# Det-CGD: Compressed Gradient Descent with Matrix Stepsizes for Non-Convex Optimization

**Anonymous Author(s)**
Affiliation
Address
email

## Abstract

This paper introduces a new method for minimizing matrix-smooth non-convex objectives through the use of novel Compressed Gradient Descent (CGD) algorithms enhanced with a matrix-valued stepsize. The proposed algorithms are theoretically analyzed first in the single-node and subsequently in the distributed settings. Our theoretical results reveal that the matrix stepsize in CGD can capture the objective's structure and lead to faster convergence compared to a scalar stepsize. As a byproduct of our general results, we emphasize the importance of selecting the compression mechanism and the matrix stepsize in a layer-wise manner, taking advantage of model structure. Moreover, we provide theoretical guarantees for free compression, by designing specific layer-wise compressors for the non-convex matrix smooth objectives. Our findings are supported with empirical evidence.

## 1 Introduction

The minimization of smooth and non-convex functions is a fundamental problem in various domains of applied mathematics. Most machine learning algorithms rely on solving optimization problems for training and inference, often with structural constraints or non-convex objectives to accurately capture the learning and prediction problems in high-dimensional or non-linear spaces. However, non-convex problems are typically NP-hard to solve, leading to the popular approach of relaxing them to convex problems and using traditional methods. Direct approaches to non-convex optimization have shown success but their convergence and properties are not well understood, making them challenging for large scale optimization. While its convex alternative has been extensively studied and is generally an easier problem, the non-convex setting is of greater practical interest often being the computational bottleneck in many applications.

In this paper, we consider the general minimization problem:

$$\min_{x\in\mathbb{R}^d} f(x), \tag{1}$$

where $f:\mathbb{R}^d\to\mathbb{R}$ is a differentiable function. In order for this problem to have a finite solution we will assume throughout the paper that $f$ is bounded from below.

**Assumption 1.** *There exists $f^{\inf}\in\mathbb{R}$ such that $f(x)\geq f^{\inf}$ for all $x\in\mathbb{R}^d$.*

The stochastic gradient descent (SGD) algorithm [MB11, B$^+$15, GLQ$^+$19] is one of the most common algorithms to solve this problem. In its most general form, it can be written as

$$x^{k+1}=x^k-\gamma g(x^k), \tag{2}$$

where $g(x^k)$ is a stochastic estimator of $\nabla f(x^k)$ and $\gamma>0$ is a positive scalar stepsize. A particular case of interest is the compressed gradient descent (CGD) algorithm [KFJ18], where the estimator $g$

Submitted to 37th Conference on Neural Information Processing Systems (NeurIPS 2023). Do not distribute.

is taken as a compressed alternative of the initial gradient:

$$g(x^k) = \mathcal{C}(\nabla f(x^k)), \tag{3}$$

and the compressor $\mathcal{C}$ is chosen to be a "sparser" estimator that aims to reduce the communication overhead in distributed or federated settings. This is crucial, as highlighted in the seminal paper by [KMY+16], which showed that the bottleneck of distributed optimization algorithms is the communication complexity. In order to deal with the limited resources of current devices, there are various compression objectives that are practical to achieve. These include also compressing the model broadcasted from server to clients for local training, and reducing the computational burden of local training. These objectives are mostly complementary, but compressing gradients has the potential for the greatest practical impact due to slower upload speeds of client connections and the benefits of averaging [KMA+21]. In this paper we will focus on this latter problem.

An important subclass of compressors are the sketches. Sketches are linear operators defined on $\mathbb{R}^d$, i.e., $\mathcal{C}(y) = \boldsymbol{S}y$ for every $y \in \mathbb{R}^d$, where $\boldsymbol{S}$ is a random matrix. A standard example of such a compressor is the Rand-$k$ compressor, which randomly chooses $k$ entries of its argument and scales them with a scalar multiplier to make the estimator unbiased. Instead of communicating all $d$ coordinates of the gradient, one communicates only a subset of size $k$, thus reducing the number of communicated bits by a factor of $d/k$. Formally, Rand-$k$ is defined as follows: $\boldsymbol{S} = \sum_{j=1}^{k} \frac{d}{k} e_{i_j} e_{i_j}^\top$, where $i_j$ are the selected coordinates of the input vector. We refer the reader to [SSR22] for an overview on compressions.

Besides the assumption that function $f$ is bounded from below, we also assume that it is $\boldsymbol{L}$ matrix smooth, as we are trying to take advantage of the entire information contained in the smoothness matrix $\boldsymbol{L}$ and the stepsize matrix $\boldsymbol{D}$.

**Assumption 2** (Matrix smoothness)**.** *There exists $\boldsymbol{L} \in \mathbb{S}_+^d$ such that*

$$f(x) \leq f(y) + \langle \nabla f(y), x - y \rangle + \frac{1}{2} \langle \boldsymbol{L}(x - y), x - y \rangle \tag{4}$$

*holds for all $x, y \in \mathbb{R}^d$.*

The assumption of matrix smoothness, which is a generalization of scalar smoothness, has been shown to be a more powerful tool for improving supervised model training. In [SHR21], the authors proposed using smoothness matrices and suggested a novel communication sparsification strategy to reduce communication complexity in distributed optimization. The technique was adapted to three distributed optimization algorithms in the convex setting, resulting in significant communication complexity savings and consistently outperforming the baselines. The results of this study demonstrate the efficacy of the matrix smoothness assumption in improving distributed optimization algorithms.

The case of block-diagonal smoothness matrices is particularly relevant in various applications, such as neural networks (NN). In this setting, each block corresponds to a layer of the network, and we characterize the smoothness with respect to nodes in the $i$-th layer by a corresponding matrix $\boldsymbol{L}_i$. Unlike in the scalar setting, we favor the similarity of certain entries of the argument over the others. This is because the information carried by the layers becomes more complex, while the nodes in the same layers are similar. This phenomenon has been observed visually in various studies, such as those by [YCN+15] and [ZCAW17].

Another motivation for using a layer-dependent stepsize has its roots in physics. In nature, the propagation speed of light in media of different densities varies due to frequency variations. Similarly, different layers in neural networks carry different information, metric systems, and scaling. Thus, the stepsizes need to be picked accordingly to achieve optimal convergence.

We study two matrix stepsized CGD-type algorithms and analyze their convergence properties for non-convex matrix-smooth functions. As mentioned earlier, we put special emphasis on the block-diagonal case. We design our sketches and stepsizes in a way that leverages this structure, and we show that in certain cases, we can achieve compression without losing in the overall communication complexity.

### 1.1 Related work

Many successful convex optimization techniques have been adapted for use in the non-convex setting. Here is a non-exhaustive list: adaptivity [DOG+19, ZKV+20], variance reduction [JRSPS16,

LBZR21], and acceleration [GNDG19]. A paper of particular importance for our work is that of [KR20], which proposes a unified scheme for analyzing stochastic gradient descent in the non-convex regime. A comprehensive overview of non-convex optimization can be found in [JK+17, DDG+22].

A classical example of a matrix stepsized method is Newton's method. This method has been popular in the optimization community for a long time [GT74, Mie80, Yam87]. However, computing the stepsize as the inverse Hessian of the current iteration results in significant computational complexity. Instead, quasi-Newton methods use an easily computable estimator to replace the inverse of the Hessian [Bro65, DM77, ABK07, ABSM14]. An example is the Newton-Star algorithm [IQR21], which we discuss in Section 2.

[GR15] analyzed sketched gradient descent by making the compressors unbiased with a sketch-and-project trick. They provided an analysis of the resulting algorithm for the linear feasibility problem. Later, [HMR18] proposed a variance-reduced version of this method.

Leveraging the layer-wise structure of neural networks has been widely studied for optimizing the training loss function. For example, [ZTJY19] propose SGD with different scalar stepsizes for each layer, [YHL+17, GCH+19] propose layer-wise normalization for Stochastic Normalized Gradient Descent, and [DBA+20, WSR22] propose layer-wise compression in the distributed setting.

DCGD, proposed by [KFJ18], has since been improved in various ways, such as in [HHH+19, LKQR20]. There is also a large body of literature on other federated learning algorithms with unbiased compressors [AGL+17, MGTR19, GBLR21, MMSR22, MSR22, HKM+23].

### 1.2 Contributions

Our paper contributes in the following ways:

- We propose two novel matrix stepsize sketch CGD algorithms in Section 2, which, to the best of our knowledge, are the first attempts to analyze a fixed matrix stepsize for non-convex optimization. We present a unified theorem in Section 3 that guarantees stationarity for minimizing matrix-smooth non-convex functions. The results shows that taking our algorithms improve on their scalar alterantives. The complexities are summarized in Table 1 for some particular cases.
- We design our algorithms' sketches and stepsize to take advantage of the layer-wise structure of neural networks, assuming that the smoothness matrix is block-diagonal. In Section 4, we prove that our algorithms achieve better convergence than classical methods.
- Assuming the that the server-to-client communication is less expensive [KMY+16, KMA+21], we propose distributed versions of our algorithms in Section 5, following the standard FL scheme, and prove weighted stationarity guarantees. Our theorem recovers the result for DCGD in the scalar case and improves it in general.
- We validate our theoretical results with experiments. The plots and framework are provided in the Appendix.

### 1.3 Preliminaries

The usual Euclidean norm on $\mathbb{R}^d$ is defined as $\|\cdot\|$. We use bold capital letters to denote matrices. By $\boldsymbol{I}_d$ we denote the $d \times d$ identity matrix, and by $\boldsymbol{O}_d$ we denote the $d \times d$ zero matrix. Let $\mathbb{S}^d_{++}$ (resp. $\mathbb{S}^d_+$) be the set of $d \times d$ symmetric positive definite (resp. semi-definite) matrices. Given $\boldsymbol{Q} \in \mathbb{S}^d_{++}$ and $x \in \mathbb{R}^d$, we write $\|x\|_{\boldsymbol{Q}} := \sqrt{\langle \boldsymbol{Q}x, x\rangle}$, where $\langle \cdot, \cdot \rangle$ is the standard Euclidean inner product on $\mathbb{R}^d$. For a matrix $\boldsymbol{A} \in \mathbb{S}^d_{++}$, we define by $\lambda_{\max}(\boldsymbol{A})$ (resp. $\lambda_{\min}(\boldsymbol{A})$) the largest (resp. smallest) eigenvalue of the matrix $\boldsymbol{A}$. Let $\boldsymbol{A}_i \in \mathbb{R}^{d_i \times d_i}$ and $d = d_1 + \ldots + d_\ell$. Then the matrix $\boldsymbol{A} = \mathrm{Diag}(\boldsymbol{A}_1, \ldots, \boldsymbol{A}_\ell)$ is defined as a block diagonal $d \times d$ matrix where the $i$-th block is equal to $\boldsymbol{A}_i$. We will use $\mathrm{diag}(\boldsymbol{A}) \in \mathbb{R}^{d \times d}$ to denote the diagonal of any matrix $\boldsymbol{A} \in \mathbb{R}^{d \times d}$. Given a function $f : \mathbb{R}^d \to \mathbb{R}$, its gradient and its Hessian at point $x \in \mathbb{R}^d$ are respectively denoted as $\nabla f(x)$ and $\nabla^2 f(x)$.

## 2 The algorithms

Below we define our two main algorithms:

$$x^{k+1} = x^k - \boldsymbol{D}\boldsymbol{S}^k \nabla f(x^k), \tag{det-CGD1}$$

and

$$x^{k+1} = x^k - \boldsymbol{T}^k \boldsymbol{D} \nabla f(x^k). \tag{det-CGD2}$$

Here, $\boldsymbol{D} \in \mathbb{S}^d_{++}$ is the fixed stepsize matrix. The sequences of random matrices $\boldsymbol{S}^k$ and $\boldsymbol{T}^k$ satisfy the next assumption.

**Assumption 3.** *We will assume that the random sketches that appear in our algorithms are i.i.d., unbiased, symmetric and positive semi-definite for each algorithm. That is*

$$\begin{aligned} &\boldsymbol{S}^k, \boldsymbol{T}^k \in \mathbb{S}^d_+, \quad \boldsymbol{S}^k \overset{iid}{\sim} \mathcal{S} \quad \textit{and} \quad \boldsymbol{T}^k \overset{iid}{\sim} \mathcal{T} \\ &\mathbb{E}\left[\boldsymbol{S}^k\right] = \mathbb{E}\left[\boldsymbol{T}^k\right] = \boldsymbol{I}_d, \quad \textit{for every} \quad k \in \mathbb{N}. \end{aligned}$$

A simple instance of det-CGD1 and det-CGD2 is the vanilla GD. Indeed, if $\boldsymbol{S}^k = \boldsymbol{T}^k = \boldsymbol{I}_d$ and $\boldsymbol{D} = \gamma \boldsymbol{I}_d$, then $x^{k+1} = x^k - \gamma \nabla f(x^k)$. In general, one may view these algorithms as Newton-type methods. In particular, our setting includes the Newton Star (NS) algorithm by [IQR21]:

$$x^{k+1} = x^k - \left(\nabla^2 f(x^{\inf})\right)^{-1} \nabla f(x^k). \tag{NS}$$

The authors prove that in the convex case it converges to the unique solution $x^{\inf}$ locally quadratically, provided certain assumptions are met. However, it is not a practical method as it requires knowledge of the Hessian at the optimal point. This method, nevertheless, hints that constant matrix stepsize can yield fast convergence guarantees. Our results allow us to choose the $\boldsymbol{D}$ depending on the smoothness matrix $\boldsymbol{L}$. The latter can be seen as a uniform upper bound on the Hessian.

The difference between det-CGD1 and det-CGD2 is the update rule. In particular, the order of the sketch and the stepsize is interchanged. When the sketch $\boldsymbol{S}$ and the stepsize $\boldsymbol{D}$ are commutative w.r.t. matrix product, the algorithms become equivalent. In general, a simple calculation shows that if we take

$$\boldsymbol{T}^k = \boldsymbol{D}\boldsymbol{S}^k\boldsymbol{D}^{-1}, \tag{5}$$

then det-CGD1 and det-CGD2 are the same. Defining $\boldsymbol{T}^k$ according to (5), we recover the unbiasedness condition:

$$\mathbb{E}\left[\boldsymbol{T}^k\right] = \boldsymbol{D}\mathbb{E}\left[\boldsymbol{S}^k\right]\boldsymbol{D}^{-1} = \boldsymbol{I}_d. \tag{6}$$

However, in general $\boldsymbol{D}\mathbb{E}\left[\boldsymbol{S}^k\right]\boldsymbol{D}^{-1}$ is not necessarily symmetric, which contradicts to Assumption 3. Thus, det-CGD1 and det-CGD2 are not equivalent for our purposes.

## 3 Main results

Before we state the main result, we present a stepsize condition for det-CGD1 and det-CGD2, respectively:

$$\mathbb{E}\left[\boldsymbol{S}^k\boldsymbol{D}\boldsymbol{L}\boldsymbol{D}\boldsymbol{S}^k\right] \preceq \boldsymbol{D}, \tag{7}$$

and

$$\mathbb{E}\left[\boldsymbol{D}\boldsymbol{T}^k\boldsymbol{L}\boldsymbol{T}^k\boldsymbol{D}\right] \preceq \boldsymbol{D}. \tag{8}$$

In the case of vanilla GD (7) and (8) become $\gamma < L^{-1}$, which is the standard condition for convergence.

Below is the main convergence theorem for both algorithms in the single-node regime.

**Theorem 1.** *Suppose that Assumptions 1-3 are satisfied. Then, for each $k \geq 0$*

$$\frac{1}{K}\sum_{k=0}^{K-1} \mathbb{E}\left[\left\|\nabla f(x^k)\right\|^2_{\boldsymbol{D}}\right] \leq \frac{2(f(x^0) - f^{\inf})}{K}, \tag{9}$$

*if one of the below conditions is true:*

*i) The vectors $x^k$ are the iterates of det-CGD1 and $\boldsymbol{D}$ satisfies (7);*

*ii) The vectors $x^k$ are the iterates of det-CGD2 and $\boldsymbol{D}$ satisfies (8).*

It is important to note that Theorem 1 yields the same convergence rate for any $\boldsymbol{D} \in \mathbb{S}^d_{++}$, despite the fact that the matrix norms on the left-hand side cannot be compared for different weight matrices. To ensure comparability of the right-hand side of (9), it is necessary to normalize the weight matrix $\boldsymbol{D}$ that is used to measure the gradient norm. We propose using determinant normalization, which involves dividing both sides of (9) by $\det(\boldsymbol{D})^{1/d}$, yielding the following:

$$\frac{1}{K}\sum_{k=0}^{K-1}\mathbb{E}\left[\left\|\nabla f(x^k)\right\|^2_{\frac{\boldsymbol{D}}{\det(\boldsymbol{D})^{1/d}}}\right] \leq \frac{2(f(x^0)-f^{\inf})}{\det(\boldsymbol{D})^{1/d}K}. \tag{10}$$

This normalization is meaningful because adjusting the weight matrix to $\frac{\boldsymbol{D}}{\det(\boldsymbol{D})^{1/d}}$ allows its determinant to be 1, making the norm on the left-hand side comparable to the standard Euclidean norm. It is important to note that the volume of the normalized ellipsoid $\left\{x \in \mathbb{R}^d \ : \ \|x\|^2_{\boldsymbol{D}/\det(\boldsymbol{D})^{1/d}} \leq 1\right\}$ does not depend on the choice of $\boldsymbol{D} \in \mathbb{S}^d_{++}$. Therefore, the results of (9) are comparable across different $\boldsymbol{D}$ in the sense that the right-hand side of (9) measures the volume of the ellipsoid containing the gradient.

### 3.1 Optimal matrix stepsize

In this section, we describe how to choose the optimal stepsize that minimizes the iteration complexity. The problem is easier for det-CGD2. We notice that (8) can be explicitly solved. Specifically, it is equivalent to

$$\boldsymbol{D} \preceq \left(\mathbb{E}\left[\boldsymbol{T}^k\boldsymbol{L}\boldsymbol{T}^k\right]\right)^{-1}. \tag{11}$$

We want to emphasize that the RHS matrix is invertible despite the sketches not being so. Indeed. The map $h : \boldsymbol{T} \to \boldsymbol{T}\boldsymbol{L}\boldsymbol{T}$ is convex on $\mathbb{S}^d_+$. Therefore, Jensen's inequality implies

$$\mathbb{E}\left[\boldsymbol{T}^k\boldsymbol{L}\boldsymbol{T}^k\right] \succeq \mathbb{E}\left[\boldsymbol{T}^k\right]\boldsymbol{L}\mathbb{E}\left[\boldsymbol{T}^k\right] = \boldsymbol{L} \succ \boldsymbol{O}_d.$$

This explicit condition on $\boldsymbol{D}$ can assist in determining the optimal stepsize. Since both $\boldsymbol{D}$ and $(\boldsymbol{T}^k\boldsymbol{L}\boldsymbol{T}^k)^{-1}$ are positive definite, then the right-hand side of (10) is minimized exactly when

$$\boldsymbol{D} = (\boldsymbol{T}^k\boldsymbol{L}\boldsymbol{T}^k)^{-1}. \tag{12}$$

The situation is different for det-CGD1. According to (10), the optimal $\boldsymbol{D}$ is defined as the solution of the following constrained optimization problem:

$$\begin{aligned} \text{minimize} \quad & \log\det(\boldsymbol{D}^{-1}) \\ \text{subject to} \quad & \mathbb{E}\left[\boldsymbol{S}^k\boldsymbol{D}\boldsymbol{L}\boldsymbol{D}\boldsymbol{S}^k\right] \preceq \boldsymbol{D} \\ & \boldsymbol{D} \in \mathbb{S}^d_{++}. \end{aligned} \tag{13}$$

**Proposition 1.** *The optimization problem (13) with respect to stepsize matrix $\boldsymbol{D} \in \mathbb{S}^d_{++}$, is a convex optimization problem with convex constraint.*

The proof of this proposition can be found in the Appendix. It is based on the reformulation of the constraint to its equivalent quadratic form inequality. Using the trace trick, we can prove that for every vector chosen in the quadratic form, it is convex. Since the intersection of convex sets is convex, we conclude the proof.

One could consider using the `CVXPY` [DB16] package to solve (13), provided that it is first transformed into a Disciplined Convex Programming (DCP) form [GBY06]. Nevertheless, (7) is not recognized as a DCP constraint in the general case. To make `CVXPY` applicable, additional steps tailored to the problem at hand must be taken.

Table 1: Summary of communication complexities of det-CGD1 and det-CGD2 with different sketches and stepsize matrices. The $\boldsymbol{D}_i$ here for det-CGD1 is $\boldsymbol{W}_i$ with the optimal scaling determined using Theorem 2, for det-CGD2 it is the optimal stepsize matrix defined in (11). The constant $2(f(x^0)-f^{\inf})/\varepsilon^2$ is hidden, $\ell$ is the number of layers, $k_i$ is the mini-batch size for the $i$-th layer if we use the rand-$k$ sketch. The notation $\tilde{\boldsymbol{L}}_{i,k}$ is defined as $\frac{d-k}{d-1}\operatorname{diag}(\boldsymbol{L}_i)+\frac{k-1}{d-1}\boldsymbol{L}_i$.

| No. | The method | $\left(\boldsymbol{S}_i^k, \boldsymbol{D}_i\right)$ | $l\geq 1, d_i, k_i, \sum_{i=1}^{\ell} k_i = k$, layer structure | $l=1, k_i=k$, general structure |
|---|---|---|---|---|
| 1. | det-CGD1 | $\left(\boldsymbol{I}_d, \gamma \boldsymbol{L}_i^{-1}\right)$ | $d\cdot \det(\boldsymbol{L})^{1/d}$ | $d\cdot \det(\boldsymbol{L})^{1/d}$ |
| 2. | det-CGD1 | $\left(\boldsymbol{I}_d, \gamma \operatorname{diag}^{-1}(\boldsymbol{L}_i)\right)$ | $d\cdot \det\left(\operatorname{diag}(\boldsymbol{L})\right)^{1/d}$ | $d\cdot \det\left(\operatorname{diag}(\boldsymbol{L})\right)^{1/d}$ |
| 3. | det-CGD1 | $\left(\boldsymbol{I}_d, \gamma \boldsymbol{I}_{d_i}\right)$ | $d\cdot \left(\prod_{i=1}^{l}\lambda_{\max}^{d_i}(\boldsymbol{L}_i)\right)^{1/d}$ | $d\cdot \lambda_{\max}(\boldsymbol{L})$ |
| 4. | det-CGD1 | $\left(\text{rand-}1, \gamma \boldsymbol{I}_{d_i}\right)$ | $\ell\cdot \left(\prod_{i=1}^{l} d_i^{d_i}\left(\max_j(\boldsymbol{L}_i)_{jj}\right)^{d_i}\right)^{1/d}$ | $d\cdot \max_j(\boldsymbol{L}_{jj})$ |
| 5. | det-CGD1 | $\left(\text{rand-}1, \gamma \boldsymbol{L}_i^{-1}\right)$ | $\ell\cdot\left(\frac{\prod_{i=1}^{l} d_i^{d_i}\lambda_{\max}^{d_i}\left(\boldsymbol{L}_i^{\frac{1}{2}}\operatorname{diag}(\boldsymbol{L}_i^{-1})\boldsymbol{L}_i^{\frac{1}{2}}\right)}{\prod_{i=1}^{l}\det(\boldsymbol{L}_i^{-1})}\right)^{1/d}$ | $\frac{d\lambda_{\max}\left(\boldsymbol{L}^{\frac{1}{2}}\operatorname{diag}\left(\boldsymbol{L}^{-1}\right)\boldsymbol{L}^{\frac{1}{2}}\right)}{\det\left(\boldsymbol{L}^{-1}\right)^{1/d}}$ |
| 6. | det-CGD1 | $\left(\text{rand-}1, \gamma \boldsymbol{L}_i^{-1/2}\right)$ | $\ell\cdot\left(\frac{\prod_{i=1}^{l} d_i^{d_i}\lambda_{\max}^{d_i}(\boldsymbol{L}_i^{1/2})}{\prod_{i=1}^{l}\det(\boldsymbol{L}_i^{-1/2})}\right)^{1/d}$ | $d\cdot \lambda_{\max}^{1/2}(\boldsymbol{L})\det(\boldsymbol{L})^{1/(2d)}$ |
| 7. | det-CGD1 | $\left(\text{rand-}1, \gamma \operatorname{diag}^{-1}(\boldsymbol{L}_i)\right)$ | $\ell\cdot\left(\frac{\prod_{i=1}^{l} d_i^{d_i}}{\prod_{j=1}^{d}(\boldsymbol{L}_{jj}^{-1})}\right)^{1/d}$ | $d\cdot \det\left(\operatorname{diag}(\boldsymbol{L})\right)^{1/d}$ |
| 8. | det-CGD1 | $\left(\text{rand-}k_i, \gamma \operatorname{diag}^{-1}(\boldsymbol{L}_i)\right)$ | $k\cdot\left(\prod_{i=1}^{l}\left(\frac{d_i}{k_i}\right)^{d_i}\det\left(\operatorname{diag}(\boldsymbol{L})\right)\right)^{1/d}$ | $d\cdot \det\left(\operatorname{diag}(\boldsymbol{L})\right)^{1/d}$ |
| 9. | det-CGD2 | $\left(\boldsymbol{I}_d, \boldsymbol{L}_i^{-1}\right)$ | $d\cdot \det(\boldsymbol{L})^{1/d}$ | $d\cdot \det(\boldsymbol{L})^{1/d}$ |
| 10. | det-CGD2 | $\left(\text{rand-}1, \frac{\operatorname{diag}^{-1}(\boldsymbol{L}_i)}{d_i}\right)$ | $\ell\cdot\left(\prod_{i=1}^{l} d_i^{d_i}\right)^{1/d}\det(\operatorname{diag}\boldsymbol{L})^{1/d}$ | $d\cdot \det(\operatorname{diag}(\boldsymbol{L}))^{1/d}$ |
| 11. | det-CGD2 | $\left(\text{rand-}k, \frac{k_i}{d_i}\tilde{\boldsymbol{L}}_{i,k_i}^{-1}\right)$ | $k\cdot\left(\prod_{i=1}^{l}\left(\frac{d_i}{k_i}\right)^{\frac{d_i}{d}}\right)\left(\prod_{i=1}^{l}\det(\tilde{\boldsymbol{L}}_{i,k_i})\right)^{1/d}$ | $d\cdot \det(\tilde{\boldsymbol{L}}_{1,k})$ |
| 12. | det-CGD2 | $\left(\text{Bern-}q_i, q_i\boldsymbol{L}_i^{-1}\right)$ | $\left(\sum_{i=1}^{l} q_i d_i\right)\cdot\prod_{i=1}^{l}\left(\frac{1}{q_i}\right)^{\frac{d_i}{d}}\det(\boldsymbol{L})^{1/d}$ | $d\cdot \det(\boldsymbol{L})^{1/d}$ |
| 13. | GD | $\left(\boldsymbol{I}_d, \lambda_{\max}^{-1}(\boldsymbol{L})\boldsymbol{I}_d\right)$ | N/A | $d\cdot \lambda_{\max}(\boldsymbol{L})$ |

# 4 Leveraging the layer-wise structure

In this section we focus on the block-diagonal case of $\boldsymbol{L}$ for both det-CGD1 and det-CGD2. In particular, we propose hyper-parameters of det-CGD1 designed specifically for training NNs. Let us assume that $\boldsymbol{L}=\operatorname{Diag}(\boldsymbol{L}_1,\ldots,\boldsymbol{L}_\ell)$, where $\boldsymbol{L}_i\in\mathbb{S}_{++}^{d_i}$. This setting is a generalization of the classical smoothness condition, as in the latter case $\boldsymbol{L}_i=L\boldsymbol{I}_{d_i}$ for all $i=1,\ldots,\ell$. Respectively, we choose both the sketches and the stepsize to be block diagonal: $\boldsymbol{D}=\operatorname{Diag}(\boldsymbol{D}_1,\ldots,\boldsymbol{D}_\ell)$ and $\boldsymbol{S}^k=\operatorname{Diag}(\boldsymbol{S}_1^k,\ldots,\boldsymbol{S}_\ell^k)$, where $\boldsymbol{D}_i,\boldsymbol{S}_i^k\in\mathbb{S}_{++}^{d_i}$.

Let us notice that the left hand side of the inequality constraint in (13) has quadratic dependence on $\boldsymbol{D}$, while the right hand side is linear. Thus, for every matrix $\boldsymbol{W}\in\mathbb{S}_{++}^d$, there exists $\gamma>0$ such that

$$\gamma^2\lambda_{\max}\left(\mathbb{E}\left[\boldsymbol{S}^k\boldsymbol{W}\boldsymbol{L}\boldsymbol{W}\boldsymbol{S}^k\right]\right)\leq\gamma\lambda_{\min}(\boldsymbol{W}).$$

Therefore, for $\gamma\boldsymbol{W}$ we deduce

$$\mathbb{E}\left[\boldsymbol{S}^k(\gamma\boldsymbol{W})\boldsymbol{L}(\gamma\boldsymbol{W})\boldsymbol{S}^k\right]\preceq\gamma^2\lambda_{\max}\left(\mathbb{E}\left[\boldsymbol{S}^k\boldsymbol{W}\boldsymbol{L}\boldsymbol{W}\boldsymbol{S}^k\right]\right)\boldsymbol{I}_d\preceq\gamma\lambda_{\min}(\boldsymbol{W})\boldsymbol{I}_d\preceq\gamma\boldsymbol{W}. \tag{14}$$

The following theorem is based on this simple fact applied to the corresponding blocks of the matrices $\boldsymbol{D},\boldsymbol{L},\boldsymbol{S}^k$ for det-CGD1.

**Theorem 2.** *Let $f:\mathbb{R}^d\to\mathbb{R}$ satisfy Assumptions 1 and 2, with $\boldsymbol{L}$ admitting the layer-separable structure $\boldsymbol{L}=\operatorname{Diag}(\boldsymbol{L}_1,\ldots,\boldsymbol{L}_\ell)$, where $\boldsymbol{L}_1,\ldots,\boldsymbol{L}_\ell\in\mathbb{S}_{++}^{d_i}$. Choose random matrices $\boldsymbol{S}_1^k,\ldots,\boldsymbol{S}_\ell^k\in\mathbb{S}_+^d$ to satisfy Assumption 3 for all $i\in[\ell]$, and let $\boldsymbol{S}^k:=\operatorname{Diag}(\boldsymbol{S}_1^k,\ldots,\boldsymbol{S}_\ell^k)$. Furthermore, choose matrices $\boldsymbol{W}_1,\ldots,\boldsymbol{W}_\ell\in\mathbb{S}_{++}^d$ and scalars $\gamma_1,\ldots,\gamma_\ell>0$ such that*

$$\gamma_i\leq\lambda_{\max}^{-1}\left(\mathbb{E}\left[\boldsymbol{W}_i^{-1/2}\boldsymbol{S}_i^k\boldsymbol{W}_i\boldsymbol{L}_i\boldsymbol{W}_i\boldsymbol{S}_i^k\boldsymbol{W}_i^{-1/2}\right]\right)\qquad\forall i\in[\ell]. \tag{15}$$

*Letting $\boldsymbol{W}:=\operatorname{Diag}(\boldsymbol{W}_1,\ldots,\boldsymbol{W}_\ell)$, $\Gamma:=\operatorname{Diag}(\gamma_1\boldsymbol{I}_{d_1},\ldots,\gamma_\ell\boldsymbol{I}_{d_\ell})$ and $\boldsymbol{D}:=\Gamma\boldsymbol{W}$, we get*

$$\frac{1}{K}\sum_{k=0}^{K-1}\mathbb{E}\left[\left\|\nabla f(x^k)\right\|^2_{\frac{\Gamma\boldsymbol{W}}{\det(\Gamma\boldsymbol{W})^{1/d}}}\right]\leq\frac{2(f(x^0)-f^{\inf})}{\det(\Gamma\boldsymbol{W})^{1/d}\,K}. \tag{16}$$

In particular, if the scalars $\{\gamma_i\}$ are chosen to be equal to their maximum allowed values from (15), then the convergence factor of (16) is equal to

$$\det\left(\Gamma \boldsymbol{W}\right)^{-\frac{1}{d}} = \left[\prod_{i=1}^{\ell} \lambda_{\max}^{d_i}\left(\mathbb{E}\left[\boldsymbol{W}_i^{-\frac{1}{2}} \boldsymbol{S}_i^k \boldsymbol{W}_i \boldsymbol{L}_i \boldsymbol{W}_i \boldsymbol{S}_i^k \boldsymbol{W}_i^{-\frac{1}{2}}\right]\right)\right]^{\frac{1}{d}} \det(\boldsymbol{W}^{-1})^{\frac{1}{d}}.$$

Table 1 contains the (expected) communication complexities of det-CGD1, det-CGD2 and GD for several choices of $\boldsymbol{W}$, $\boldsymbol{D}$ and $\boldsymbol{S}^k$. Here are a few comments about the table. We deduce that taking a matrix stepsize without compression (row 1) we improve GD (row 13). A careful analysis reveals that the result in row 5 is always worse than row 7 in terms of both communication and iteration complexity. However, the results in row 6 and row 7 are not comparable in general, meaning that neither of them is universally better. More discussion on this table can be found in the Appendix.

**Compression for free.** Now, let us focus on row 12, which corresponds to a sampling scheme where the $i$-th layer is independently selected with probability $q_i$. Mathematically, it goes as follows:

$$\boldsymbol{T}_i^k = \frac{\eta_i}{q_i} \boldsymbol{I}_{d_i}, \quad \text{where} \quad \eta_i \sim \text{Bernoulli}(q_i). \tag{17}$$

Jensen's inequality implies that

$$\left(\sum_{i=1}^{l} q_i d_i\right) \cdot \prod_{i=1}^{l} \left(\frac{1}{q_i}\right)^{\frac{d_i}{d}} \geq d. \tag{18}$$

The equality is attained when $q_i = q$ for all $i \in [\ell]$. The expected bits transferred per iteration of this algorithm is then equal to $k_{\text{exp}} = qd$ and the communication complexity equals $d \det(\boldsymbol{L})^{1/d}$. Comparing with the results for det-CGD2 with rand-$k_{\text{exp}}$ on row 11 and using the fact that $\det(\boldsymbol{L}) \leq \det\left(\text{diag}(\boldsymbol{L})\right)$, we deduce that the Bernoulli scheme is better than the uniform sampling scheme. Notice also, the communication complexity matches the one for the uncompressed det-CGD2 displayed on row 9. This, in particular means that using the Bern-$q$ sketches we can compress the gradients for free. The latter means that we reduce the number of bits broadcasted at each iteration without losing in the total communication complexity. In particular, when all the layers have the same width $d_i$, the number of broadcasted bits for each iteration is reduced by a factor of $q$.

## 5 Distributed setting

In this section we describe the distributed versions of our algorithms and present convergence guarantees for them. Let us consider an objective function that is sum decomposable:

$$f(x) := \frac{1}{n} \sum_{i=1}^{n} f_i(x),$$

where each $f_i : \mathbb{R}^d \to \mathbb{R}$ is a differentiable function. We assume that $f$ satisfies Assumption 1 and the component functions satisfy the below condition.

**Assumption 4.** *Each component function $f_i$ is $\boldsymbol{L}_i$-smooth and is bounded from below: $f_i(x) \geq f_i^{\inf}$ for all $x \in \mathbb{R}^d$.*

This assumption also implies that $f$ is of matrix smoothness with $\bar{\boldsymbol{L}} \in \mathbb{S}^d_{++}$, where $\bar{\boldsymbol{L}} = \frac{1}{n}\sum_{i=1}^n \boldsymbol{L}_i$. Following the standard FL framework [KMY+16, MMR+17, KFJ18], we assume that the $i$-th component function $f_i$ is stored on the $i$-th client. At each iteration, the clients in parallel compute and compress the local gradient $\nabla f_i$ and communicate it to the central server. The server, then aggregates the compressed gradients, computes the next iterate, and in parallel broadcasts it to the clients. See the algorithms below for the pseudo-codes.

**Algorithm 1** Distributed det-CGD1

**Input:** Starting point $x_0$, stepsize matrix $\boldsymbol{D}$, number of iterations $K$
**for** $k = 0, 1, 2, \ldots, K-1$ **do**
  The devices in parallel:
    sample $\boldsymbol{S}_i^k \sim \mathcal{S}$;
    compute $\boldsymbol{S}_i^k \nabla f_i(x_k)$;
    broadcast $\boldsymbol{S}_i^k \nabla f_i(x_k)$.
  The server:
    combines $g_k = \frac{\boldsymbol{D}}{n} \sum_i^n \boldsymbol{S}_i^k \nabla f_i(x_k)$;
    computes $x_{k+1} = x_k - g_k$;
    broadcasts $x_{k+1}$.
**end for**
**Return:** $x_K$

**Algorithm 2** Distributed det-CGD2

**Input:** Starting point $x_0$, stepsize matrix $\boldsymbol{D}$, number of iterations $K$
**for** $k = 0, 1, 2, \ldots, K-1$ **do**
  The devices in parallel:
    sample $\boldsymbol{T}_i^k \sim \mathcal{T}$;
    compute $\boldsymbol{T}_i^k \boldsymbol{D} \nabla f_i(x_k)$;
    broadcast $\boldsymbol{T}_i^k \boldsymbol{D} \nabla f_i(x_k)$.
  The server:
    combines $g_k = \frac{1}{n} \sum_i^n \boldsymbol{T}_i^k \boldsymbol{D} \nabla f_i(x_k)$;
    computes $x_{k+1} = x_k - g_k$;
    broadcasts $x_{k+1}$.
**end for**
**Return:** $x_K$

**Theorem 3.** *Let $f_i : \mathbb{R}^d \to \mathbb{R}$ satisfy Assumption 4 and let $f$ satisfy Assumption 1 and Assumption 2 with smoothness matrix $\boldsymbol{L}$. If the stepsize satisfies*

$$\boldsymbol{DLD} \preceq \boldsymbol{D}, \tag{19}$$

*then the following convergence bound is true for the iteration of Algorithm 1:*

$$\min_{0 \le k \le K-1} \mathbb{E}\left[ \left\| \nabla f(x^k) \right\|^2_{\frac{\boldsymbol{D}}{\det(\boldsymbol{D})^{1/d}}} \right] \le \frac{2(1 + \frac{\lambda_{\boldsymbol{D}}}{n})^K \left( f(x^0) - f^{\inf} \right)}{\det(\boldsymbol{D})^{1/d} K} + \frac{2\lambda_{\boldsymbol{D}} \Delta^{\inf}}{\det(\boldsymbol{D})^{1/d} n}, \tag{20}$$

*where $\Delta^{\inf} := f^{\inf} - \frac{1}{n} \sum_{i=1}^n f_i^{\inf}$ and*

$$\lambda_{\boldsymbol{D}} := \max_i \left\{ \lambda_{\max} \left( \mathbb{E}\left[ \boldsymbol{L}_i^{\frac{1}{2}} \left( \boldsymbol{S}_i^k - \boldsymbol{I}_d \right) \boldsymbol{DLD} \left( \boldsymbol{S}_i^k - \boldsymbol{I}_d \right) \boldsymbol{L}_i^{\frac{1}{2}} \right] \right) \right\}.$$

The same result is true for Algorithm 2 with a different constant $\lambda_{\boldsymbol{D}}$. The proof of Theorem 3 and its analogue for Algorithm 2 are presented in the Appendix. The analysis is largely inspired by [KR20, Theorem 1]. Now, let us examine the right-hand side of (20). We start by observing that the first term has exponential dependence in $K$. However, the term inside the brackets, $1 + \lambda_{\boldsymbol{D}}/n$, depends on the stepsize $\boldsymbol{D}$. Furthermore, it has a second-order dependence on $\boldsymbol{D}$, implying that $\lambda_{\alpha \boldsymbol{D}} = \alpha^2 \lambda_{\boldsymbol{D}}$, as opposed to $\det(\alpha \boldsymbol{D})^{1/d}$, which is linear in $\alpha$. Therefore, we can choose a small enough coefficient $\alpha$ to ensure that $\lambda_{\boldsymbol{D}}$ is of order $n/K$. This means that for a fixed number of iterations $K$, we choose the matrix stepsize to be "small enough" to guarantee that the denominator of the first term is bounded. The following corollary summarizes these arguments, and its proof can be found in the Appendix.

**Corollary 1.** *We reach an error level of $\varepsilon^2$ in (20) if the following conditions are satisfied:*

$$\boldsymbol{DLD} \preceq \boldsymbol{D}, \quad \lambda_{\boldsymbol{D}} \le \min\left\{ \frac{n}{K}, \frac{n\varepsilon^2}{4\Delta^{\inf}} \det(\boldsymbol{D})^{1/d} \right\}, \quad K \ge \frac{12(f(x^0) - f^{\inf})}{\det(\boldsymbol{D})^{1/d} \varepsilon^2}. \tag{21}$$

Proposition 2 in the Appendix proves that these conditions with respect to $\boldsymbol{D}$ are convex. In order to minimize the iteration complexity for getting $\varepsilon^2$ error, one needs to solve the following optimization problem

$$\begin{array}{ll} \text{minimize} & \log\det(\boldsymbol{D}^{-1}) \\ \text{subject to} & \boldsymbol{D} \quad \text{satisfies} \quad (21). \end{array}$$

Choosing the optimal stepsize for Algorithm 1 is analogous to solving (13). One can formulate the distributed counterpart of Theorem 2 and attempt to solve it for different sketches. Furthermore, this leads to a convex matrix minimization problem involving $\boldsymbol{D}$. We provide a formal proof of this property in the Appendix. Similar to the single-node case, computational methods can be employed using the `CVXPY` package. However, some additional effort is required to transform (21) into the disciplined convex programming (DCP) format.

The second term in (20) corresponds to the convergence neighborhood of the algorithm. It does not depend on the number of iteration, thus it remains unchanged, after we choose the stepsize.

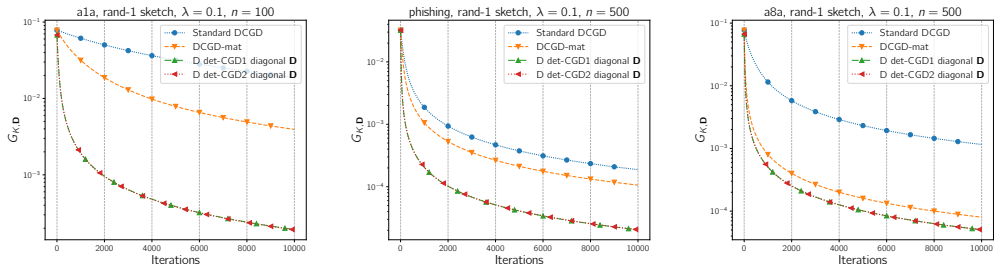


Figure 1: Comparison of standard DCGD, DCGD with matrix smoothness, D-det-CGD1 and D-det-CGD2 with optimal diagonal stepsizes under rand-1 sketch. The stepsize for standard DCGD is determined using [KR20, Proposition 4], the stepsize for DCGD with matrix smoothness along with $\boldsymbol{D}_1$, $\boldsymbol{D}_2$ is determined using Corollary 1, the error level is set to be $\varepsilon^2 = 0.0001$. Here $G_{K,\boldsymbol{D}} := \frac{1}{K}\left(\sum_{k=0}^{K-1} \left\|\nabla f(x^k)\right\|^2_{\boldsymbol{D}/\det(\boldsymbol{D})^{1/d}}\right)$.

Nevertheless, it depends on the number of clients $n$. In general, the term $\Delta^{\inf}/n$ can be unbounded, when $n \to +\infty$. However, per Corollary 1, we require $\lambda_{\boldsymbol{D}}$ to be upper-bounded by $n/K$. Thus, the neighborhood term will indeed converge to zero when $K \to +\infty$, if we choose the stepsize accordingly.

We compare our results with the existing results for DCGD. In particular we use the technique from [KR20] for the scalar smooth DCGD with scalar stepsizes. This means that the parameters of algorithms are $\boldsymbol{L}_i = L_i \boldsymbol{I}_d, \boldsymbol{L} = L\boldsymbol{I}_d, \boldsymbol{D} = \gamma \boldsymbol{I}_d, \omega = \lambda_{\max}\left(\mathbb{E}\left[\left(\boldsymbol{S}_i^k\right)^\top \boldsymbol{S}_i^k\right]\right) - 1$. One may check that (21) reduces to

$$\gamma \leq \min\left\{\frac{1}{L}, \sqrt{\frac{n}{KL_{\max}L\omega}}, \frac{n\varepsilon^2}{4\Delta^{\inf}L_{\max}L\omega}\right\} \quad \text{and} \quad K\gamma \geq \frac{12(f(x^0) - f^{\inf})}{\varepsilon^2} \tag{22}$$

As expected, this coincides with the results from [KR20, Corollary 1]. See the Appendix for the details on the analysis of [KR20]. Finally, we back up our theoretical findings with experiments. See Figure 1 for a simple experiment confirming that Algorithms 1 and 2 have better iteration and communication complexity compared to scalar stepsized DCGD. For more details on the experiments we refer the reader to the corresponding section in the Appendix.

# 6 Conclusion

## 6.1 Limitations

It is worth noting that every point in $\mathbb{R}^d$ can be enclosed within some volume 1 ellipsoid. To see this, let $0 \neq v \in \mathbb{R}^d$ and define $\boldsymbol{Q} := \frac{\alpha}{\|v\|^2} vv^\top + \beta \sum_{i=1}^d v_i v_i^\top$, where $v_1 = \frac{v}{\|v\|}, v_2, \ldots, v_d$ form an orthonormal basis. The eigenvalues of $\boldsymbol{Q}$ are $\beta$ (with multiplicity $d-1$) and $\alpha$ (with multiplicity 1), so we have $\det(\boldsymbol{Q}) = \beta^{d-1}\alpha \leq 1$. Furthermore, we have $\|v\|^2_{\boldsymbol{Q}} = v^\top \boldsymbol{Q} v = \alpha \|v\|^2$. By choosing $\alpha = \frac{1}{\|v\|^2}$ and $\beta = \|v\|^{2/(d-1)}$, we can obtain $\det(\boldsymbol{Q}) = 1$ while $\|v\|^2_{\boldsymbol{Q}} \leq 1$. Therefore, having the average $\boldsymbol{D}$-norm of the gradient bounded by a small number does not guarantee that the average Euclidean norm is small. This implies that the theory does not guarantee stationarity in the Euclidean sense.

## 6.2 Future work

Matrix stepsize gradient methods are still not well studied and require further analysis. Although many important algorithms have been proposed using scalar stepsizes and are known to have good performance, their matrix analogs have yet to be thoroughly examined. The distributed algorithms proposed in Section 5 follow the structure of DCGD by [KFJ18]. However, other federated learning mechanisms such as MARINA, which has variance reduction [GBLR21], or EF21 by [RSF21], which has powerful practical performance, should also be explored.

# References


[ABK07] Mehiddin Al-Baali and H Khalfan. An overview of some practical quasi-newton methods for unconstrained optimization. *Sultan Qaboos University Journal for Science [SQUJS]*, 12(2):199–209, 2007.

[ABSM14] Mehiddin Al-Baali, Emilio Spedicato, and Francesca Maggioni. Broyden's quasi-Newton methods for a nonlinear system of equations and unconstrained optimization: a review and open problems. *Optimization Methods and Software*, 29(5):937–954, 2014.

[AGL+17] Dan Alistarh, Demjan Grubic, Jerry Li, Ryota Tomioka, and Milan Vojnovic. QSGD: Communication-efficient SGD via gradient quantization and encoding. *Advances in neural information processing systems*, 30, 2017.

[B+15] Sébastien Bubeck et al. Convex optimization: Algorithms and complexity. *Foundations and Trends® in Machine Learning*, 8(3-4):231–357, 2015.

[Bro65] Charles G Broyden. A class of methods for solving nonlinear simultaneous equations. *Mathematics of computation*, 19(92):577–593, 1965.

[CL11] Chih-Chung Chang and Chih-Jen Lin. LIBSVM: a library for support vector machines. *ACM transactions on intelligent systems and technology (TIST)*, 2(3):1–27, 2011.

[DB16] Steven Diamond and Stephen Boyd. CVXPY: A Python-embedded modeling language for convex optimization. *The Journal of Machine Learning Research*, 17(1):2909–2913, 2016.

[DBA+20] Aritra Dutta, El Houcine Bergou, Ahmed M Abdelmoniem, Chen-Yu Ho, Atal Narayan Sahu, Marco Canini, and Panos Kalnis. On the discrepancy between the theoretical analysis and practical implementations of compressed communication for distributed deep learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 3817–3824, 2020.

[DDG+22] Marina Danilova, Pavel Dvurechensky, Alexander Gasnikov, Eduard Gorbunov, Sergey Guminov, Dmitry Kamzolov, and Innokentiy Shibaev. Recent theoretical advances in non-convex optimization. In *High-Dimensional Optimization and Probability: With a View Towards Data Science*, pages 79–163. Springer, 2022.

[DM77] John E Dennis, Jr and Jorge J Moré. Quasi-Newton methods, motivation and theory. *SIAM review*, 19(1):46–89, 1977.

[DOG+19] Darina Dvinskikh, Aleksandr Ogaltsov, Alexander Gasnikov, Pavel Dvurechensky, Alexander Tyurin, and Vladimir Spokoiny. Adaptive gradient descent for convex and non-convex stochastic optimization. *arXiv preprint arXiv:1911.08380*, 2019.

[GBLR21] Eduard Gorbunov, Konstantin P Burlachenko, Zhize Li, and Peter Richtárik. MARINA: Faster non-convex distributed learning with compression. In *International Conference on Machine Learning*, pages 3788–3798. PMLR, 2021.

[GBY06] Michael Grant, Stephen Boyd, and Yinyu Ye. Disciplined convex programming. *Global optimization: From theory to implementation*, pages 155–210, 2006.

[GCH+19] Boris Ginsburg, Patrice Castonguay, Oleksii Hrinchuk, Oleksii Kuchaiev, Vitaly Lavrukhin, Ryan Leary, Jason Li, Huyen Nguyen, Yang Zhang, and Jonathan M Cohen. Stochastic gradient methods with layer-wise adaptive moments for training of deep networks. *arXiv preprint arXiv:1905.11286*, 2019.

[GLQ+19] Robert Mansel Gower, Nicolas Loizou, Xun Qian, Alibek Sailanbayev, Egor Shulgin, and Peter Richtárik. SGD: General analysis and improved rates. In *International Conference on Machine Learning*, pages 5200–5209. PMLR, 2019.

[GNDG19] SV Guminov, Yu E Nesterov, PE Dvurechensky, and AV Gasnikov. Accelerated primal-dual gradient descent with linesearch for convex, nonconvex, and nonsmooth optimization problems. In *Doklady Mathematics*, volume 99, pages 125–128. Springer, 2019.

[GR15] Robert M Gower and Peter Richtárik. Randomized iterative methods for linear systems. *SIAM Journal on Matrix Analysis and Applications*, 36(4):1660–1690, 2015.

[GT74] William B Gragg and Richard A Tapia. Optimal error bounds for the Newton–Kantorovich theorem. *SIAM Journal on Numerical Analysis*, 11(1):10–13, 1974.

[HHH+19] Samuel Horváth, Chen-Yu Ho, Ludovit Horvath, Atal Narayan Sahu, Marco Canini, and Peter Richtárik. Natural compression for distributed deep learning. *CoRR*, abs/1905.10988, 2019.

[HKM+23] Samuel Horváth, Dmitry Kovalev, Konstantin Mishchenko, Peter Richtárik, and Sebastian Stich. Stochastic distributed learning with gradient quantization and double-variance reduction. *Optimization Methods and Software*, 38(1):91–106, 2023.

[HMR18] Filip Hanzely, Konstantin Mishchenko, and Peter Richtárik. SEGA: Variance reduction via gradient sketching. *Advances in Neural Information Processing Systems*, 31, 2018.

[IQR21] Rustem Islamov, Xun Qian, and Peter Richtárik. Distributed second order methods with fast rates and compressed communication. In *International conference on machine learning*, pages 4617–4628. PMLR, 2021.

[JK+17] Prateek Jain, Purushottam Kar, et al. Non-convex optimization for machine learning. *Foundations and Trends® in Machine Learning*, 10(3-4):142–363, 2017.

[JRSPS16] Sashank J Reddi, Suvrit Sra, Barnabas Poczos, and Alexander J Smola. Proximal stochastic methods for nonsmooth nonconvex finite-sum optimization. *Advances in neural information processing systems*, 29, 2016.

[KFJ18] Sarit Khirirat, Hamid Reza Feyzmahdavian, and Mikael Johansson. Distributed learning with compressed gradients. *arXiv preprint arXiv:1806.06573*, 2018.

[KMA+21] Peter Kairouz, H Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Kallista Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, et al. Advances and open problems in federated learning. *Foundations and Trends® in Machine Learning*, 14(1–2):1–210, 2021.

[KMY+16] Jakub Konečnỳ, H Brendan McMahan, Felix X Yu, Peter Richtárik, Ananda Theertha Suresh, and Dave Bacon. Federated learning: Strategies for improving communication efficiency. *arXiv preprint arXiv:1610.05492*, 2016.

[KR20] Ahmed Khaled and Peter Richtárik. Better theory for SGD in the nonconvex world. *arXiv preprint arXiv:2002.03329*, 2020.

[LBZR21] Zhize Li, Hongyan Bao, Xiangliang Zhang, and Peter Richtárik. PAGE: A simple and optimal probabilistic gradient estimator for nonconvex optimization. In *International conference on machine learning*, pages 6286–6295. PMLR, 2021.

[LKQR20] Zhize Li, Dmitry Kovalev, Xun Qian, and Peter Richtárik. Acceleration for compressed gradient descent in distributed and federated optimization. *arXiv preprint arXiv:2002.11364*, 2020.

[MB11] Eric Moulines and Francis Bach. Non-asymptotic analysis of stochastic approximation algorithms for machine learning. *Advances in neural information processing systems*, 24, 2011.

[MGTR19] Konstantin Mishchenko, Eduard Gorbunov, Martin Takáč, and Peter Richtárik. Distributed learning with compressed gradient differences. *arXiv preprint arXiv:1901.09269*, 2019.

[Mie80] George J Miel. Majorizing sequences and error bounds for iterative methods. *Mathematics of Computation*, 34(149):185–202, 1980.

[MMR+17] H. Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2017.

[MMSR22] Konstantin Mishchenko, Grigory Malinovsky, Sebastian Stich, and Peter Richtarik. ProxSkip: Yes! Local gradient steps provably lead to communication acceleration! Finally! In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato, editors, *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 15750–15769. PMLR, 17–23 Jul 2022.

[MSR22] Artavazd Maranjyan, Mher Safaryan, and Peter Richtárik. GradSkip: Communication-Accelerated Local Gradient Methods with Better Computational Complexity. *arXiv preprint arXiv:2210.16402*, 2022.

[RSF21] Peter Richtárik, Igor Sokolov, and Ilyas Fatkhullin. EF21: A new, simpler, theoretically better, and practically faster error feedback. *Advances in Neural Information Processing Systems*, 34:4384–4396, 2021.

[SHR21] Mher Safaryan, Filip Hanzely, and Peter Richtárik. Smoothness matrices beat smoothness constants: Better communication compression techniques for distributed optimization. *Advances in Neural Information Processing Systems*, 34:25688–25702, 2021.

[SSR22] Mher Safaryan, Egor Shulgin, and Peter Richtárik. Uncertainty principle for communication compression in distributed and federated learning and the search for an optimal compressor. *Information and Inference: A Journal of the IMA*, 11(2):557–580, 2022.

[Sti19] Sebastian U Stich. Unified optimal analysis of the (stochastic) gradient method. *arXiv preprint arXiv:1907.04232*, 2019.

[WSR22] Bokun Wang, Mher Safaryan, and Peter Richtárik. Theoretically better and numerically faster distributed optimization with smoothness-aware quantization techniques. *Advances in Neural Information Processing Systems*, 35:9841–9852, 2022.

[Yam87] Tetsuro Yamamoto. A convergence theorem for newton-like methods in banach spaces. *Numerische Mathematik*, 51:545–557, 1987.

[YCN+15] Jason Yosinski, Jeff Clune, Anh Nguyen, Thomas Fuchs, and Hod Lipson. Understanding neural networks through deep visualization. *arXiv preprint arXiv:1506.06579*, 2015.

[YHL+17] Adams Wei Yu, Lei Huang, Qihang Lin, Ruslan Salakhutdinov, and Jaime Carbonell. Block-normalized gradient method: An empirical study for training deep neural network. *arXiv preprint arXiv:1707.04822*, 2017.

[ZCAW17] Luisa M Zintgraf, Taco S Cohen, Tameem Adel, and Max Welling. Visualizing deep neural network decisions: Prediction difference analysis. *arXiv preprint arXiv:1702.04595*, 2017.

[ZKV+20] Jingzhao Zhang, Sai Praneeth Karimireddy, Andreas Veit, Seungyeon Kim, Sashank Reddi, Sanjiv Kumar, and Suvrit Sra. Why are adaptive methods good for attention models? *Advances in Neural Information Processing Systems*, 33:15383–15393, 2020.

[ZTJY19] Qinghe Zheng, Xinyu Tian, Nan Jiang, and Mingqiang Yang. Layer-wise learning based stochastic gradient descent method for the optimization of deep convolutional neural network. *Journal of Intelligent & Fuzzy Systems*, 37(4):5641–5654, 2019.