Peter, Johnson, Matthew James, Leary, Chris, Maclaurin, Dougal, Necula, George, Paszke, Adam, VanderPlas, Jake, Wanderman-Milne, Skye, & Zhang, Qiao. 2018. JAX: composable transformations of Python+NumPy programs.

Bubeck, Sébastien, *et al.* 2015. Convex optimization: Algorithms and complexity. *Foundations and Trends® in Machine Learning*, **8**(3-4), 231–357.

Chen, Keyi, Langford, John, & Orabona, Francesco. 2022. Better parameter-free stochastic optimization with ODE updates for coin-betting. *Pages 6239–6247 of: Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 36.

Cutkosky, Ashok, & Orabona, Francesco. 2018. Black-box reductions for parameter-free online learning in banach spaces. *Pages 1493–1529 of: Conference On Learning Theory*. PMLR.

Defazio, Aaron, & Mishchenko, Konstantin. 2023. Learning-rate-free learning by D-adaptation. *arXiv preprint arXiv:2301.07733*.

Drori, Yoel, & Shamir, Ohad. 2020. The complexity of finding stationary points with stochastic gradient descent. *Pages 2658–2667 of: International Conference on Machine Learning*. PMLR.

Duchi, John, Hazan, Elad, & Singer, Yoram. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research*, **12**(7).

Foster, Dylan J, Kale, Satyen, Mohri, Mehryar, & Sridharan, Karthik. 2017. Parameter-free online learning via model selection. *Advances in Neural Information Processing Systems*, **30**.

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

Gilmer, Justin M., Dahl, George E., & Nado, Zachary. 2021. init2winit: a JAX codebase for initialization, optimization, and tuning research.

Gupta, Vineet, Koren, Tomer, & Singer, Yoram. 2018. Shampoo: Preconditioned stochastic tensor optimization. *Pages 1842–1850 of: International Conference on Machine Learning*. PMLR.

He, Kaiming, Zhang, Xiangyu, Ren, Shaoqing, & Sun, Jian. 2016. Deep residual learning for image recognition. *Pages 770–778 of: Proceedings of the IEEE conference on computer vision and pattern recognition*.

Ivgi, Maor, Hinder, Oliver, & Carmon, Yair. 2023. DoG is SGD's Best Friend: A Parameter-Free Dynamic Step Size Schedule. *arXiv preprint arXiv:2302.12022*.

Jacobsen, Andrew, & Cutkosky, Ashok. 2022. Parameter-free mirror descent. *Pages 4160–4211 of: Conference on Learning Theory*. PMLR.

Jouppi, Norman P, Young, Cliff, Patil, Nishant, Patterson, David, Agrawal, Gaurav, Bajwa, Raminder, Bates, Sarah, Bhatia, Suresh, Boden, Nan, Borchers, Al, *et al.* 2017. In-datacenter performance analysis of a tensor processing unit. *Pages 1–12 of: Proceedings of the 44th annual international symposium on computer architecture*.

Karimi, Hamed, Nutini, Julie, & Schmidt, Mark. 2016. Linear convergence of gradient and proximal-gradient methods under the polyak-łojasiewicz condition. *Pages 795–811 of: Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer.

Kempka, Michal, Kotlowski, Wojciech, & Warmuth, Manfred K. 2019. Adaptive scale-invariant online algorithms for learning linear models. *Pages 3321–3330 of: International conference on machine learning*. PMLR.

Kingma, Diederik P, & Ba, Jimmy. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Krizhevsky, Alex, Hinton, Geoffrey, *et al.* 2009. Learning multiple layers of features from tiny images.

Li, Xiaoyu, & Orabona, Francesco. 2019. On the convergence of stochastic gradient descent with adaptive stepsizes. *Pages 983–992 of: The 22nd International Conference on Artificial Intelligence and Statistics*. PMLR.

Liu, Hong, Li, Zhiyuan, Hall, David, Liang, Percy, & Ma, Tengyu. 2023. Sophia: A Scalable Stochastic Second-order Optimizer for Language Model Pre-training. *arXiv preprint arXiv:2305.14342*.

Loshchilov, Ilya, & Hutter, Frank. 2016. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*.

Loshchilov, Ilya, & Hutter, Frank. 2017. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*.

Luo, Haipeng, & Schapire, Robert E. 2015. Achieving all with no parameters: Adanormalhedge. *Pages 1286–1304 of: Conference on Learning Theory*. PMLR.

Merity, Stephen, Xiong, Caiming, Bradbury, James, & Socher, Richard. 2016. Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843*.

Naumov, Maxim, Mudigere, Dheevatsa, Shi, Hao-Jun Michael, Huang, Jianyu, Sundaraman, Narayanan, Park, Jongsoo, Wang, Xiaodong, Gupta, Udit, Wu, Carole-Jean, Azzolini, Alisson G, *et al.* 2019. Deep learning recommendation model for personalization and recommendation systems. *arXiv preprint arXiv:1906.00091*.

Orabona, Francesco. 2014. Simultaneous model selection and optimization through parameter-free stochastic learning. *Advances in Neural Information Processing Systems*, **27**.

Orabona, Francesco, & Pál, Dávid. 2016. Coin betting and parameter-free online learning. *Advances in Neural Information Processing Systems*, **29**.

Orabona, Francesco, & Tommasi, Tatiana. 2017. Training deep networks without learning rates through coin betting. *Advances in Neural Information Processing Systems*, **30**.

Paquette, Courtney, & Scheinberg, Katya. 2020. A stochastic line search method with expected complexity analysis. *SIAM Journal on Optimization*, **30**(1), 349–376.

Paszke, Adam, Gross, Sam, Massa, Francisco, Lerer, Adam, Bradbury, James, Chanan, Gregory, Killeen, Trevor, Lin, Zeming, Gimelshein, Natalia, Antiga, Luca, *et al.* 2019. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, **32**.

Rolinek, Michal, & Martius, Georg. 2018. L4: Practical loss-based stepsize adaptation for deep learning. *Advances in neural information processing systems*, **31**.

Russakovsky, Olga, Deng, Jia, Su, Hao, Krause, Jonathan, Satheesh, Sanjeev, Ma, Sean, Huang, Zhiheng, Karpathy, Andrej, Khosla, Aditya, Bernstein, Michael, *et al.* 2015. Imagenet large scale visual recognition challenge. *International journal of computer vision*, **115**, 211–252.

Schaul, Tom, Zhang, Sixin, & LeCun, Yann. 2013. No more pesky learning rates. *Pages 343–351 of: International conference on machine learning*. PMLR.

Shazeer, Noam, & Stern, Mitchell. 2018. Adafactor: Adaptive learning rates with sublinear memory cost. *Pages 4596–4604 of: International Conference on Machine Learning*. PMLR.

Tropp, Joel A, *et al.* 2015. An introduction to matrix concentration inequalities. *Foundations and Trends® in Machine Learning*, **8**(1-2), 1–230.

Vaswani, Sharan, Mishkin, Aaron, Laradji, Issam, Schmidt, Mark, Gidel, Gauthier, & Lacoste-Julien, Simon. 2019. Painless stochastic gradient: Interpolation, line-search, and convergence rates. *Advances in neural information processing systems*, **32**.

Ward, Rachel, Wu, Xiaoxia, & Bottou, Leon. 2020. Adagrad stepsizes: Sharp convergence over nonconvex landscapes. *The Journal of Machine Learning Research*, **21**(1), 9047–9076.

Wiseman, Sam, & Rush, Alexander M. 2016. Sequence-to-sequence learning as beam-search optimization. *arXiv preprint arXiv:1606.02960*.

Wu, Yuhuai, Ren, Mengye, Liao, Renjie, & Grosse, Roger. 2018. Understanding short-horizon bias in stochastic meta-optimization. *arXiv preprint arXiv:1803.02021*.

Zagoruyko, Sergey, & Komodakis, Nikos. 2016. Wide residual networks. *arXiv preprint arXiv:1605.07146*.

Zhang, Michael, Lucas, James, Ba, Jimmy, & Hinton, Geoffrey E. 2019. Lookahead optimizer: k steps forward, 1 step back. *Advances in neural information processing systems*, **32**.

Zhuang, Juntang, Tang, Tommy, Ding, Yifan, Tatikonda, Sekhar C, Dvornek, Nicha, Papademetris, Xenophon, & Duncan, James. 2020. Adabelief optimizer: Adapting stepsizes by the belief in observed gradients. *Advances in neural information processing systems*, **33**, 18795–18806.

A Proofs from Section 2

A.1 Proof of Thm. 2.3

Using the descent lemma (Thm. 2.1) we get that:

$$f(\mathbf{x}_t) - f(\mathbf{x}_{t+1}) \geq \frac{\langle \nabla f(\mathbf{x}_t), \mathbf{g}_t \rangle^2}{2L\|\mathbf{g}_t\|^2}$$

Taking expectation over the noise, and summing for $t = 1, \ldots, T$ we get:

$$\mathbb{E}[f(\mathbf{x}_1) - f(\mathbf{x}_T)] \geq \sum_{t=1}^{T} \mathbb{E}\left[\frac{\langle \nabla f(\mathbf{x}_t), \mathbf{g}_t \rangle^2}{2L\|\mathbf{g}_t\|^2}\right]$$

Dividing both sides by T and using that $f(\mathbf{x}^*)$ is a global minimum we get that:

$$\frac{1}{T} \cdot \mathbb{E}[f(\mathbf{x}_1) - f(\mathbf{x}^*)] \geq \frac{1}{T} \cdot \sum_{t=1}^{T} \mathbb{E}\left[\frac{\langle \nabla f(\mathbf{x}_t), \mathbf{g}_t \rangle^2}{2L\|\mathbf{g}_t\|^2}\right]$$

Which implies that there is $t_0 \in [T]$ such that:

$$\mathbb{E}\left[\frac{\langle \nabla f(\mathbf{x}_{t_0}), \mathbf{g}_{t_0} \rangle^2}{2L\|\mathbf{g}_{t_0}\|^2}\right] \leq \frac{2LR}{T} \tag{4}$$

where we used that $\|f(\mathbf{x}_1) - f(\mathbf{x}^*)\| \leq R$. The noise is independent at every iteration, hence $\mathbb{E}[\mathbf{g}_{t_0}|\mathbf{x}_{t_0}] = \mathbb{E}[\nabla f(\mathbf{x}_{t_0}) + \boldsymbol{\xi}_{t_0}|\mathbf{x}_{t_0}] = \mathbb{E}[\nabla f(\mathbf{x}_{t_0})|\mathbf{x}_{t_0}]$. Applying this we get:

$$\begin{aligned}\left(\mathbb{E}[\|\nabla f(\mathbf{x}_{t_0})\|^2]\right)^2 &= \left(\mathbb{E}[\langle \nabla f(\mathbf{x}_{t_0}), \mathbf{g}_{t_0}\rangle]\right)^2 \\ &= \left(\mathbb{E}\left[\frac{\langle \nabla f(\mathbf{x}_{t_0}), \mathbf{g}_{t_0}\rangle \|\mathbf{g}_{t_0}\|}{\|\mathbf{g}_{t_0}\|}\right]\right)^2 \\ &\leq \mathbb{E}\left[\frac{\langle \nabla f(\mathbf{x}_{t_0}), \mathbf{g}_{t_0}\rangle^2}{\|\mathbf{g}_{t_0}\|^2}\right] \cdot \mathbb{E}[\|\mathbf{g}_{t_0}\|^2] \\ &\leq \frac{2LR}{T}(\mathbb{E}[\|\nabla f(\mathbf{x}_{t_0})\|^2] + \sigma^2), \end{aligned} \tag{5}$$

where in the first inequality we used Cauchy-Schwartz and in the second inequality we used Eq. (4) and the fact that $\mathbb{E}[\|\mathbf{g}_{t_0}\|^2] = \mathbb{E}[\|\nabla f(\mathbf{x}_{t_0}) + \boldsymbol{\xi}_{t_0}\|^2] \leq \mathbb{E}[\|\nabla f(\mathbf{x}_{t_0})\|^2] + \sigma^2$ which is true since $\mathbb{E}[\boldsymbol{\xi}_{t_0}] = 0$ and $\mathbb{E}[\|\boldsymbol{\xi}_{t_0}\|^2] \leq \sigma^2$. In total, Eq. (5) gives us a quadratic inequality in $\mathbb{E}[\|\nabla f(\mathbf{x}_{t_0})\|^2]$. Solving this inequality attains:

$$\mathbb{E}[\|\nabla f(\mathbf{x}_{t_0})\|^2] \leq \frac{2LR}{T} + \sigma\sqrt{\frac{2LR}{T}} .$$

A.2 Proof of Thm. 2.4

We denote $\nabla_t := \nabla f(\mathbf{x}_t)$, also denote $\gamma_t := \frac{\|\nabla_t\|^2}{\|\nabla_t\|^2+\sigma^2}\mathbf{g}_t$. We denote by $\mathbb{E}_t[\cdot]$ expectation conditioned over $\boldsymbol{\xi}_1, \ldots, \boldsymbol{\xi}_{t-1}$, for $t = 1$ this expectation is unconditioned. We first have that:

$$\begin{aligned}\mathbb{E}_t[\langle \gamma_t, \nabla_t\rangle] &= \mathbb{E}_t\left[\langle \frac{\|\nabla_t\|^2}{\|\nabla_t\|^2+\sigma^2}(\nabla_t + \boldsymbol{\xi}_t), \nabla_t\rangle\right] \\ &= \frac{\|\nabla_t\|^4}{\|\nabla_t\|^2+\sigma^2} + \mathbb{E}_t\left[\frac{\|\nabla_t\|^2}{\|\nabla_t\|^2+\sigma^2}\langle \nabla_t, \boldsymbol{\xi}_t\rangle\right] \\ &= \frac{\|\nabla_t\|^4}{\|\nabla_t\|^2+\sigma^2} \end{aligned} \tag{6}$$

where the last equality is since $\mathbb{E}_t[\boldsymbol{\xi}_t] = \mathbb{E}[\boldsymbol{\xi}_t] = 0$. We can also bound:

$$\begin{aligned}\mathbb{E}_t[\|\gamma_t\|^2] &= \mathbb{E}_t\left[\langle \frac{\|\nabla_t\|^2}{\|\nabla_t\|^2+\sigma^2}(\nabla_t + \boldsymbol{\xi}_t), \frac{\|\nabla_t\|^2}{\|\nabla_t\|^2+\sigma^2}\mathbf{g}_t\rangle(\nabla_t + \boldsymbol{\xi}_t)\right] \\ &= \left(\frac{\|\nabla_t\|^2}{\|\nabla_t\|^2+\sigma^2}\right)^2 \cdot \mathbb{E}_t\left[\|\nabla_t\|^2 + 2\langle \nabla_t, \boldsymbol{\xi}_t\rangle + \|\boldsymbol{\xi}_t\|^2\right] \\ &\leq \left(\frac{\|\nabla_t\|^2}{\|\nabla_t\|^2+\sigma^2}\right)^2 \cdot (\|\nabla_t\|^2 + \sigma^2) \\ &= \frac{\|\nabla_t\|^4}{\|\nabla_t\|^2+\sigma^2} \end{aligned} \tag{7}$$

Using the descent lemma in expectation over $\boldsymbol{\xi}_t$ with Eq. (6) and Eq. (7) we get:

$$\begin{aligned}\mathbb{E}_t[f(\mathbf{x}_t) - f(\mathbf{x}_{t+1})] &\geq \mathbb{E}_t\left[\langle \nabla_t, \frac{1}{L}\gamma_t\rangle - \frac{L}{2}\left\|\frac{1}{L}\cdot\gamma_t\right\|^2\right] \\ &= \frac{1}{L}\mathbb{E}_t[\langle \nabla_t, \gamma_t\rangle] - \frac{L}{2L^2}\mathbb{E}_t[\|\gamma_t\|^2] \\ &\geq \frac{1}{L}\frac{\|\nabla_t\|^4}{\|\nabla_t\|^2+\sigma^2} - \frac{1}{2L}\frac{\|\nabla_t\|^4}{\|\nabla_t\|^2+\sigma^2} \\ &= \frac{1}{2L}\cdot\frac{\|\nabla_t\|^4}{\|\nabla_t\|^2+\sigma^2} \end{aligned}$$

Summing over all the iterations, dividing by T and taking expectation w.r.t $\boldsymbol{\xi}_1, \ldots, \boldsymbol{\xi}_T$ we get:

$$\begin{aligned}\frac{1}{T} \cdot \mathbb{E}\left[f(\mathbf{x}_1) - f(\mathbf{x}_T)\right] &= \frac{1}{T} \cdot \mathbb{E}\left[\sum_{t=1}^{T} f(\mathbf{x}_t) - f(\mathbf{x}_{t+1})\right] \\ &= \frac{1}{T}\sum_{t=1}^{T} \mathbb{E}\left[f(\mathbf{x}_t) - f(\mathbf{x}_{t+1})\right] \\ &= \frac{1}{T}\sum_{t=1}^{T} \mathbb{E}\left[\mathbb{E}_t\left[f(\mathbf{x}_t) - f(\mathbf{x}_{t+1})\right]\right] \\ &\geq \frac{1}{T}\sum_{t=1}^{T} \mathbb{E}\left[\frac{1}{2L} \cdot \frac{\|\nabla_t\|^4}{\|\nabla_t\|^2 + \sigma^2}\right].\end{aligned}$$

Note that on the right hand side we have mean over T, and the minimum over $t = 1, \ldots, T$ is smaller than the mean:

$$\begin{aligned}\min_{t=1,\ldots,T} \frac{1}{2L} \cdot \mathbb{E}\left[\frac{\|\nabla_t\|^4}{\|\nabla_t\|^2 + \sigma^2}\right] &\leq \frac{1}{T} \cdot \mathbb{E}\left[f(\mathbf{x}_1) - f(\mathbf{x}_T)\right] \\ &\leq \frac{1}{T} \cdot \mathbb{E}\left[f(\mathbf{x}_1) - f(\mathbf{x}^*)\right] \leq \frac{R}{T}\end{aligned}$$

where $\mathbf{x}^*$ is a global minimum of f, hence $f(\mathbf{x}_T) \leq f(\mathbf{x}^*)$. Rearranging the inequality we get:

$$\min_{t=0,\ldots,T} \mathbb{E}\left[\frac{\|\nabla_t\|^4}{\|\nabla_t\|^2 + \sigma^2}\right] \leq \frac{2LR}{T} \tag{8}$$

Denote the iteration that attains the minimum as t_0, and denote $\nabla := \nabla_{t_0}$. We now have the following:

$$\begin{aligned}\left(\mathbb{E}\left[\|\nabla\|^2\right]\right)^2 &= \left(\mathbb{E}\left[\frac{\|\nabla\|^2}{\sqrt{\|\nabla\|^2 + \sigma^2}} \cdot \sqrt{\|\nabla\|^2 + \sigma^2}\right]\right)^2 \\ &\leq \mathbb{E}\left[\frac{\|\nabla\|^4}{\|\nabla\|^2 + \sigma^2}\right] \cdot \mathbb{E}\left[\|\nabla\|^2 + \sigma^2\right] \\ &\leq \frac{2LR}{T}\left(\mathbb{E}\left[\|\nabla\|^2\right] + \sigma^2\right)\end{aligned}$$

where in the first inequality we used Cauchy-Schwartz, and in the second we used Eq. (8). In total, we got a quadratic inequality on $\mathbb{E}\left[\|\nabla\|^2\right]$. Solving this inequality attains:

$$\mathbb{E}[\|\nabla\|^2] \leq \frac{2LR}{T} + \sigma\sqrt{\frac{2LR}{T}}\,.$$

A.3 Proof of Thm. 2.5

We denote $\nabla_t := \nabla f(\mathbf{x}_t)$, and also assume w.l.o.g that $M, G, \sigma \geq 1$, otherwise we just replace them in the proofs with 1. Denote the variance of the noise at iteration t by $\sigma_t := \mathbb{E}[\|\boldsymbol{\xi}_t^i\|^2]$, note that this term is unknown and bounded above by σ^2. To distinguish between the two sets of stochastic gradients the noise vectors as $\boldsymbol{\xi}_t^i$, $\boldsymbol{\zeta}_t^i$ for $\mathbf{h}_t^i$ and $\mathbf{g}_t^i$ respectfully. That is, we write $\mathbf{h}_t^i = \nabla f(\mathbf{x}_t) + \boldsymbol{\xi}_t^i$, $\mathbf{g}_t^i = \nabla f(\mathbf{x}_t) + \boldsymbol{\zeta}_t^i$.

First note that the estimators μ_t and γ_t can be simplified in the following manner:

$$\begin{aligned}\mu_t &= \frac{1}{n(n-1)}\sum_{i\neq j}\langle \mathbf{h}_t^i, \mathbf{h}_t^j\rangle = \frac{1}{n(n-1)}\left(\left\|\sum_{i=1}^{n}\mathbf{h}_t^i\right\|^2 - \sum_{i=1}^{n}\|\mathbf{h}_t^i\|^2\right) \\ \gamma_t &= \frac{1}{n^2}\sum_{i,j=1}^{n}\langle \mathbf{h}_t^i, \mathbf{h}_t^j\rangle = \frac{1}{n^2}\left\|\sum_{i=1}^{n}\mathbf{h}_t^i\right\|^2\end{aligned}$$

We first focus on γ_t, we can write

$$\frac{1}{n^2}\left\|\sum_{i=1}^{n}\mathbf{h}_t^i\right\|^2 = \left\|\nabla_t + \frac{1}{n}\sum_{i=1}^{n}\boldsymbol{\xi}_t^i\right\|^2$$
$$= \|\nabla_t\|^2 + \frac{2}{n}\sum_{i=1}^{n}\langle\nabla_t, \boldsymbol{\xi}_t^i\rangle + \left\|\frac{1}{n}\sum_{i=1}^{n}\boldsymbol{\xi}_t^i\right\|^2 \tag{9}$$

We will show that the two last terms in Eq. (9) are close to zero w.h.p. For the first term, we rewrite it as $\sum_{i=1}^{n}\langle\nabla_t, \frac{2}{n}\boldsymbol{\xi}_t^i\rangle$. For every i, the random variable $\langle\nabla_t, \frac{2}{n}\boldsymbol{\xi}_t^i\rangle$ has zero mean, since $\mathbb{E}[\boldsymbol{\xi}_i^t] = 0$. Also by Cauchy-Schwartz, $|\langle\nabla_t, \frac{2}{n}\boldsymbol{\xi}_t^i\rangle| \leq \frac{2MG}{n}$. Using Hoeffding's inequality we get:

$$P\left(\left|\frac{2}{n}\sum_{i=1}^{n}\langle\nabla_t, \boldsymbol{\xi}_t^i\rangle\right| \geq \frac{\epsilon}{40}\right) = P\left(\left|\sum_{i=1}^{n}\langle\nabla_t, 2\boldsymbol{\xi}_t^i\rangle\right| \geq \frac{n\epsilon}{40}\right)$$
$$\leq 2\exp\left(-\frac{2\epsilon^2 n^2}{n(160MG)^2}\right) = 2\exp\left(-\frac{2\epsilon^2 n}{(160MG)^2}\right) \tag{10}$$

For the last term in Eq. (9), we will use a matrix version of Bernstein's inequality, see Theorem 6.1.1 in Tropp *et al.* (2015). We have $\left\|\frac{1}{n}\boldsymbol{\xi}_t^i\right\| \leq \frac{G}{n}$ and $\mathbb{E}\left[\frac{1}{n}\boldsymbol{\xi}_t^i\right] = 0$. Denote $Z = \sum_{i=1}^{n}\frac{1}{n}\boldsymbol{\xi}_t^i$, to use Bernstein's inequality we need to bound:

$$\nu(Z) := \max\left\{\left\|\sum_{i=1}^{n}\frac{1}{n^2}\mathbb{E}[\|\boldsymbol{\xi}_t^i\|^2]\right\|, \left\|\sum_{i=1}^{n}\frac{1}{n^2}\mathbb{E}[\boldsymbol{\xi}_t^i\boldsymbol{\xi}_t^{i^\top}]\right\|\right\}$$

The first term by our assumptions is bounded by $\frac{\sigma_t^2}{n}$. The second term can be bounded in the following way:

$$\left\|\sum_{i=1}^{n}\frac{1}{n^2}\mathbb{E}[\boldsymbol{\xi}_t^i\boldsymbol{\xi}_t^{i^\top}]\right\| \leq \frac{1}{n^2}\sum_{i=1}^{n}\left\|\mathbb{E}[\boldsymbol{\xi}_t^i\boldsymbol{\xi}_t^{i^\top}]\right\|$$
$$\leq \frac{1}{n^2}\sum_{i=1}^{n}\mathbb{E}[\left\|\boldsymbol{\xi}_t^i\boldsymbol{\xi}_t^{i^\top}\right\|]$$

Where in the last inequality we used Jensen's inequality. Each matrix $\boldsymbol{\xi}_t^i\boldsymbol{\xi}_t^{i^\top}$ is rank-1 with an eigenvalue equal to $\mathbb{E}[\|\boldsymbol{\xi}_t^i\|^2]$. Hence, in total this term can also be bounded by $\frac{\sigma_t^2}{n}$. Using matrix Bernstein's inequality we get:

$$P\left(\left\|\frac{1}{n}\sum_{i=1}^{n}\boldsymbol{\xi}_t^i\right\|^2 \geq \frac{\epsilon}{40}\right) = P\left(\left\|\frac{1}{n}\sum_{i=1}^{n}\boldsymbol{\xi}_t^i\right\| \geq \sqrt{\frac{\epsilon}{40}}\right)$$
$$\leq (d+1)\exp\left(-\frac{\epsilon}{80(\sigma_t^2/n + G\sqrt{\epsilon}/3n)}\right)$$
$$\leq (d+1)\exp\left(-\frac{\epsilon n}{80\sigma_t^2 G}\right) \tag{11}$$

Using a union bound on the events in Eq. (10) and Eq. (11), we get that there is a universal constant $c_1 > 0$ such that:

$$P\left(\left|\gamma_t - \|\nabla_t\|^2\right| \geq \frac{\epsilon}{20}\right) = P\left(\left|\frac{2}{n}\sum_{i=1}^{n}\langle\nabla_t, \boldsymbol{\xi}_t^i\rangle + \left\|\frac{1}{n}\sum_{i=1}^{n}\boldsymbol{\xi}_t^i\right\|^2\right| \geq \frac{\epsilon}{20}\right)$$
$$\leq 2\exp\left(-\frac{2\epsilon^2 n}{(160MG)^2}\right) + (d+1)\exp\left(-\frac{\epsilon n}{80\sigma_t^2 G}\right)$$
$$\leq (d+3)\exp\left(-\frac{\epsilon^2 n c_1}{\sigma_t^2 M^2 G^2}\right)$$

We will now use a similar method to bound μ_t. Note that μ_t have two terms, both multiplied by a coefficient of $\frac{1}{n(n-1)}$. The second term can be written as:

$$\sum_{i=1}^{n} \|\mathbf{h}_t^i\|^2 = \sum_{i=1}^{n} \|\nabla_t + \boldsymbol{\xi}_t^i\|^2 = n\|\nabla_t\|^2 + \sum_{i=1}^{n} \langle \nabla_t, \boldsymbol{\xi}_t^i \rangle + \sum_{i=1}^{n} \|\boldsymbol{\xi}_t^i\|^2 \tag{12}$$

By our assumption that $n \geq 2$ we have that $\frac{1}{n(n-1)} \left\| \sum_{i=1}^{n} \mathbf{h}_t^i \right\| \leq \frac{2}{n^2} \left\| \sum_{i=1}^{n} \mathbf{h}_t^i \right\|$. Hence, by a similar analysis as above, there is a constant $c_2 > 0$ such that:

$$P\left(\left| \frac{1}{n(n-1)} \left(\left\| \sum_{i=1}^{n} \mathbf{h}_t^i \right\|^2 - n\|\nabla_t\|^2 \right) - \|\nabla_t\|^2 \right| \geq \frac{\epsilon}{60} \right) \leq (d+3) \exp\left(-\frac{\epsilon^2 n c_2}{\sigma^2 M^2 G^2} \right) \tag{13}$$

To bound the second of Eq. (12) we use that $\langle \nabla_t, \boldsymbol{\xi}_t^i \rangle \leq GM$, $\|\boldsymbol{\xi}_t^i\| \leq G$ and Hoeffding's inequality to get:

$$P\left(\left| \frac{1}{n(n-1)} \sum_{i=1}^{n} \langle \nabla_t, \boldsymbol{\xi}_t^i \rangle \right| \geq \frac{\epsilon}{60} \right) \leq 2 \exp\left(-\frac{\epsilon^2 n(n-1)^2}{(60GM)^2} \right) \tag{14}$$

For the third term of Eq. (12) we also use Hoeffding's inequality:

$$P\left(\left| \frac{1}{n(n-1)} \sum_{i=1}^{n} \|\boldsymbol{\xi}_t^i\|^2 - \frac{\sigma_t^2}{n} \right| \geq \frac{\epsilon}{60} \right) \leq \exp\left(-\frac{\epsilon^2 (n-1)^2}{n(60G)^2} \right) \tag{15}$$

Combining Eq. (13), Eq. (14) and Eq. (15) we get that there is a constant $c_3 > 0$ such that:

$$P\left(\left| \mu_t - \|\nabla_t\|^2 + \frac{\sigma_t^2}{n} \right| \geq \frac{\epsilon}{20} \right) \leq (d+5) \exp\left(-\frac{\epsilon^2 n c_3}{\sigma_t^2 M^2 G^2} \right)$$

Finally, we combine both probability bounds on μ_1^t and μ_2^t to get that there is some constant $c > 0$ s.t w.p $> 1 - d \exp\left(-\frac{\epsilon^2 n c}{\sigma^2 M^2 G^2} \right)$ we have both:

$$\left| \mu_t - \|\nabla_t\|^2 + \frac{\sigma_t^2}{n} \right| \leq \frac{\epsilon}{20} \tag{16}$$

$$\left| \gamma_t - \|\nabla_t\|^2 \right| \leq \frac{\epsilon}{20} \tag{17}$$

Note that inside the exponent we replaced σ_t with σ, since this is a lower bound on the probability and $\sigma_t \leq \sigma$.

We would like to condition on the above events, however these events also depend on the given ∇_t for iteration t, while ∇_t depends on all the noise vectors from previous iterations. To this end, we denote by $\mathbb{E}_t[\cdot]$ the expectation conditioned on the noise vectors $\boldsymbol{\xi}_1^i, \ldots \boldsymbol{\xi}_t^i$, $\boldsymbol{\zeta}_1, \ldots, \boldsymbol{\zeta}_{t-1}$ for every $i \in [n]$ and on the events in Eq. (16) and Eq. (17). Specifically note that we conditioned on the noise vectors $\boldsymbol{\xi}_t^i$ including the current iteration, so that we could also condition on the two events above, while on the noise vectors $\boldsymbol{\zeta}_t^i$ we haven't conditioned on the current iteration, since we would use that it is drawn independently of all the previous noise vectors with zero mean.

We also denote $\alpha_t := \frac{\mu_t}{\gamma_t}\mathbf{g}_t$, and we use the assumption that $n \geq \frac{20\sigma^2}{\epsilon}$ to get that:

$$\begin{aligned}
\mathbb{E}_t\left[\langle \alpha_t, \nabla_t\rangle\right] &= \mathbb{E}_t\left[\langle \frac{\mu_1^t}{\mu_2^t}\mathbf{g}_t, \nabla_t\rangle\right] \\
&\geq \mathbb{E}_t\left[\langle \frac{\|\nabla_t\|^2 - \frac{\sigma_t^2}{n} - \epsilon/20}{\|\nabla_t\|^2 + \epsilon/20}\left(\nabla_t + \frac{1}{n}\sum_{i=1}^n \zeta_t^i\right), \nabla_t\rangle\right] \\
&\geq \mathbb{E}_t\left[\langle \frac{\|\nabla_t\|^2 - \epsilon/10}{\|\nabla_t\|^2 + \frac{\sigma_t^2}{n} + \epsilon/20}\left(\nabla_t + \frac{1}{n}\sum_{i=1}^n \zeta_t^i\right), \nabla_t\rangle\right] \\
&\geq \frac{18}{21}\mathbb{E}_t\left[\langle \frac{\|\nabla_t\|^2}{\|\nabla_t\|^2 + \frac{\sigma_t^2}{n}}\left(\nabla_t + \frac{1}{n}\sum_{i=1}^n \zeta_t^i\right), \nabla_t\rangle\right] \\
&= \frac{18}{21}\cdot\frac{\|\nabla_t\|^4}{\|\nabla_t\|^2 + \frac{\sigma_t^2}{n}} + \frac{18}{21n}\sum_{i=1}^n \mathbb{E}_t\left[\frac{\|\nabla_t\|^2}{\|\nabla_t\|^2 + \frac{\sigma_t^2}{n}}\langle \nabla_t, \zeta_t^i\rangle\right] \\
&= \frac{18}{21}\cdot\frac{\|\nabla_t\|^4}{\|\nabla_t\|^2 + \frac{\sigma_t^2}{n}}
\end{aligned} \tag{18}$$

where we used that $\mathbb{E}_t[\zeta_t^i] = \mathbb{E}[\zeta_t^i] = 0$. We also have that:

$$\begin{aligned}
\mathbb{E}_t\left[\|\alpha_t\|^2\right] &= \mathbb{E}_t\left[\langle \frac{\mu_t}{\gamma_t}\left(\nabla_t + \frac{1}{n}\sum_{i=1}^n \zeta_t^i\right), \frac{\mu_t}{\gamma_t}\left(\nabla_t + \frac{1}{n}\sum_{i=1}^n \zeta_t^i\right)\rangle\right] \\
&= \left(\frac{\mu_t}{\gamma_t}\right)^2 \cdot \mathbb{E}_t\left[\|\nabla_t\|^2 + \frac{2}{n}\sum_{i=1}^n \langle \nabla_t, \zeta_t^i\rangle + \left\|\frac{1}{n}\sum_{i=1}^n \zeta_t^i\right\|^2\right] \\
&\leq \left(\frac{\|\nabla_t\|^2 - \frac{\sigma_t^2}{n} + \epsilon/20}{\|\nabla_t\|^2 - \epsilon/20}\right)^2 \cdot \left(\|\nabla_t\|^2 + \frac{\sigma_t^2}{n}\right) \\
&\leq \left(\frac{\|\nabla_t\|^2 + \epsilon/20}{\|\nabla_t\|^2 + \frac{\sigma_t^2}{n} - \epsilon/10}\right)^2 \cdot \left(\|\nabla_t\|^2 + \frac{\sigma_t^2}{n}\right) \\
&\leq \left(\frac{21}{18}\right)^2 \cdot \left(\frac{\|\nabla_t\|^2}{\|\nabla_t\|^2 + \frac{\sigma_t^2}{n}}\right) \cdot \left(\|\nabla_t\|^2 + \frac{\sigma_t^2}{n}\right) \\
&= \left(\frac{21}{18}\right)^2 \frac{\|\nabla_t\|^4}{\|\nabla_t\|^2 + \frac{\sigma_t^2}{n}}
\end{aligned} \tag{19}$$

Combining Eq. (18) and Eq. (19) and the descent lemma we get that:

$$\begin{aligned}
\mathbb{E}_t\left[f(\mathbf{x}_t) - f(\mathbf{x}_{t+1})\right] &\geq \mathbb{E}_t\left[\langle \nabla_t, \frac{1}{L}\alpha_t\rangle - \frac{L}{2}\left\|\frac{1}{L}\cdot\alpha_t\right\|^2\right] \\
&= \frac{1}{L}\mathbb{E}_t\left[\langle \nabla_t, \alpha_t\rangle\right] - \frac{L}{2L^2}\mathbb{E}_t\left[\|\alpha_t\|^2\right] \\
&\geq \frac{18}{21L}\frac{\|\nabla_t\|^4}{\|\nabla_t\|^2 + \frac{\sigma_t^2}{n}} - \frac{441}{648L}\frac{\|\nabla_t\|^4}{\|\nabla_t\|^2 + \frac{\sigma_t^2}{n}} \\
&\geq \frac{1}{6L}\cdot\frac{\|\nabla_t\|^4}{\|\nabla_t\|^2 + \frac{\sigma_t^2}{n}} \\
&\geq \frac{1}{6L}\cdot\frac{\|\nabla_t\|^4}{\|\nabla_t\|^2 + \frac{\sigma^2}{n}}
\end{aligned}$$

where in the last inequality we used that $\sigma_t \leq \sigma$. We now follow in the same manner as in Thm. 2.3 where we replace σ^2 by $\frac{\sigma^2}{n}$. Namely, we sum over all the iteration $t \in [T]$, divide by T and take expectation w.r.t all the noise vectors $\boldsymbol{\xi}_t^i,\ \boldsymbol{\zeta}_t^i$ for $t \in [T],\ i \in [n]$ to get:

$$\begin{aligned}
\frac{1}{T}\cdot\mathbb{E}\left[f(\mathbf{x}_1) - f(\mathbf{x}_T)\right] &= \frac{1}{T}\cdot\mathbb{E}\left[\sum_{t=1}^{T} f(\mathbf{x}_t) - f(\mathbf{x}_{t+1})\right] \\
&= \frac{1}{T}\sum_{t=1}^{T}\mathbb{E}\left[f(\mathbf{x}_t) - f(\mathbf{x}_{t+1})\right] \\
&= \frac{1}{T}\sum_{t=1}^{T}\mathbb{E}\left[\mathbb{E}_t\left[f(\mathbf{x}_t) - f(\mathbf{x}_{t+1})\right]\right] \\
&\geq \frac{1}{T}\sum_{t=1}^{T}\mathbb{E}\left[\frac{1}{6L}\cdot\frac{\|\nabla_t\|^4}{\|\nabla_t\|^2 + \frac{\sigma^2}{n}}\right].
\end{aligned}$$

On the right hand side of the above inequality we have mean over T, and the minimum over $t = 1, \ldots, T$ is smaller than the mean:

$$\begin{aligned}
\min_{t=1,\ldots,T}\frac{1}{6L}\cdot\mathbb{E}\left[\frac{\|\nabla_t\|^4}{\|\nabla_t\|^2 + \frac{\sigma^2}{n}}\right] &\leq \frac{1}{T}\cdot\mathbb{E}\left[f(\mathbf{x}_1) - f(\mathbf{x}_T)\right] \\
&\leq \frac{1}{T}\cdot\mathbb{E}\left[f(\mathbf{x}_1) - f(\mathbf{x}^*)\right] \leq \frac{R}{T}
\end{aligned}$$

where $\mathbf{x}^*$ is a global minimum of f, hence $f(\mathbf{x}_T) \leq f(\mathbf{x}^*)$. Rearranging the inequality we get:

$$\min_{t=1,\ldots,T}\mathbb{E}\left[\frac{\|\nabla_t\|^4}{\|\nabla_t\|^2 + \frac{\sigma^2}{n}}\right] \leq \frac{6LR}{T}\ . \tag{20}$$

Denote the iteration that attains the minimum as t_0, and denote $\nabla := \nabla_{t_0}$. We now have the following:

$$\begin{aligned}
\left(\mathbb{E}\left[\|\nabla\|^2\right]\right)^2 &= \left(\mathbb{E}\left[\frac{\|\nabla\|^2}{\sqrt{\|\nabla\|^2 + \frac{\sigma^2}{n}}}\cdot\sqrt{\|\nabla\|^2 + \frac{\sigma^2}{n}}\right]\right)^2 \\
&\leq \mathbb{E}\left[\frac{\|\nabla\|^4}{\|\nabla\|^2 + \frac{\sigma^2}{n}}\right]\cdot\mathbb{E}\left[\|\nabla\|^2 + \frac{\sigma^2}{n}\right] \\
&\leq \frac{6LR}{T}\left(\mathbb{E}\left[\|\nabla\|^2\right] + \frac{\sigma^2}{n}\right)
\end{aligned}$$

where in the first inequality we used Cauchy-Schwartz, and in the second we used Eq. (20). In total, we got a quadratic inequality on $\mathbb{E}\left[\|\nabla\|^2\right]$. Solving this inequality attains:

$$\mathbb{E}[\|\nabla\|^2] \leq \frac{6LR}{T} + \frac{\sigma}{\sqrt{n}}\sqrt{\frac{6LR}{T}} \ .$$

In particular, by our choice of $n = \Omega\left(\frac{\sigma^2}{\epsilon}\right)$ we get that after $T = O\left(\frac{1}{\epsilon}\right)$ iterations we achieve $\mathbb{E}[\|\nabla\|^2] \leq \epsilon$. Recall that this convergence result is conditioned on the events in Eq. (16) and Eq. (17), which happens w.p. $> 1 - dT\exp\left(-\frac{\epsilon^2 nc}{\sigma^2 M^2 G^2}\right)$. In total, we get the convergence result w.h.p. by choosing $n = \Omega\left(\frac{\sigma^2 M^2 G^2 \log(d)}{\epsilon^2}\right)$, where we achieve $\mathbb{E}[\|\nabla\|^2] \leq \epsilon$ by receiving a total of $O\left(\frac{\sigma^2 M^2 G^2 \log(d)}{\epsilon^3}\right)$ noisy gradients.

B Proof from Section 2.2

B.1 Proof of Thm. 3.1

We first have that:

$$\begin{aligned}
\mathbb{E}[\mu] &= \frac{1}{n(n-1)}\sum_{i\neq j}\mathbb{E}[\langle \mathbf{g}_i, \mathbf{g}_j\rangle] \\
&= \frac{1}{n(n-1)}\sum_{i\neq j}\mathbb{E}[\langle \nabla + \boldsymbol{\xi}_i, \nabla + \boldsymbol{\xi}_j\rangle] \\
&= \|\nabla\| + \frac{1}{n(n-1)}\sum_{i\neq j}\mathbb{E}[\langle \boldsymbol{\xi}_i, \boldsymbol{\xi}_j\rangle] = \|\nabla\|
\end{aligned}$$

where we used that the $\boldsymbol{\xi}_i$'s are independent with zero mean. For the variance calculation, we have:

$$\begin{aligned}
\mathrm{Var}(\mu) &= \mathrm{Var}\left(\frac{1}{n(n-1)}\sum_{i\neq j}\langle \mathbf{g}_i, \mathbf{g}_j\rangle\right) \\
&= \frac{1}{n^2(n-1)^2}\mathrm{Var}\left(\sum_{i\neq j}\langle \nabla + \boldsymbol{\xi}_i, \nabla + \boldsymbol{\xi}_j\rangle\right) \\
&= \frac{1}{n^2(n-1)^2}\mathrm{Var}\left(n(n-1)\|\nabla\| + \sum_{i\neq j}\langle \nabla, \boldsymbol{\xi}_i\rangle + \langle \nabla, \boldsymbol{\xi}_j\rangle + \langle \boldsymbol{\xi}_i, \boldsymbol{\xi}_j\rangle\right) \\
&= \frac{1}{n^2(n-1)^2}\mathrm{Var}\left(2(n-1)\sum_{i=1}^{n}\langle \nabla, \boldsymbol{\xi}_i\rangle + \sum_{i\neq j}\langle \boldsymbol{\xi}_i, \boldsymbol{\xi}_j\rangle\right)
\end{aligned}$$

Here, we used that ∇ is a constant (non-random) vector. We calculate this variance by extending it as an expectation:

$$\begin{aligned}
\mathrm{Var}(\mu) = \frac{1}{n^2(n-1)^2}\Bigg(&\mathbb{E}\left[\left(2(n-1)\sum_{i=1}^{n}\langle \nabla, \boldsymbol{\xi}_i\rangle + \sum_{i\neq j}\langle \boldsymbol{\xi}_i, \boldsymbol{\xi}_j\rangle\right)^2\right] \\
&- \left(\mathbb{E}\left[2(n-1)\sum_{i=1}^{n}\langle \nabla, \boldsymbol{\xi}_i\rangle + \sum_{i\neq j}\langle \boldsymbol{\xi}_i, \boldsymbol{\xi}_j\rangle\right]\right)^2\Bigg)
\end{aligned}$$

We will calculate each term of the above separately. For the second term, note that $\mathbb{E}[\langle \nabla, \boldsymbol{\xi}_i\rangle] = 0$, and for $i \neq j$: $\mathbb{E}[\langle \boldsymbol{\xi}_i, \boldsymbol{\xi}_j\rangle] = 0$. Hence, by the linearity of the expectation, the second term is equal to zero. The first term (without

the coefficient) is equal to:

$$\begin{aligned}
&\mathbb{E}\left[\left(2(n-1)\sum_{i=1}^{n}\langle\nabla,\boldsymbol{\xi}_i\rangle+\sum_{i\neq j}\langle\boldsymbol{\xi}_i,\boldsymbol{\xi}_j\rangle\right)^2\right]\\
=&\mathbb{E}\left[4(n-1)^2\left(\sum_{i=1}^{n}\langle\nabla,\boldsymbol{\xi}_i\rangle\right)^2\right]+\mathbb{E}\left[\left(\sum_{i\neq j}\langle\boldsymbol{\xi}_i,\boldsymbol{\xi}_j\rangle\right)^2\right]+\\
&+\mathbb{E}\left[4(n-1)\left(\sum_{i=1}^{n}\langle\nabla,\boldsymbol{\xi}_i\rangle\right)\cdot\left(\sum_{k\neq j}\langle\boldsymbol{\xi}_k,\boldsymbol{\xi}_j\rangle\right)\right] \qquad (21)
\end{aligned}$$

For the first term in Eq. (21) we have:

$$\begin{aligned}
\mathbb{E}\left[4(n-1)^2\left(\sum_{i=1}^{n}\langle\nabla,\boldsymbol{\xi}_i\rangle\right)^2\right]&=4(n-1)^2\mathbb{E}\left[\sum_{i=1}^{n}\sum_{j=1}^{n}\langle\nabla,\boldsymbol{\xi}_i\rangle\cdot\langle\nabla,\boldsymbol{\xi}_j\rangle\right]\\
&=4(n-1)^2\mathbb{E}\left[\sum_{i=1}^{n}\langle\nabla,\boldsymbol{\xi}_i\rangle^2\right]\\
&\leq 4(n-1)^2\sum_{i=1}^{n}\mathbb{E}\left[\|\nabla^2\|\cdot\|\boldsymbol{\xi}_i\|^2\right] \qquad (22)\\
&=4n(n-1)^2\|\nabla\|^2\sigma^2
\end{aligned}$$

where we used Cauchy-Schwartz and that the $\boldsymbol{\xi}_i$'s are independent with zero mean, hence $\mathbb{E}[\langle\nabla,\boldsymbol{\xi}_i\rangle\cdot\langle\nabla,\boldsymbol{\xi}_j\rangle]=0$ for $i\neq j$. For the second term in Eq. (21) we have:

$$\begin{aligned}
\mathbb{E}\left[\left(\sum_{i\neq j}\langle\boldsymbol{\xi}_i,\boldsymbol{\xi}_j\rangle\right)^2\right]&=\mathbb{E}\left[\left(\sum_{i\neq j}\langle\boldsymbol{\xi}_i,\boldsymbol{\xi}_j\rangle\right)\cdot\left(\sum_{k\neq M}\langle\boldsymbol{\xi}_k,\boldsymbol{\xi}_M\rangle\right)\right]\\
&=\sum_{i\neq j}\sum_{k\neq M}\mathbb{E}\left[\langle\boldsymbol{\xi}_i,\boldsymbol{\xi}_j\rangle\cdot\langle\boldsymbol{\xi}_k,\boldsymbol{\xi}_M\rangle\right]\\
&=\sum_{i\neq j}\mathbb{E}\left[\langle\boldsymbol{\xi}_i,\boldsymbol{\xi}_j\rangle^2\right]\\
&\leq\sum_{i\neq j}\mathbb{E}[\|\boldsymbol{\xi}_i\|^2\cdot\|\boldsymbol{\xi}_j\|^2] \qquad (23)\\
&=n(n-1)\sigma^4
\end{aligned}$$

For the third term in Eq. (21) we get:

$$\begin{aligned}
&E\left[4(n-1)\left(\sum_{i=1}^{n}\langle\nabla,\boldsymbol{\xi}_i\rangle\right)\cdot\left(\sum_{k\neq j}\langle\boldsymbol{\xi}_k,\boldsymbol{\xi}_j\rangle\right)\right]\\
=&4(n-1)\sum_{i=1}^{n}\sum_{k\neq j}\mathbb{E}\left[\langle\nabla,\boldsymbol{\xi}_i\rangle\cdot\langle\boldsymbol{\xi}_k,\boldsymbol{\xi}_j\rangle\right]=0
\end{aligned}$$

where again we used the the $\boldsymbol{\xi}_i$'s are independent with zero mean. Combining the three terms above, we get that:

$$\mathrm{Var}(\mu)\leq\frac{4}{n}\|\nabla\|^2\sigma^2+\frac{1}{n(n-1)}\sigma^4$$

In particular, if we assume that $\boldsymbol{\xi}_i\sim\mathcal{N}\left(0,\frac{\sigma^2}{d}I_d\right)$, then instead of the inequalities in Eq. (22) and Eq. (23) we can derive equalities. For Eq. (22) we have:

$$4(n-1)^2\mathbb{E}\left[\sum_{i=1}^{n}\langle\nabla,\boldsymbol{\xi}_i\rangle^2\right]=4n(n-1)^2\|\nabla\|\frac{\sigma^2}{d}$$

where we used that $\boldsymbol{\xi}_i$ have a spherically symmetric distribution, hence $\langle\nabla,\boldsymbol{\xi}_i\rangle$ have the same distribution for any fixed ∇, then can assume w.l.o.g that $\nabla=\|\nabla\|\cdot\mathbf{e}_1$. For Eq. (23) we have

$$\begin{aligned}\sum_{i\neq j}\mathbb{E}\left[\langle\boldsymbol{\xi}_i,\boldsymbol{\xi}_j\rangle^2\right]&=\sum_{i\neq j}\mathbb{E}\left[\left(\sum_{k=1}^{d}\xi_{i,k}\xi_{j,k}\right)^2\right]\\&=\sum_{i\neq j}\mathbb{E}\left[\sum_{k=1}^{d}\xi_{i,k}^2\xi_{j,k}^2\right]\\&=\sum_{i\neq j}\sum_{k=1}^{d}\mathbb{E}[\xi_{i,k}^2]\mathbb{E}[\xi_{j,k}^2]\\&=\frac{\sigma^4}{dn(n-1)}\end{aligned}$$

where we used that each coordinate is distributed i.i.d. Summing everything together, we have that:

$$\mathrm{Var}(\mu)=\frac{4\|\nabla\|\sigma^2}{nd}+\frac{\sigma^4}{dn(n-1)}$$

C Additional Details for the Practical Implementation of GLyDER

Efficient implementation of the inner products. Note that to efficiently estimate the norm using Eq. (3) with n stochastic gradients we need to do $O(n^2)$ inner products. Instead, we can use the formulas:

$$\left\|\sum_{i=1}^{n}\mathbf{g}_t^i\right\|^2-\sum_{i=1}^{n}\left\|\mathbf{g}_t^i\right\|^2=\sum_{i\neq j}\langle\mathbf{g}_t^i,\mathbf{g}_t^j\rangle\,,\quad\left\|\sum_{i=1}^{n}\mathbf{g}_t^i\right\|^2=\sum_{i,j=1}^{n}\langle\mathbf{g}_t^i,\mathbf{g}_t^j\rangle$$

which requires performing only $O(n)$ operations. We note that these estimators could also be used in algorithm 1 and the analysis would remain the same, however we decided to present it as inner products to make the presentation clearer.

Also note that the estimator of those terms (as done in Thm. 2.5) may be negative, although they should be non-negative as they estimate the norm of the gradient and the variance of the noise. To overcome this deviation, we clip the learning rate at 0, i.e. if the term is negative we define the learning rate at this iteration to be 0.

Utilizing parallel computational units. Common modern machine learning libraries (e.g. PyTorch Paszke *et al.* (2019), Tensorflow Abadi *et al.* (2016)) calculate the gradient for an entire batch of samples, instead of separately for each sample. This means, that in practice even if we are given a batch of samples, we receive only a single stochastic gradient which is an aggregation (sum or mean) of the gradient of the loss on each sample.

A common practice in modern machine learning application is using parallelized computation, where each computational node receive only a part of the batch. This is done e.g. when using TPUs (Jouppi *et al.*, 2017), which is a commonly used processing unit for training neural networks. We can model it as if we are given samples $\mathbf{x}_1,\ldots,\mathbf{x}_m$, and we have $k\ll m$ processing units, each given $\frac{m}{k}$ samples (assume for simplicity that $\frac{m}{k}$ is an integer). The processing units calculate in parallel the stochastic gradients $\mathbf{h}_1,\ldots,\mathbf{h}_k$, each w.r.t its own batch of samples, and aggregates them. To estimate the learning rate, we can use the $\mathbf{h}_i$'s in Eq. (3). Note that according to Thm. 3.1, using these gradients will have the same bound on the variance (up to a constant) than calculating the gradient w.r.t each sample separately and estimating the norm using m stochastic gradients instead of k.

Initial learning rate. Our learning rate scheduler in theory does not require an initial learning rate. However, adding exponential averaging to reduce the noise does require an initial learning rate, otherwise the first iterations (which may be very noisy) will have a very significant effect on performance of the algorithm. We add a tunable parameter η_0 which will be the initial learning rate. We emphasize that this is the only parameter in our learning rate scheduler which is being tuned.

C.1 Extension of GLyDER to Other Optimizers

We provide an extension of the GLyDER stepsize scheduler to general optimization algorithms in algorithm 3, where we focus only on the first option for smoothness estimation in Subsection 3.1, i.e. projection to a 1-dimensional function. Our assumption of the algorithm is that at each iteration t the algorithm outputs a descent direction $\mathbf{d}_t$. This descent direction might not be the gradient of the objective, although it often depend on the gradient in some manner. Typical examples include momentum SGD, where the descent direction is the gradient plus some momentum term which depends on the gradients from previous iterations. Other examples include per-parameter algorithms such as Adam or Adagrad, where the descent direction is calculated using the gradient of each parameter separately.

We additionally assume that we have access to the stochastic second derivative of the objective $f(\cdot)$ at $\mathbf{x}_{t-1}$ projected on a given direction. We now explain why this is a reasonable assumption which is applicable to practically all use-cases in supervised learning: At each iteration in supervised learning under a stochastic setting the algorithm receives a batch of labeled samples, and calculates the gradient based on the loss on those samples. In other words, the objective function is the loss on those batch of samples, and the stochastic gradient is the gradient of this loss function. Thus, calculating the second derivative of the objective $f(\cdot)$ is done in a similar way, and w.r.t the batch of samples. We emphasize that calculating the second derivative *projected* on a given direction can be done in time complexity similar to that of calculating the gradient itself.

Algorithm 3: GLyDER scheduler for general optimization algorithms

Input: $\mathbf{x}_0,\ n,\ \eta_0, \beta$
for $t = 1, 2, \ldots, T$ **do**
  Sample stochastic gradients $\mathbf{g}_t^1, \ldots, \mathbf{g}_t^n$
  Set:
    $\mu_t := \left\|\sum_{i=1}^n \mathbf{g}_t^i\right\|^2 - \sum_{i=1}^n \left\|\mathbf{g}_t^i\right\|^2$
    $\gamma_t := \left\|\sum_{i=1}^n \mathbf{g}_t^i\right\|^2$
    $\mathbf{g}_t = \sum_{i=1}^n \mathbf{g}_t^i$
  if $\gamma_t = 0$ *or* $\mu_t \leq 0$ **then**
    Set $\frac{\mu_t}{\gamma_t} := 0$
  Receive a descent direction $\mathbf{d}_t$
  estimate L_t using a projection to a 1-dimensional function, projected on the direction $\mathbf{d}_t$
  Set: $\eta_t := (1-\beta)\eta_{t-1} + \beta \cdot \frac{1}{L_t} \cdot \frac{\mu_t}{\gamma_t}$
  Update $\mathbf{x}_t = \mathbf{x}_{t-1} - \eta_t \mathbf{d}_t$

C.2 GLyDeR Stepsize Examples

In figure 3 we show the learning rate from algorithm 2 when training on the CIFAR100 dataset, for different initial learning rates. We observe a "warmup-like" behavior in the first iterations, and then a decrease where the speed and intensity of the decrease depends on the initial learning rate. This behavior might be due to the exponential averaging.

D Experimental Details

In the following section we will detail the full experimental details for all the experiments performed in the paper. We emphasize that beyond the stepsize scheduler and choice of optimization algorithm, we used the default hyper-parameter choice from the init2winit framework on all the experiments. All the parameters are chosen as to optimize the performance on each task separately.

Datasets

CIFAR10/100. We used a wide-RedNet architecture, batch size of 128, and trained for 300 epochs.

Imagenet. We used a ResNet50 architecture with batch size of 512, and trained for 100 epochs.

WikiText-2 We used an LSTM model, with an embedding dimension of 200, batch size of 32 and trained for 500000 steps.

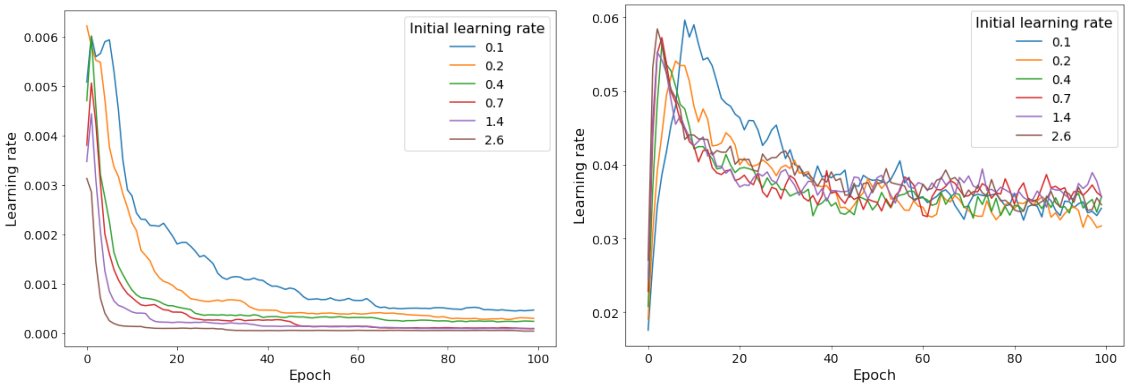


Figure 3: The greedy learning rate scheduler trained on CIFAR100 for different initial learning rates. **Left:** Smoothness estimation using projection to a 1-dimensional function (option 1). **Right:** Smoothness estimation using GNB estimation (option 2).

Criteo We used a DLRM model with and bottom MLP dimensions of $(512, 256, 128)$ and top MLP dimensions of $(1024, 1024, 512, 256, 1)$. The embedding dimension is 128, batch size of $524,288$ and trained for 1.3 epochs.

Optimizers

For momentum SGD we used a momentum parameter of 0.9. For Adam we used $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 10^{-7}$. For vanilla SGD there we no hyper-parameters beyond these of the stepsize scheduler. For all the experiments and all the schedulers we used the exact same parameters.

Hyper-parameter search

Initial stepsize: We employ a set of 20 different stepsizes, evenly distributed in a logarithmic scale spanning from 10^{-3} to 10^{2} in base 10. These 20 stepsizes were used for all the schedulers and algorithms.

Squash steps for rsqrt: We employed the following squash steps for the rsqrt scheduler: $0.5,\ 1, 5, 10, 15, 50, 100, 200, 500, 1000$.

Number of epochs: We refrained from conducting a hyper-parameter search for the number of epochs, since altering the training duration can substantially impact the impartiality of the outcome. However, we note that we used the default number of epoch from the init2winit framework, which is optimized for the default scheduler used in each specific dataset, in our cases either cosine or rsqrt. Hence, in practice for each dataset we used the number of epochs which is optimized to work on one of the schedulers which is compared to GLyDER.

E Additional Experiments

In addition to the experiments shown in Table 1, we have performed similar experiments when training with Adam and vanilla SGD. The experiments are shown in Table 2 and Table 3. The GLyDER scheduler is comparative with the other manually tuned schedulers up to a small error. Note that there is no one scheduler which outperforms all the others, e.g. cosine decay performs well on most tasks but also under-performs on a few of them (e.g. Criteo with SGD and CIFAR100 with Adam). Thus, choosing the right scheduler for the task can be seen as an extra hyper-parameter that requires tuning.

In Table 4 we provide the perplexity of the experiments done with the WikiText-2 dataset, over all the schedulers and optimization algorithms.

	GLyDER + 1– d proj	GLyDER + GNB	Constant	Cosine	rsqrt
CIFAR10 ↓	4.2% ± 0.03	4.4% ± 0.2	4.5% ± 0.3	3.0% ± 0.2	4.4% ± 0.1
CIFAR100 ↓	19.6% ± 0.2	22.6% ± 0.6	23.9% ± 0.6	18.8% ± 0.1	21.5% ± 0.2
Imagenet ↓	24.8% ± 0.5	34.9% ± 0.5	32.0% ± 0.2	23.6% ± 0.03	28.8% ± 0.3
WikiText-2 ↓	78.9% ± 0.9	79.9%±0.1$	75.9% ± 0.0	76.0% ± 0.02	75.9% ± 0.1
Criteo ↑	0.76 ± 0.003	0.68 ± 0.005	0.74 ± 0.001	0.72 ± 0.003	0.7 ± 0.003

Table 2: Similar to Table 1, except that here the experiments are done using vanilla SGD.

	GLyDER + 1– d proj	GLyDER + GNB	Constant	Cosine	rsqrt
CIFAR10 ↓	3.6% ± 0.5	4.3% ± 0.1	8.2% ± 0.5	5.3% ± 0.3	4.6% ± 0.2
CIFAR100 ↓	19.3% ± 0.4	20.5% ± 0.3	27.7% ± 0.3	21.5% ± 0.1	21.7% ± 0.5
Imagenet ↓	41.5% ± 0.5	29.6% ± 0.07	43.7% ± 0.7	29.1% ± 0.09	28.3% ± 0.2
WikiText-2 ↓	96.1% ± 2.0	77.8% ± 0.4	77.0% ± 0.02	77.3% ± 1.1	76.6% ± 0.09
Criteo ↑	0.78 ± 0.0	0.78 ± 0.03	0.78 ± 0.0	0.78 ± 0.0	0.79 ± 0.004

Table 3: Similar to Table 1, except that here the experiments are done using the Adam optimizer.

	GLyDER + 1– d proj	GLyDER + GNB	Constant	Cosine	rsqrt
SGD	245 ± 32	277 ± 2	151 ± 0.6	153. ± 0.9	152 ± 2.4
Momentum SGD	177 ± 2	172 ± 1	148 ± 0.4	149 ± 0.4	1532 ± 0.2
Adam	5659 ± 509	256 ± 21	2002 ± 48	3567 ± 72	211 ± 2.1

Table 4: Perplexity for the experiments with the WikiText-2 dataset.