Hybrid Decentralized Optimization: Leveraging Both First- and Zeroth-Order Optimizers for Faster Convergence

Shayan Talaei[1*], **Matin Ansaripour**[2*], **Giorgi Nadiradze**[3], **Dan Alistarh**[3]
[1]Stanford University
[2]École Polytechnique Fédérale de Lausanne (EPFL)
[3]Institute of Science and Technology Austria (ISTA)
stalaei@stanford.edu, matin.ansaripour@epfl.ch, {giorgi.nadiradze, dan.alistarh}@ista.ac.at

Abstract

Distributed optimization is the standard way of speeding up machine learning training, and most of the research in the area focuses on distributed first-order, gradient-based methods. Yet, there are settings where some computationally-bounded nodes may not be able to implement *first-order, gradient-based* optimization, while they could still contribute to joint optimization tasks. In this paper, we initiate the study of hybrid decentralized optimization, studying settings where nodes with zeroth-order and first-order optimization capabilities co-exist in a distributed system, and attempt to jointly solve an optimization task over some data distribution. We essentially show that, under reasonable parameter settings, such a system can not only withstand noisier zeroth-order agents but can even benefit from integrating such agents into the optimization process, rather than ignoring their information. At the core of our approach is a new analysis of distributed optimization with noisy and possibly-biased gradient estimators, which may be of independent interest. Our results hold for both convex and non-convex objectives. Experimental results on standard optimization tasks confirm our analysis, showing that hybrid first-zeroth order optimization can be practical, even when training deep neural networks.

Code — https://github.com/ShayanTalaei/HDO
Extended version — https://arxiv.org/abs/2210.07703

Introduction

One key enabler of the extremely rapid recent progress of machine learning has been *distributed optimization*: the ability to efficiently optimize over large quantities of data, and large parameter counts, among multiple nodes or devices, in order to share the computational load, and therefore reduce end-to-end training time. Distributed machine learning has become commonplace, and it is not unusual to encounter systems which distribute model training among tens or even hundreds of nodes.

By and large, the standard distribution strategy in the context of machine learning tasks has been *data-parallel* (Bottou 2010), using *first-order* gradient estimators. We can formalize this as follows: considering a classical empirical risk minimization setting, we have a set of samples S from a distribution, and wish to minimize the function $f : \mathbb{R}^d \to \mathbb{R}$, which is the average of losses over samples from S. In other words, we wish to find $x^\star = \text{argmin}_x \sum_{s\in S} f_s(x)/|S|$. Assuming that we have n compute nodes which can process samples in parallel, data-parallel SGD consists of iterations in which each node computes gradient estimator for a batch of samples, and then nodes then exchange this information, either globally, via all-to-all communication, or *pair-wise*. Specifically, in this paper we will focus on the highly-popular *decentralized optimization case*, in which nodes interact in randomly chosen pairs, exchanging model information, following each local optimization step.

There is already a vast amount of literature on decentralized optimization in the case where nodes have access to *first-order, gradient-based* estimators. While this setting is prevalent, it does not cover the interesting case where, among the set of nodes, a fraction only have access to weaker, *zeroth-order* gradient estimators, corresponding to less computationally-capable devices, but which may still possess useful local data and computation.

In this paper, we initiate the study of *hybrid decentralized optimization* in the latter setting. Specifically, we aim to answer the following key question:

Can zeroth-order estimators be integrated in a decentralized setting, and can they boost convergence?

Roughly, we show that the answer to this question is affirmative. To arrive at it, we must overcome a number of non-trivial technical obstacles, and the answer must be qualified by key parameters, such as the first-order/zeroth-order split in the population, and the estimator variance and bias. More precisely, a key difficulty we must overcome in the algorithm and in the analysis is the fact that, under standard implementations, zeroth-order estimators are *biased*, breaking one of the key analytic assumptions in existing work on decentralized optimization, e.g. (Lian et al. 2017a; Wang and Joshi 2021; Koloskova et al. 2020a,b; Nadiradze et al. 2021).

Our analysis approach overcomes this obstacle and provides the first convergence bounds for hybrid decentralized optimization via a novel potential argument. Roughly, assuming a d-dimensional and L-smooth finite-sum objective function f, and a population of n nodes, in which n_1 have first-order stochastic gradient estimators of variance σ_1, and

*These authors contributed equally.

n_0 have zeroth-order estimators of variance σ_0, then our analysis shows that the "stochastic noise" in the convergence of our hybrid decentralized optimization algorithm in this population is given, up to constants, by the following three quantities:

$$\frac{\eta(dn_0\varsigma_0^2 + n_1\varsigma_1^2)}{n^2}, \frac{\eta(dn_0\sigma_0^2 + n_1\sigma_1^2)}{n^2}, \eta^2\big(\frac{Ldn_0}{n}\big)^k. \quad (1)$$

In this expression, η is the learning rate, and the quantities ς_1 and ς_0 are bounds on the *average* variance of first-order and zeroth-order estimators at the nodes, respectively, given by the way in which the data is split among these two types of agents. Intuitively, the first term is the variance due to the (random) data split, whereas the second term is the added variance due to noise in the two types of gradient estimators. (The zeroth-order terms are scaled by the dimension, as is common in this case.) The third term bounds *the bias* induced by the zeroth-order gradient estimators, where k equals 1 for the convex case and 2 for the non-convex case. Using this characterization, we show that there exist reasonable parameter settings such that, if zeroth-order nodes do not have extremely high variance, they may in fact be useful for convergence, especially since the third bias term can be controlled via the learning rate η.

Our analysis approach should be of independent interest: first, we provide a simple and general way of characterizing convergence in a population mixing first- and zeroth-order agents, which can be easily parametrized given population and estimator properties, for both convex and non-convex objectives. (For instance, we can directly cover the case when the zeroth-order estimators are unbiased (Chen 2020) as in this case the bias term becomes zero.) Second, we do so in a very general communication model which allows agents to interact at different rates (due to randomness), covering both the pair-wise interaction model (Angluin et al. 2006; Nadiradze et al. 2021) and the global matching interactions model (Lian et al. 2017a; Wang and Joshi 2021; Koloskova et al. 2020a,b).

A key remaining question is whether the above characterization can be validated for practical setting. For this, we implemented our algorithm and examined the convergence under various optimization tasks, population relative sizes, and estimator implementations. Specifically, we implemented three different types of zeroth-order estimators: a standard biased one, e.g. (Nesterov and Spokoiny 2017), a de-biased estimator (Chen 2020), and the novel gradient-free estimator of (Baydin et al. 2022), and examined their behavior when mixed with first-order estimators. In brief, our results show that, even for high-dimensional and complex tasks, such as fine-tuning the ResNet-18 (He et al. 2015), or a Transformer model (Vaswani et al. 2023), our approach continues to converge. Importantly, we observe that our approach allows a system to incorporate information from the zeroth-order agents in an efficient and robust, showing higher convergence speed relative to the case where only first-order information is considered for optimization.

Related Work. The study of decentralized optimization algorithms dates back to Tsitsiklis (1984), and is related to the study of *gossip* algorithms for information dissemination (Kempe, Dobra, and Gehrke 2003; Xiao and Boyd 2004). The distinguishing feature of this setting is that optimization occurs jointly, but in the absence of a coordinator node. Several classic first-order algorithms have been ported and analyzed in the gossip setting, such as subgradient methods for convex objectives (Nedic and Ozdaglar 2009; Johansson, Rabi, and Johansson 2009; Shamir and Srebro 2014) or ADMM (Wei and Ozdaglar 2012; Iutzeler et al. 2013). References (Lian et al. 2017a,b; Assran et al. 2018) consider SGD-type algorithms in the non-convex setting, while references (Tang et al. 2018; Koloskova et al. 2020a; Nadiradze et al. 2021) analyzed the use of quantization in the gossip setting. By contrast, zeroth-order optimization has been relatively less investigated: Sahu and Kar (2020) proposes a distributed deterministic zeroth-order Frank-Wolfe-type algorithm, whereas other works by (Yuan et al. 2024) and (Mhanna and Assaad 2023) investigated the rates which can be achieved by decentralized zeroth-order algorithms, proposing multi-stage methods which can match the rate of centralized algorithms in some parameter regimes. Relative to the latter reference, we focus on simpler decentralized algorithms, which can easily interface with first-order optimizers, and perform a significantly more in-depth experimental validation.

Stochastic zeroth-order optimization has been classically applied for gradient-free optimization of convex functions, e.g. (Nesterov and Spokoiny 2017), and has been extended to tackling high-dimensionality and saddle-point constraints, e.g. (Balasubramanian and Ghadimi 2022). (The area has tight connections to bandit online optimization, under time-varying objective functions, e.g. (Flaxman, Kalai, and McMahan 2004; Agarwal, Dekel, and Xiao 2010; Shamir 2017); however, our results are not immediately relevant to this direction, as we are interested in interactions with agents possessing first-order information as well.) In this paper, we also investigate improved single-point function evaluation for better gradient estimation (Jongeneel, Yue, and Kuhn 2021) as well as the forward-mode unbiased estimator of Baydin et al. (2022).

Preliminaries

The System Model

We consider a standard model for the decentralized optimization setting, which is similar to (Koloskova et al. 2020a,b; Lian et al. 2017a; Nadiradze et al. 2021). Specifically, we have $n \geq 2$ agents, of which n_0 agents have zeroth-order gradient oracles, and n_1 have first-order gradient oracles. (We describe the exact optimization setup in the next section.) Beyond their oracle type, the agents are assumed to be anonymous for the purposes of the protocol. The execution will proceed in discrete *steps*, or *rounds*, where in each step, two agents are chosen to interact, uniformly at random. Specifically, when chosen, each agent performs some local computation, e.g. obtains some gradient information from their local oracle. Then, the two agents exchange parameter information, and update their local models, after which they are ready to proceed to the next round. Notice that

this random interaction model is asynchronous, in the sense that the number of interactions taken by agents up to some point in time may be different, due to randomness. The basic unit of time used in the analysis, which we call *fine-grained time*, will be the total number of interactions among agents up to some given point in the execution. To express global progress, we will consider *parallel time*, which is the *average* number of interactions up to some point, and can be obtained by dividing by n the total number of interactions. This corresponds to the intuition that $\Theta(n)$ interactions may occur in parallel. In experiments, we will examine the convergence of the local model at a fixed node.

This model is an instantiation of the classic *population model* of distributed computing (Angluin et al. 2006), in an optimization setting. The model is similar to the one adopted by Nadiradze et al. (2021) for analyzing asynchronous decentralized SGD, and is more general than the ones adopted by Koloskova et al. (2020a,b); Lian et al. (2017a); Wang and Joshi (2021) for decentralized analysis, since the latter assume that nodes are paired via perfect global random matchings in each round. (Our analysis would easily extend to global matching, yielding virtually the same results.)

Optimization Setup

We assume each node i has a local data distribution $\mathcal{D}^i$, and that the loss function corresponding to the samples at node i, denoted by $f^i(x) : \mathbb{R}^d \to \mathbb{R}$ can be approximated using its stochastic form $F^i(x, \xi^i)$ for each parameter $x \in \mathbb{R}^d$ and (randomly chosen) sample $\xi^i \sim \mathcal{D}^i$, where $f^i(x) = \mathbb{E}_{\xi^i \sim \mathcal{D}^i}\left[F^i(x, \xi^i)\right]$. For simplicity of notation, we assume that nodes in the set $N_0 = \{1, 2, ..., n_0\}$ are zeroth-order nodes and the nodes in the set $N_1 = [n]/N_0$ are first-order nodes. Let n_0 and n_1 be the sizes of the sets N_0 and N_1 correspondingly.

In this setup nodes communicate to solve a distributed stochastic optimization problem, i.e.

$$f^* = \min_{x \in \mathbb{R}^d}\Big[f(x) := \frac{1}{n_0}\sum_{i \in N_0} f^i(x) + \frac{1}{n_1}\sum_{i \in N_1} f^i(x)\Big].$$

This means that we wish to optimize the function f which corresponds to the loss over all data samples. Since in the analysis we will wish to throttle the ratio of zeroth-order to first-order agents, we split the entire data among zeroth-order nodes, and we do the same thing for the first-order nodes. (Our analysis can be extended to settings where this is not the case, but this will allow us for instance to study what happens when either n_0 or n_1 goes to zero, without changing our objective function.) We make the following assumptions on the optimization objectives:

Assumption 1 (Strong convexity)**.** *We assume that the function f is strongly convex with parameter $\ell > 0$, i.e. for all $x, y \in \mathbb{R}^d$:*

$$(x - y)^T(\nabla f(x) - \nabla f(y)) \geq \ell\|x - y\|^2.$$

Assumption 2 (Smooth gradient)**.** *All the stochastic gradients ∇F^i are L-Lipschitz for some constant $L > 0$, i.e. for all $\xi^i \sim \mathcal{D}^i$ and $x, y \in \mathbb{R}^d$:*

$$\|\nabla F^i(x, \xi^i) - \nabla F^i(y, \xi^i)\| \leq L\|x - y\|. \tag{2}$$

If in addition F^i are convex functions, then

$$\|\nabla F^i(x, \xi^i) - \nabla F^i(y, \xi^i)\| \leq 2L(F^i(x, \xi^i)) - F^i(y,\xi^i) - \langle x - y, \nabla F^i(y, \xi^i)\rangle).$$

Using Assumption 2, one can easily find that the gradients of f and $f^i(x)\ \forall i \in [n]$ are also satisfying the above inequalities. Further, we make the following assumptions about the data split and the stochastic gradient estimators:

Assumption 3 (Balanced data distribution)**.** *The average variance of $\nabla f^i(x)$s for both zero and first order nodes is bounded by a global constant values, i.e. for all $x \in \mathbb{R}^d$:*

$$\frac{1}{n_0}\sum_{i \in N_0}\|\nabla f^i(x) - \nabla f(x)\|^2 \leq \varsigma_0^2;$$
$$\frac{1}{n_1}\sum_{i \in N_1}\|\nabla f^i(x) - \nabla f(x)\|^2 \leq \varsigma_1^2.$$

Assumption 4 (Unbiasedness and bounded variance)**.** *For each i, $\nabla F^i(x, \xi^i)$ is an unbiased estimator of $\nabla f^i(x)$ and its variance is bounded by a constant s_i^2, i.e. for all $x \in \mathbb{R}^d$:*

$$\mathbb{E}_{\xi^i \sim \mathcal{D}^i}[\nabla F^i(x, \xi^i)] = \nabla f^i(x);$$
$$\mathbb{E}_{\xi^i}\|\nabla F^i(x, \xi^i) - \nabla f^i(x)\| \leq s_i^2.$$

Each node has access to an estimator $G^i(x)$ that estimates the local gradient $\nabla f^i(x)$ at point x. For nodes which can perform the gradient computation over a batch of data, i.e. first-order nodes, $G^i(x)$ is $\nabla F^i(x, \xi^i)$, where $\xi^i \sim \mathcal{D}^i$.

Definition 1. *We define the average of s_i for the zeroth and first order populations as σ_0^2 and σ_1^2 respectively. Formally, we define*

$$\sigma_0^2 := \frac{1}{n_0}\sum_{i \in n_0} s_i^2, \quad \sigma_1^2 := \frac{1}{n_1}\sum_{i \in n_1} s_i^2.$$

Zeroth-order Optimization

We now provide a brief introduction relative to standard basic facts and assumptions concerning zeroth-order optimization. Let the function $f_\nu^i(x) := \mathbb{E}_u[f^i(x + \nu u)]$, $u \sim N(0, I_d)$ be the smoothed version of each function $f^i(x)$. Then, node i can estimate the gradient of f_ν^i by only evaluating some points of f^i.

Definition 2 (Zeroth-order estimator)**.**

$$G_\nu^i(x, u, \xi^i) = \frac{F^i(x + \nu u, \xi^i) - F^i(x, \xi^i)}{\nu}u, \tag{3}$$

where $u \sim N(0, I_d)$ and $\xi^i \sim \mathcal{D}^i$.

Note that under Assumption 4, one can easily prove that $G_\nu^i(x, u, \xi^i)$ is an unbiased estimator of ∇f_ν^i since

$$\mathbb{E}_{u,\xi^i}[G_\nu^i(x, u, \xi^i)] = \mathbb{E}_u[\frac{f^i(x + \nu u) - f^i(x)}{\nu}u] = \nabla f_\nu^i(x). \tag{4}$$

As a technical note, in our analysis we will set $\nu := \frac{\eta}{c}$, where η is the learning rate and c is a constant to be defined later. Therefore, for simplicity we can define $G^i(x) :=$

$G^i_\nu(x,u,\xi^i)$, where $G^i_\nu(x,u,\xi)$ is as defined in Definition 2 and $\nu = \frac{\eta}{c}$. Since zeroth-order nodes cannot perform gradient computation directly, we use this $G^i(x)$ as their gradient estimator. We restate the following well-known fact:

Lemma 1 ((Nesterov and Spokoiny 2017), Theorem 1.1 in (Balasubramanian and Ghadimi 2022))**.** *For a Gaussian random vector $u \sim N(0, I_d)$ we have that*

$$\mathbb{E}[\|u\|^k] \leq (d+k)^{k/2} \tag{5}$$

for any $k \geq 2$. Moreover, the following statements hold for any function f whose gradient is Lipschitz continuous with constant L.

a) The gradient of f_ν is Lipschitz continuous with constant L_ν such that $L_\nu \leq L$.

b) For any $x \in \mathbb{R}^d$,

$$|f_\nu(x) - f(x)| \leq \frac{\nu^2}{2} Ld,$$
$$\|\nabla f_\nu(x) - \nabla f^i(x)\| \leq \frac{\nu}{2} L(d+3)^{\frac{3}{2}}.$$

c) For any $x \in \mathbb{R}^n$,

$$\frac{1}{\nu^2}\,\mathbb{E}_u[\{f(x+\nu u) - f(x)\}^2 \|u\|^2] \leq$$
$$\frac{\nu^2}{2} L^2 (d+6)^3 + 2(d+4)\|\nabla f(x)\|^2.$$

The HDO Algorithm

Algorithm Description. We now describe a decentralized optimization algorithm, designed to be executed by a population of n nodes, interacting in pairs chosen uniformly at random as per our model. We assume that n_1 of the nodes have access to first-order estimators and n_0 of them have access to zeroth-order estimators, hence $n = n_1 + n_0$. Two copies of the training data are distributed, once among the first-orders and once among the zeroth-orders. Thus, each first- and zeroth-order node has access to $\frac{1}{n_1}$, $\frac{1}{n_0}$ of the entire training data, respectively. We assume that each node i has access to a local stochastic estimator of the gradient, which we denote by G^i, and maintains a model estimate X^i, as well as the global learning rate η. Without loss of generality, we assume that the models are initialized to the same randomly-chosen point. Specifically, upon every interaction, the interacting agents i and j perform the following steps:

Algorithm 1: HDO pseudocode for each interaction between randomy chosen nodes i and j

```
// Nodes perform local steps.
X^i ← X^i − ηG^i(X^i);
X^j ← X^j − ηG^j(X^j);
// Nodes average their local models.
avg ← (X^i + X^j)/2;
X^i ← avg;
X^j ← avg;
```

In a nutshell, upon each interaction, each node first performs a local model update based on its estimator, and then nodes average their local models following the interaction. We do not distinguish between estimator types in this interaction. The nodes are then ready to proceed to the next round.

The Convergence of the HDO Algorithm

This section is dedicated to proving that the following result

Theorem 1. *Assume an objective function $f : \mathbb{R}^d \to \mathbb{R}$, equal to the average loss over all data samples, whose optimum x^* we are trying to find using Algorithm 1. Let n_0 be the number of zeroth-order nodes, and n_1 be the number of first-order agents. Given the data split described in the previous section, let f_i be the local objective function of node i. Assume that zeroth-order nodes use estimators with $\nu = \frac{\eta}{\sqrt{d}}$. Let the total number of steps in the algorithm T and $\mu_t = \sum_{i=1}^n X_t^i/n$, then we can derive the following convergence rates.*

Non-Convex: *Under assumptions 2, 3 and 4, and letting T be large enough such that $T = \Omega\left(\max\left\{\frac{L^2(dn_0+n_1)^2}{dn^2}, n^2L^2, nn_0^3 d\right\}\right)$, we have*

$$\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\,\|\nabla f(\mu_t)\|^2 = \sqrt{\frac{d}{T}} \times O\Bigg((f(\mu_0) - f^*) +$$
$$L\Big(\frac{dn_0\varsigma_0^2 + n_1\varsigma_1^2}{nd}\Big) + L\Big(\frac{dn_0\sigma_0^2 + n_1\sigma_1^2}{nd}\Big) + L^2\sqrt{\frac{n_0}{n}}\Bigg).$$

Adding, convexity (Assumption 1), we prove the following:

Strongly Convex: *If we assume that the functions f and f_i satisfy Assumptions 1, 2, 3 and 4, and let T be large enough such that $\frac{T}{\log T} = \Omega\left(\frac{n(d+n)(L+1)\left(\frac{1}{\ell}+1\right)}{\ell}\right)$, and let the learning rate be $\eta = \frac{4n\log T}{T\ell}$. For $1 \leq t \leq T$, let the sequence of weights w_t be given by $w_t = \left(1 - \frac{\eta\ell}{2n}\right)^{-t}$ and let $S_T = \sum_{t=1}^T w_T$. Finally, define $y_T = \sum_{t=1}^T \frac{w_t\mu_{t-1}}{S_T}$ to be the mean over local model parameters. Then, we can show that HDO provides the following convergence rate:*

$$\mathbb{E}[f(y_T) - f(x^*)] + \frac{\ell\,\mathbb{E}\,\|\mu_T - x^*\|^2}{8}$$
$$= O\Bigg(\frac{L\|\mu_0 - x^*\|^2}{T\log T} + \frac{\log(T)(dn_0\varsigma_0^2 + n_1\varsigma_1^2)}{T\ell n}$$
$$+ \frac{\log(T)(dn_0\sigma_0^2 + n_1\sigma_1^2)}{T\ell n} + \frac{\log(T)dn_0}{T\ell n}\Bigg).$$

Speedup. Here, the time T refers to the *total number of interactions* among agents, as opposed to *parallel time*, corresponding to the average number of interactions T/n. These rates are reminiscent of sequential SGD. However, there are some distinctions: we are counting the total number of gradient oracle queries by the nodes, and there are some additional trailing terms, whose meanings we discuss below.

We interpret this formula from the perspective of an arbitrary local model. For this, notice that the notion of *parallel time* corresponding to the number of total interactions T, which is by definition $T_p = T/n$, corresponds (up to constants) to the *average* number of interactions and gradient oracle queries performed by each node up to time T. Therefore, in the strongly convex setup, for any single model, convergence with respect to its number of performed SGD steps T_p would be $O(\log(nT_p)/(nT_p))$ (assuming all parameters are constant), which would correspond to $\Omega(\frac{n}{\log(nT_p)}) = \Omega(\frac{n}{\log(T)})$ speedup compared to a variant of sequential SGD. Notice that this is quite favorable to our algorithm, since we are considering *biased* zeroth-order estimators for some of the nodes in the population. Hence, assuming that T is polynomial in n, we get an almost-linear speedup of $\Omega\Big(\frac{n}{\log(n)}\Big)$. Similarly, in the non-convex case, we get a speedup $\Omega(\sqrt{n})$, which shows the scalability of our algorithm.

Impact of Zeroth-Order Nodes. Notice that our convergence bounds cleanly separate in the terms which come from zeroth-order nodes and terms which come from first-order nodes. For $n_0 = 0$, we get asymptotically the same bound as we would get if all nodes performed pure first-order SGD steps. Similarly, when $n_0 = n$ we should be able to achieve asymptotically-optimal convergence for biased zeroth-order estimators. Further, notice that, if the bias is negligible, then the last term in each of the upper bounds disappears, and we obtain a trade-off between two populations with different variances. We can also observe the following theoretical threshold: we asymptotically match the convergence rate in the case with all nodes performing SGD steps, as long as $dn_0 = O(n)$ (assuming all other parameters are constant).

Analysis

As an example, we discuss the proof overview for the strongly convex case. The notations and proof steps are closely aligned with those used in the non-convex case.

Proof Overview. The convergence proof, given in full in the Appendix, can be split conceptually into two steps. The first aims to bound the variance of the local models X_t^i for each time step t and node i with respect to the mean $\mu_t = \sum_i X_t^i$. It views this variance as a potential Γ_t, which as we show has supermartingale-like behavior for small enough learning rate: specifically, this quantity tends to increase due to gradient steps, but is pushed towards the mean μ_t by the averaging process.

The key component here is Lemma 2, which carefully bounds the evolution of the potential at a step, by modeling optimization as a dynamic load balancing process: each interaction corresponds to a *weight generation* step (in which gradient estimators are generated) and a *load balancing step*, in which the "loads" of the two nodes (corresponding to their model values) are balanced through averaging.

In the second step, we first bound the rate at which the mean μ_t converges towards x^*, where we crucially (and carefully) leverage the variance bound obtained above. The main challenge in this part is dealing with biased zeroth-order estimators. In fact, even dealing with biased first-order estimators is not trivial, since for example, they are the main reason for the usage of error feedback when stochastic gradients are compressed using biased quantization (Alistarh et al. 2018). This is our second key technical result.

With this in hand, we can complete the proof by applying a standard argument which characterizes the rate at which $\mathbb{E}[f(y_T) - f(x^*)]$ and $\mathbb{E}[\|\mu_t - x^*\|^2$ converge towards 0.

Notation and Preliminaries. In this section, we provide a more in-depth sketch of the analysis of the HDO protocol. We begin with some notation. Recall that n is the number of nodes, split into first-order (n_1) and zeroth-order (n_0). We will analyze a sequence of *time steps* $t = 1, 2, \ldots, T$, each corresponding to an individual interaction between two nodes, which are usually denoted by i and j.

Step 1: Parameter Concentration. Next, let X_t be a vector of model estimates at time step t, that is $X_t = (X_t^1, X_t^2, ..., X_t^n)$. Also, let $\mu_t = \frac{1}{n}\sum_{i=1}^{n} X_t^i$, be an average estimate at time step t. The following potential function measures the variance of the models:

$$\Gamma_t = \frac{1}{n}\sum_{i=1}^{n} \|X_t^i - \mu_t\|^2.$$

With this in place, one of our key technical results is to provide a supermartingale-type bound on the evolution of the potential Γ_t, in terms η, and average second moment of estimators at step t, defined as $M_t^G := \frac{1}{n}\sum_i \left\|G^i(X_t^i)\right\|^2$.

Lemma 2. *For any time step t :*

$$\mathbb{E}\left[\Gamma_{t+1}\right] \leq \left(1 - \frac{1}{2n}\right)\mathbb{E}\left[\Gamma_t\right] + \frac{4}{n}\eta^2\,\mathbb{E}\left[M_t^G\right].$$

Notice that, if we had a universal second moment bound on the estimators, that is, for any vector X and node i $\mathbb{E}\left\|G^i(X)\right\|^2 \leq M$, for some $M > 0$, then we would be able to unroll the recursion, and, for any $t \geq 0$ upper bound $E[\Gamma_t]$ by $\eta^2 M^2$. In the absence of such upper bound we must derive the following upper bound on $\mathbb{E}\left[M_t^G\right]$:

Lemma 3. *Assume $\nu := \frac{\eta}{c}$ is fixed, where η and c are the learning rate and a constant respectively. Then, for any time step t we have:*

$$\begin{aligned}\mathbb{E}\left[M_t^G\right] \leq\ & 6(d+4)L^2\,\mathbb{E}[\Gamma_t] + \frac{6(d+4)n_0\varsigma_0^2 + 3n_1\varsigma_1^2}{n} \\ & + 6(2d+9)L\,\mathbb{E}[f(\mu_t) - f(x^*)] \\ & + \frac{2(d+4)\sum_{i\in N_0} s_i^2 + \sum_{i\in N_1} s_i^2}{n} \\ & + \eta^2\frac{n_0}{2nc^2}L^2(d+6)^3.\end{aligned}$$

First, we check how this upper bound affects the upper bound given by Lemma 2. For small enough η, the term containing $\mathbb{E}[\Gamma_t]$ (which comes from the upper bound on $\mathbb{E}\left[M_t^G\right]$) can be upper bounded by $\frac{1}{4n}\mathbb{E}[\Gamma_t]$, and hence it will just change the factor in front of $\mathbb{E}[\Gamma_t]$ to $(1 - 1/4n)$.

Second, since the above bound contains the term with $\mathbb{E}[f(\mu_t) - f(x^*)]$ we are not able to bound the potential

Γ per step, instead, for weights $w_t = (1 - \frac{\eta\ell}{2n})^{-t}$, we can upper bound $\sum_{t=1}^{T} w_t \, \mathbb{E}[\Gamma_{t-1}]$ (please see Lemma **??** in the Appendix). The crucial property is that the upper bound on the weighted sum $\sum_{t=1}^{T} w_t \, \mathbb{E}[\Gamma_{t-1}]$, is

$$O(\eta^2 \sum_{t=1}^{T} w_t \, \mathbb{E}[f(\mu_{t-1}) - f(x^*)]) + \sum_{t=1}^{T} w_t O(\eta^2).$$

(for simplicity, above we assumed that all other parameters are constant.)

Step 2: Convergence of the Mean and Risk Bound. The above result allows us to characterize how well the individual parameters are concentrated around their mean. In turn, this will allow us to provide a recurrence for how fast the parameter average is moving towards the optimum. To help with the intuition, we provide the lemma which is simplified version of the one given in the additional material:

Lemma 4. *For small enough η and $t \geq 1$ we have that:*

$$\begin{aligned}\mathbb{E}\left\|\mu_t - x^*\right\|^2 \leq (1 - \frac{\ell\eta}{2n}) \, \mathbb{E}\left\|\mu_{t-1} - x^*\right\|^2 \\ - \Omega(\frac{\eta}{n}) \, \mathbb{E}\left[f(\mu_{t-1}) - f(x^*)\right] \\ + O\left(\frac{\eta}{n}\right) \mathbb{E}[\Gamma_{t-1}] + O(\frac{\eta^2}{n^2}).\end{aligned}$$

Note that O and Ω hide all other parameters (we assume that all other parameters are constant). As mentioned, *the main challenge in the proof of this lemma is taking care of biased zeroth-order estimators*.

Recall that $w_t = (1 - \frac{\eta\ell}{2n})^{-t}$, by definition. We proceed by multiplying both sides of the above inequality by w_t and then summing it up for $1 \leq t \leq T$. Then, once we plug the upper bound on $\sum_{t=1}^{T} w_t \, \mathbb{E}[\Gamma_{t-1}]$, for small enough η the term $O(\frac{\eta}{n})O(\eta^2 \sum_{t=1}^{T} w_t \, \mathbb{E}[f(\mu_{t-1} - f(x^*)])$ vanishes as it is dominated by the term $-\sum_{t=1}^{T} \Omega(\frac{\eta}{n}) \, \mathbb{E}\left[f(\mu_{t-1}) - f(x^*)\right]$.

We get the final convergence bound after some simple calculations involving division of both sides by $S_T = \sum_{t=1}^{T} w_t$, and using $\eta = \frac{4nlog(T)}{T\ell}$ together with the upper bound on T (in turn, this makes sure that η is small enough, so that all upper bounds we mentioned hold).

Experimental Results

Experimental Setup and Goals. In this section, we validate our results by simulating the HDO algorithm under different conditions, including varying the number of nodes, the ratios between first-order (FO) and zeroth-order (ZO) nodes, and the "strength" of the zeroth-order gradient estimators. Our focus is on the algorithm's convergence behavior, relative to the total number of optimization steps, which we measure by tracking the loss over time or the accuracy on the hidden validation set. Our goal is to determine whether a hybrid system, combining both FO and ZO agents, can achieve better convergence compared to a system that relies on a single type of agent. Additionally, we aim to demonstrate the convergence speed, in terms of training loss at a fixed node, for different sizes of *mono-type populations*, each consisting of only one type of estimator.

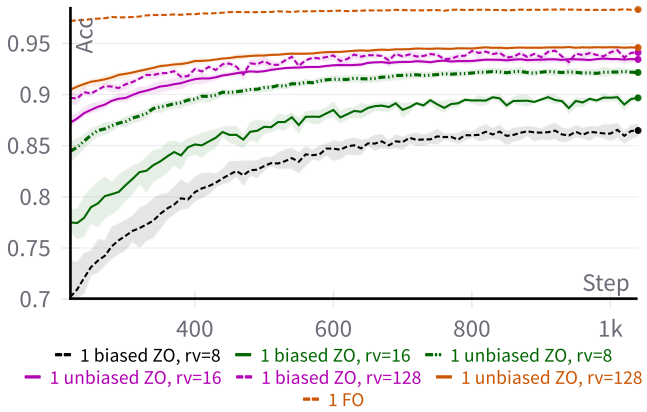

Figure 1: Number of random vectors (rv) impact on the biased/unbiased ZO estimators Accuracy (Acc), using a CNN model on MNIST.

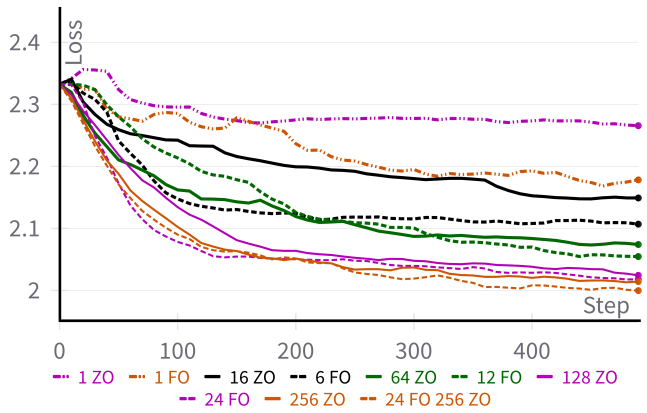

Figure 2: Validation loss vs. various population configurations for regression model on MNIST.

In the implementation of HDO, each step involves nodes updating their models, followed by the formation of $O(n)$ random disjoint pairs. Each pair exchanges their models and replaces their own with the averaged model. To demonstrate the effectiveness of zeroth-order nodes under reasonable parameter settings, we evaluate the mean validation loss and accuracy across all nodes and analyze the consensus of the models by measuring the standard deviation of losses. Additional details on the models, datasets, and the complete experimental setup are provided in the Appendix.

Results. At first, we examine the performance of individual zeroth-order gradient estimators (with $\nu = 10^{-4}$) over time, as a function of the number of random vectors (rv) used for the gradient estimation (Figure 1); that is the number of u's used to estimate the gradient using the equation 4. To do so, we use the MNIST classification task (Deng 2012) using a Convolutional Neural Network (CNN) (Lecun et al. 1998) model. We choose values 8, 16, and 128 for the number of random vectors, and compare against the unbi-

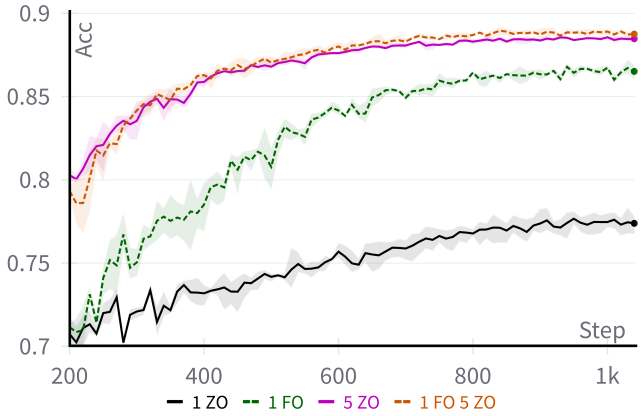

Figure 3: Validation accuracy (Acc) comparison between the hybrid and mono-type estimator population for ResNet-18 on the CIFAR-10 dataset.

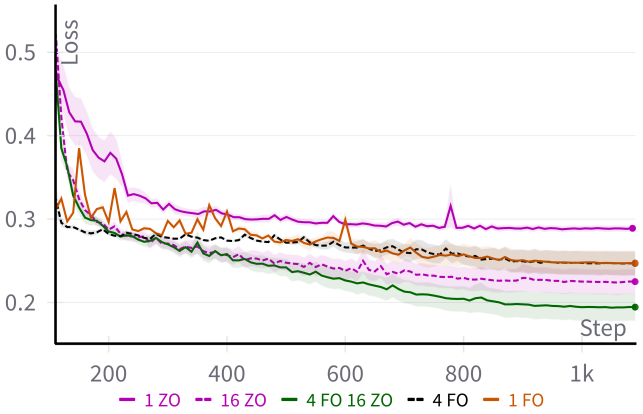

Figure 4: Validation loss comparison between the hybrid and mono-type estimator population for transformer model on the synthetic Brackets dataset.

ased forward-only estimator recently proposed by (Baydin et al. 2022). The results clearly demonstrate an accuracy-versus-steps advantage for a higher number of random vectors and for unbiased zeroth-order estimators compared to biased ones. Since the computational overhead of unbiasing estimators is relatively low (Chen 2020), we will use unbiased zeroth-order estimators in the subsequent experiments. A detailed explanation and experimental evaluation of the accuracy-efficiency trade-offs associated with the number of random vectors will be provided in the Appendix.

For the next set of experiments, we evaluate different *mono-type populations* of ZO and FO optimizers and compare their performance with that of a hybrid population. In Figure 3, we fine-tune the ResNet-18 (He et al. 2015) model, pre-trained with ImageNet-1K (Russakovsky et al. 2015), on CIFAR-10 (Krizhevsky 2012) using populations of 1 ZO, 1 FO, 5 ZO, and a hybrid system consisting of 1 FO and 5 ZO nodes. To demonstrate the scalability of HDO in both convex and non-convex scenarios, we consider two settings: for the convex case, we assess the performance of a Logistic model on the MNIST dataset with various mono-type populations and a hybrid configuration of 24 FO and 256 ZO (Figure 2). For the non-convex case, we use a Transformer model (Vaswani et al. 2023) on "Brackets" dataset (Ebrahimi, Gelda, and Zhang 2020) (see the Appendix for more details). The populations that we study here are 1 ZO, 1 FO, 4 FO, 16 ZO, and a hybrid combination of 4 FO and 16 ZO (Figure 4).

The results presented in Figure 2 (for the convex case), and Figures 3 & 4 (for the non-convex case) confirm the intuition, as well as our analysis; first-order nodes always outperform the same or lower number of zeroth-order ones, as shown in all the figures. However, zeroth-order nodes can in fact outperform first-order ones if their number is larger. We can conclude that 1) a larger population of ZO nodes can outperform smaller populations of FO or ZO nodes; and that 2) this larger uniform population is itself outperformed by a hybrid population.

Our experiments demonstrate the desired speedup and scalability; populations with a large number of ZO nodes have faster convergence. Interestingly, aligned with the Theorem 1 that the speedup caused by the ZO nodes will appear after sufficiently large T, in Figure 2, we observe that the (24-FO) group has a lower validation loss initially but the (256-ZO) and (24-FO, 256-ZO) groups outperform the former group after step 200. A similar phenomenon can also be observed in Figure 4. These results validate the theoretical finding, showing that with a sufficient number of steps, hybrid populations achieve faster convergence compared to *homogeneous populations* of first-order optimizers.

Discussion, Limitations, and Future Work

We provided a first analysis of the convergence of decentralized gradient-based methods in a population mixing first- and zeroth-order gradient estimators for both convex and non-convex objectives. Our results show that even biased or noisy zeroth-order information can enhance convergence when integrated into a protocol.

The experimental results validate our analysis and premise, demonstrating that first- and zeroth-order estimators can be effectively hybridized in a decentralized population. This is promising for environments with heterogeneous computational power, allowing agents to leverage local data even without gradient extraction capabilities.

A practical embodiment could be a decentralized learning system where computationally powerful agents perform backpropagation as first-order agents, while computationally limited nodes estimate gradients via forward passes over local data, sharing this information during pairwise interactions.

We focused on *decentralized optimization*, where nodes interact in randomly chosen pairs. However, our analysis can extend to more general interaction graph topologies, where convergence depends on the eigenvalue gap. Another extension we plan to explore includes additional gradient estimators and large-scale practical deployments to validate our approach. Our experimental simulations confirm the feasibility of our method, achieving their intended goal.

Acknowledgements

This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No 805223 ScaleML). The authors would like to acknowledge Eugenia Iofinova for useful discussions during the inception of this project.

References

Agarwal, A.; Dekel, O.; and Xiao, L. 2010. Optimal Algorithms for Online Convex Optimization with Multi-Point Bandit Feedback. In *Colt*, 28–40. Citeseer.

Alistarh, D.; Hoefler, T.; Johansson, M.; Konstantinov, N.; Khirirat, S.; and Renggli, C. 2018. The convergence of sparsified gradient methods. In *NIPS*, 5977–5987.

Angluin, D.; Aspnes, J.; Diamadi, Z.; Fischer, M. J.; and Peralta, R. 2006. Computation in networks of passively mobile finite-state sensors. *Distributed computing*, 18(4): 235–253.

Assran, M.; Loizou, N.; Ballas, N.; and Rabbat, M. 2018. Stochastic gradient push for distributed deep learning. *arXiv preprint arXiv:1811.10792*.

Balasubramanian, K.; and Ghadimi, S. 2022. Zeroth-Order Nonconvex Stochastic Optimization: Handling Constraints, High Dimensionality, and Saddle Points. *Found. Comput. Math.*, 22(1): 35–76.

Baydin, A. G.; Pearlmutter, B. A.; Syme, D.; Wood, F.; and Torr, P. 2022. Gradients without Backpropagation.

Bottou, L. 2010. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*, 177–186. Springer.

Chen, G. 2020. Unbiased Gradient Simulation for Zeroth-Order Optimization. In *2020 Winter Simulation Conference (WSC)*, 2947–2959.

Deng, L. 2012. The mnist database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine*, 29(6): 141–142.

Ebrahimi, J.; Gelda, D.; and Zhang, W. 2020. How Can Self-Attention Networks Recognize Dyck-n Languages? arXiv:2010.04303.

Flaxman, A. D.; Kalai, A. T.; and McMahan, H. B. 2004. Online convex optimization in the bandit setting: gradient descent without a gradient. *arXiv preprint cs/0408007*.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2015. Deep Residual Learning for Image Recognition. arXiv:1512.03385.

Iutzeler, F.; Bianchi, P.; Ciblat, P.; and Hachem, W. 2013. Asynchronous distributed optimization using a randomized alternating direction method of multipliers. In *52nd IEEE conference on decision and control*, 3671–3676. IEEE.

Johansson, B.; Rabi, M.; and Johansson, M. 2009. A randomized incremental subgradient method for distributed optimization in networked systems. *SIAM Journal on Optimization*, 20(3): 1157–1170.

Jongeneel, W.; Yue, M.-C.; and Kuhn, D. 2021. Small errors in random zeroth-order optimization are imaginary.

Kempe, D.; Dobra, A.; and Gehrke, J. 2003. Gossip-based computation of aggregate information. In *44th Annual IEEE Symposium on Foundations of Computer Science, 2003. Proceedings.*, 482–491. IEEE.

Koloskova, A.; Lin, T.; Stich, S. U.; and Jaggi, M. 2020a. Decentralized Deep Learning with Arbitrary Communication Compression. In *International Conference on Learning Representations*.

Koloskova, A.; Loizou, N.; Boreiri, S.; Jaggi, M.; and Stich, S. U. 2020b. A Unified Theory of Decentralized SGD with Changing Topology and Local Updates. In *ICML*, 5381–5393.

Krizhevsky, A. 2012. Learning Multiple Layers of Features from Tiny Images. *University of Toronto*.

Lecun, Y.; Bottou, L.; Bengio, Y.; and Haffner, P. 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11): 2278–2324.

Lian, X.; Zhang, C.; Zhang, H.; Hsieh, C.-J.; Zhang, W.; and Liu, J. 2017a. Can Decentralized Algorithms Outperform Centralized Algorithms? A Case Study for Decentralized Parallel Stochastic Gradient Descent. *arXiv preprint arXiv:1705.09056*.

Lian, X.; Zhang, W.; Zhang, C.; and Liu, J. 2017b. Asynchronous decentralized parallel stochastic gradient descent. *arXiv preprint arXiv:1710.06952*.

Mhanna, E.; and Assaad, M. 2023. Single Point-Based Distributed Zeroth-Order Optimization with a Non-Convex Stochastic Objective Function. In Krause, A.; Brunskill, E.; Cho, K.; Engelhardt, B.; Sabato, S.; and Scarlett, J., eds., *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, 24701–24719. PMLR.

Nadiradze, G.; Sabour, A.; Davies, P.; Li, S.; and Alistarh, D. 2021. Asynchronous decentralized SGD with quantized and local updates. *Advances in Neural Information Processing Systems*, 34.

Nedic, A.; and Ozdaglar, A. 2009. Distributed subgradient methods for multi-agent optimization. *IEEE Transactions on Automatic Control*, 54(1): 48.

Nesterov, Y.; and Spokoiny, V. 2017. Random gradient-free minimization of convex functions. *Foundations of Computational Mathematics*, 17(2): 527–566.

Russakovsky, O.; Deng, J.; Su, H.; Krause, J.; Satheesh, S.; Ma, S.; Huang, Z.; Karpathy, A.; Khosla, A.; Bernstein, M.; Berg, A. C.; and Fei-Fei, L. 2015. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3): 211–252.

Sahu, A. K.; and Kar, S. 2020. Decentralized zeroth-order constrained stochastic optimization algorithms: Frank–Wolfe and variants with applications to black-box adversarial attacks. *Proceedings of the IEEE*, 108(11): 1890–1905.

Shamir, O. 2017. An optimal algorithm for bandit and zero-order convex optimization with two-point feedback. *The Journal of Machine Learning Research*, 18(1): 1703–1713.

Shamir, O.; and Srebro, N. 2014. Distributed stochastic optimization and learning. In *2014 52nd Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, 850–857. IEEE.

Tang, H.; Zhang, C.; Gan, S.; Zhang, T.; and Liu, J. 2018. Decentralization meets quantization. *CoRR, abs/1803.06443*.

Tsitsiklis, J. N. 1984. Problems in decentralized decision making and computation. Technical report, Massachusetts Inst of Tech Cambridge Lab for Information and Decision Systems.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, L.; and Polosukhin, I. 2023. Attention Is All You Need. arXiv:1706.03762.

Wang, J.; and Joshi, G. 2021. Cooperative SGD: A Unified Framework for the Design and Analysis of Local-Update SGD Algorithms. *Journal of Machine Learning Research*, 22(213): 1–50.

Wei, E.; and Ozdaglar, A. 2012. Distributed alternating direction method of multipliers. In *2012 IEEE 51st IEEE Conference on Decision and Control (CDC)*, 5445–5450. IEEE.

Xiao, L.; and Boyd, S. 2004. Fast linear iterations for distributed averaging. *Systems & Control Letters*, 53(1): 65–78.

Yuan, D.; Wang, L.; Proutiere, A.; and Shi, G. 2024. Distributed zeroth-order optimization: Convergence rates that match centralized counterpart. *Automatica*, 159: 111328.