Do We Need Asynchronous SGD? On the Near-Optimality of Synchronous Methods

Grigory Begunov
AXXX, Moscow, Russia
MSU, Moscow, Russia

Alexander Tyurin
AXXX, Moscow, Russia
Applied AI Institute, Moscow, Russia

Abstract

Modern distributed optimization methods mostly rely on traditional *synchronous* approaches, despite substantial recent progress in *asynchronous* optimization. We revisit Synchronous SGD and its robust variant, called m-Synchronous SGD, which can be interpreted as a method with a backup-worker strategy, and *theoretically* show that they are nearly optimal in many heterogeneous computation and communication scenarios, which is somewhat unexpected. We analyze the synchronous methods under random times and adversarial partial participation of workers, and prove that their time complexities are optimal in many practical regimes, up to logarithmic factors. While synchronous methods are not universal solutions and there exist tasks where asynchronous methods may be necessary, we show theoretically that they are sufficient for many modern heterogeneous computation and communication scenarios.

1 Introduction

We consider the stochastic optimization problem of minimizing a function:

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

where $f : \mathbb{R}^d \to \mathbb{R}$ is L–smooth function (Assumption 1.1) and bounded (Assumption 1.2). There are n workers who can only calculate stochastic gradients $\nabla f(x; \xi)$ at any point $x \in \mathbb{R}^d$ with $\xi \sim \mathcal{D}$, where $\mathcal{D}$ is some distribution, such that $\nabla f(x; \xi)$ is an unbiased estimate of $\nabla f(x)$ and has σ^2–bounded variance (Assumption 1.3). Our goal in the nonconvex setting is to find a (potentially random) vector $\bar{x} \in \mathbb{R}^d$ such that $\mathbb{E}[\|\nabla f(\bar{x})\|^2] \leq \varepsilon$, i.e., an ε–stationary point. This is a standard optimization problem in machine learning, federated learning, training deep neural networks, and large language models (Konečný et al., 2016; McMahan et al., 2016).

We focus on the setting in which multiple workers (e.g., GPUs, servers) work together to find an ε–stationary point, and we consider realistic scenarios where the computation times of stochastic gradients are not fixed and may be different for each worker, random due to natural computation fluctuations and spontaneous outages, and potentially varying over time. In Section B, we extend the results to scenarios where communication also takes time. In this work, we assume vector aggregation times and system-dependent delays are negligible compared with the computation and communication times of stochastic gradients. We consider the standard assumptions:

Assumption 1.1. Function f is L–smooth: $\|\nabla f(x) - \nabla f(y)\| \leq L \|x - y\|$ for all $x, y \in \mathbb{R}^d$.

Assumption 1.2. There exist $f^* \in \mathbb{R}$ such that $f(x) \geq f^*$ for all $x \in \mathbb{R}^d$. We define $\Delta := f(x^0) - f^*$, where x^0 is the starting point of optimization methods.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

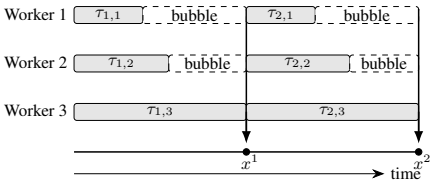


Figure 1: Wall-clock time visualization of Synchronous SGD (Algorithm 1 or 3).

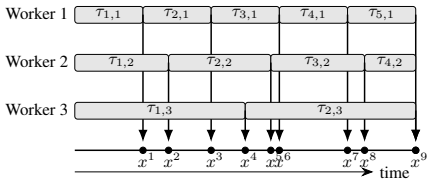


Figure 2: Wall-clock time visualization of Asynchronous SGD (Algorithm 2).

Surprisingly, this paper demonstrates that "bubbles" are not a fundamental obstacle, and that Synchronous SGD and m-Synchronous SGD can remain nearly optimal despite their presence.

Assumption 1.3. The stochastic gradients $\nabla f(x;\xi)$ are unbiased and have σ^2–bounded variance: $\mathbb{E}_\xi\left[\nabla f(x;\xi)\right] = \nabla f(x)$ and $\mathbb{E}_\xi[\|\nabla f(x;\xi) - \nabla f(x)\|^2] \leq \sigma^2$ for all $x \in \mathbb{R}^d$, where $\sigma > 0$ is some constant.

Known methods in asynchronous optimization. The most naive and simplest method in this setting is Synchronous SGD (Algorithm 1) where all workers calculate *one* stochastic gradient each, aggregate them, and perform stochastic gradient descent. The method has a clear disadvantage: in every iteration, it waits for the slowest worker, leading to "bubbles" in computation and idle workers (see Figure 1). The same problem applies to the synchronous ADAM and Muon methods[1] (Kingma and Ba, 2014; Jordan et al., 2024).

In order to improve computational efficiency and avoid "bubbles," several asynchronous methods have been proposed. One of the most celebrated is Asynchronous SGD (Tsitsiklis et al., 1986; Recht et al., 2011) (Algorithm 2). The main idea is that, instead of waiting for all workers, Algorithm 2 immediately updates the iterate once a stochastic gradient is received from any worker (see Figure 2). The theoretical properties of it have been analyzed in (Lian et al., 2015; Feyzmahdavian et al., 2016; Stich and Karimireddy, 2020; Sra et al., 2016; Cohen et al., 2021; Koloskova et al., 2022; Mishchenko et al., 2022; Maranjyan et al., 2025b).

Another method designed for asynchronous settings is Rennala SGD (Tyurin and Richtárik, 2023). The main idea is slightly different compared to Asynchronous SGD: instead of immediately updating the current iterate, Rennala SGD employs an asynchronous batch aggregation strategy to reduce "bubbles" and idle time. Additionally, there are many other asynchronous methods with state-of-the-art theoretical properties (Tyurin and Sivtsov, 2026).

Time complexities. Many synchronous and asynchronous methods have been developed. *How can we theoretically compare them?* Mishchenko et al. (2022) proposed assuming that, for all $i \in [n]$, the i^{th} worker requires τ_i seconds to calculate a stochastic gradient. Without loss of generality (w.l.o.g.), we assume that $\tau_1 \leq \cdots \leq \tau_n$. Using this assumption, we can compare methods. One can prove that the *time complexity* of Synchronous SGD (Algorithm 1) is

$$\mathcal{O}\left(\tau_n \max\left\{\frac{L\Delta}{\varepsilon}, \frac{\sigma^2 L\Delta}{n\varepsilon^2}\right\}\right) \tag{1}$$

seconds, where the result follows from the fact that Synchronous SGD converges after $\max\left\{L\Delta/\varepsilon, \sigma^2 L\Delta/n\varepsilon^2\right\}$ iterations (Lan, 2020), and each iteration requires τ_n seconds, since the method waits for the slowest worker. At the same time, it was shown (Tyurin and Richtárik, 2023; Maranjyan et al., 2025b; Tyurin and Sivtsov, 2026) that the time complexity of Rennala SGD, Asynchronous SGD (Ringmaster ASGD), and other asynchronous methods is

$$T_{\text{optimal}} := \Theta\left(\min_{m\in[n]}\left[\left(\frac{1}{m}\sum_{i=1}^{m}\frac{1}{\tau_i}\right)^{-1}\left(\max\left\{\frac{L\Delta}{\varepsilon}, \frac{\sigma^2 L\Delta}{m\varepsilon^2}\right\}\right)\right]\right), \tag{2}$$

which is also *optimal*. The time complexities (1) and (2) formalize the intuition that Synchronous SGD is suboptimal and does not fully utilize the workers' time. In particular, (1) is never better than (2), but can be arbitrarily worse. For instance, take $\tau_n \to \infty$, i.e., the last worker is extremely slow. In this case, (1) $\to \infty$, while (2) remains stable due to its harmonic dependence on $\{\tau_i\}$.

[1] In this paper, we focus on the SGD update since it is theoretically optimal (Arjevani et al., 2022; Tyurin and Richtárik, 2023); however, all our conclusions potentially apply to other updates.

Algorithm 1 Synchronous SGD

Input: point $x^0 \in \mathbb{R}^d$, stepsize $\gamma > 0$
for $k = 0, \ldots, K-1$ **do**
  Workers calculate stoch. gradients at x^k
  $g^k = \frac{1}{n} \sum_{i=1}^n \nabla f(x^k; \xi_i)$
  Update $x^{k+1} = x^k - \gamma g^k$
end for

Algorithm 2 Asynchronous SGD

Input: point $x^0 \in \mathbb{R}^d$, stepsizes $\gamma_k \geq 0$
Workers start computing stoch. gradients at x^0
for $k = 0, \ldots, K-1$ **do**
  $\nabla f(x^{k-\delta^k}; \xi_i^{k-\delta^k})$ arrives from worker i
  Update $x^{k+1} = x^k - \gamma_k \nabla f(x^{k-\delta^k}; \xi_i^{k-\delta^k})$
  Worker i begins calculating $\nabla f(x^{k+1}; \xi_i^{k+1})$
end for

Is Synchronous SGD fundamentally limited? Indeed, Synchronous SGD (Algorithm 1) is not robust to large computation times of slow workers. This leads to our first research question: instead of designing and using asynchronous methods, would it be better to simply ignore slow workers and use Synchronous SGD with the fastest ones during optimization? To this end, consider m-Synchronous SGD (Algorithm 3) with the only difference being that, at each iteration, it aggregates *one stochastic gradient* from each of the first m workers to finish, rather than from all n workers:

Algorithm 3 m-Synchronous SGD

Input: point $x^0 \in \mathbb{R}^d$, stepsize γ, # of workers $m \in [n]$
for $k = 0, 1, \ldots, K-1$ **do**
  Each idle worker starts computing a stochastic gradient at x^k
  Aggregate[2] the first m workers at x^k: $g^k = \frac{1}{m} \sum_{i \in S_k} \nabla f(x^k; \xi_i^k)$, $S_k \subseteq [n]$, $|S_k| = m$
  Update $x^{k+1} = x^k - \gamma g^k$
  Discard any gradients for iteration k that arrive after the update
end for

Notice that m-Synchronous SGD is still synchronous and asks for only one stochastic gradient from each of the m participating workers; it reduces to Synchronous SGD when $m = n$ and can be viewed as a robust version of it that ignores $n - m$ stragglers. Another important interpretation of this method is as a backup-worker strategy, which is widely used in practice instead of Asynchronous SGD (Chen et al., 2016).

A related question is the following: while (1) $\to \infty$ as $\tau_n \to \infty$, in practice $\tau_n < \infty$, and it may be the case that the resulting gap is not too large. Even if $\tau_n \gg \tau_1$, it might be possible that (1) is optimal or nearly optimal[3]. Moreover, even if the computation times are random, their variance may not be large enough to make Synchronous SGD (Algorithm 1) asymptotically suboptimal.

Synchronous methods are still the de facto and most widely used, there is empirical evidence that it is actually fast and yields better accuracy (Chen et al., 2016; Shoeybi et al., 2019; Rajbhandari et al., 2020; Grattafiori et al., 2024). Can we provide a *theoretical justification* for their time performance?

1.1 Contributions

We provide theoretical evidence that synchronous methods are nearly as fast as modern asynchronous methods in many heterogeneous computational scenarios.

♠ We first show that, under Assumption 2.2, m-Synchronous SGD (Algorithm 3) is nearly optimal up to a $\log(n+1)$ factor for arbitrary heterogeneous worker computation times, a somewhat unexpected result (see Section 2). This also holds for m-Synchronous SGD with minibatching in the presence of communication times (see Section B). In particular, this is the first theoretical evidence that a synchronous method with a backup-worker strategy can be nearly as fast as asynchronous methods, a fact that was previously known only empirically (Chen et al., 2016).

♣ We then extend this result to Assumption 3.1, where worker computation times are random, and show theoretically that m-Synchronous SGD (Algorithm 3) remains nearly optimal in many practical settings (see also Section B for the case with communication times). To establish optimality, we derive a new lower bound (Theorem 3.3), which we believe is of independent interest (see Section 3).

[2] Not necessarily from worker 1 to m. It can be any of the first m workers from the set $[n]$ who finish computing at x^k.
[3] We say that a method is nearly optimal if it is optimal up to a constant or logarithmic factor.

♦ Even the standard Synchronous SGD method *with all workers participating* is nearly optimal in some important regimes: (i) when computation times are random with equal means, for example, when they follow the exponential distribution (see Section 3 and Corollary 3.4); and (ii) when the means are not equal but follow a power law $\tau_m = \tau_1 m^\alpha + \delta_m$, with $0 \leq \alpha \leq 1$ and bounded δ_m. In this case, Synchronous SGD is still nearly optimal, for instance, for the truncated normal distribution, even though $\tau_n \gg \tau_1$ (see Section 4 and Proposition 4.2).

♥ We also consider scenarios in which computation trends are non-stationary and vary over time. One important result is that m-Synchronous SGD (Algorithm 3) is optimal under arbitrary, and even adversarial, worker participation (Assumption 3.1) (see Section 5).

We acknowledge in Section 6 that synchronous methods are not a silver bullet, and that there are several scenarios where asynchronous methods are still needed. Nevertheless, somewhat counterintuitively, we prove that Synchronous SGD and m-Synchronous SGD are nearly optimal in a wide range of heterogeneous computation settings, which may be sufficient in practice.

2 Synchronous Methods Are Nearly Optimal under Fixed Computation

In order to present our main result, we first recall the classical SGD result for the required number of iterations to converge:

Theorem 2.1 (Iteration Complexity; (Lan, 2020)). *Under Assumptions 1.1, 1.2, and 1.3, let* $\gamma = \min\left\{1/2L, \varepsilon m/4L\sigma^2\right\}$ *in Algorithm 3. Then* $\frac{1}{K}\sum_{k=0}^{K-1}\mathbb{E}[\|\nabla f(x^k)\|^2] \leq \varepsilon$ *after* $K := \left\lceil 16\max\left\{L\Delta/\varepsilon, \sigma^2 L\Delta/m\varepsilon^2\right\}\right\rceil$ *iterations, where* $m \in [n]$ *is an arbitrary number.*

Theorem 2.1 holds since Algorithm 3 is SGD with batch size m. At the beginning, we consider the fixed computation model, discussed in Section 1:

Assumption 2.2. For all $i \in [n]$, the i^{th} worker requires τ_i seconds to calculate a stochastic gradient. W.l.o.g., we assume that $\tau_1 \leq \cdots \leq \tau_n$.

Recall the optimal time complexity T_{optimal} from (2). In the following theorem, we prove our first result: m-Synchronous SGD, with a proper choice of m, achieves the optimal time complexity up to a logarithmic factor.

Theorem 2.3 (Time Complexity under Fixed Computation Model). *Under the setting of Theorem 2.1, additionally assume that Assumption 2.2 holds. Then, the time complexity of Algorithm 3 to find an* ε*–stationary point is*

$$T_{\text{sync}} := \frac{16L\Delta}{\varepsilon}\min_{m\in[n]}\left[\tau_m \max\left\{1, \frac{\sigma^2}{m\varepsilon}\right\}\right] \tag{3}$$

if m *is a minimizer of* $\min_{m\in[n]}\left[\tau_m \max\left\{1, \frac{\sigma^2}{m\varepsilon}\right\}\right]$. *More importantly, the result is nearly optimal:*

$$T_{\text{sync}} = \mathcal{O}\left(T_{\text{optimal}} \times \log(n+1)\right). \tag{4}$$

To the best of our knowledge, this is the first result showing that a synchronous method is nearly as fast as the optimal asynchronous algorithms for stochastic optimization in the presence of heterogeneous computation times (Assumption 2.2). This result appears to have been overlooked in previous works (Cohen et al., 2021; Koloskova et al., 2022; Mishchenko et al., 2022; Tyurin and Richtárik, 2023; Maranjyan et al., 2025b). It turns out that it is possible to design a nearly optimal *synchronous* method under Assumption 2.2. Theorem B.4 shows a similar result with communication times.

Proof sketch. The first term in (3) is straightforward and follows from Theorem 2.1 and the fact that the method waits for the fastest m workers. Then, we choose $m \in [n]$ that minimizes $\tau_m \max\left\{1, \sigma^2/m\varepsilon\right\}$. Under Assumption 2.2, the fastest m workers are those with indices $\{1, \ldots, m\}$.

It is left to prove (4). For simplicity, assume that $\sigma^2/\varepsilon \geq n$, then $m = \arg\min_{m\in[n]} \tau_m \max\left\{1, \sigma^2/m\varepsilon\right\} = \arg\min_{m\in[n]} \tau_m/m$. Thus, $\tau_m/m \leq \tau_i/i$ and $\tau_i \geq \tau_m i/m$ for all $i \in [n]$. Substituting the last inequality to (2), if $\sigma^2/\varepsilon \geq n$, then

$$T_{\text{optimal}} = \Theta\left(\left(\frac{1}{n}\sum_{i=1}^{n}\frac{1}{\tau_i}\right)^{-1}\frac{\sigma^2 L\Delta}{n\varepsilon^2}\right) \geq \Omega\left(\frac{L\Delta}{\varepsilon}\left(\sum_{i=1}^{n}\frac{1}{i}\right)^{-1}\frac{\tau_m\sigma^2}{m\varepsilon}\right) = \Omega\left(\frac{1}{\log(n+1)}T_{\text{sync}}\right)$$

since $\sum_{i=1}^{n} \frac{1}{i} = \Theta(\log(n+1))$, and we get (4). The full proof is provided in Section D. Moreover, the logarithmic term is unavoidable in the case where $\tau_i = i$ for all $i \in [n]$.

Why is this result surprising? Theorem 2.3 shows that we can achieve a nearly optimal rate using standard synchronous SGD with the fastest m workers, *which still compute one stochastic gradient per iteration*. Therefore, the "bubble" problem (Figure 1) is still present, since the fastest worker with computation time τ_1 has to wait for slower workers 2, 3, and so on, up to worker m. Nevertheless, this strategy is nearly optimal. Intuitively, it is always possible to choose m in Algorithm 3 such that the size of the "bubbles" and the idle time are not too large, and it is as fast as more advanced asynchronous methods. It is also interesting to note that even Synchronous SGD (Algorithm 1) can be nearly optimal despite $\tau_n \gg \tau_1$ (see Section 4).

3 Random Computation Model

The obtained result holds not only for the fixed computation model (Assumption 2.2). One way to generalize Assumption 2.2 is to assume that the computation times are random:

Assumption 3.1. For all $i \in [n]$, the i^{th} worker requires $\bar{\tau}_i$ seconds to calculate a stochastic gradient, where $\bar{\tau}_i$ is a (τ_i, R)–sub-exponential random variable, i.e., $\mathbb{E}\left[\exp(|\bar{\tau}_i - \tau_i| / R)\right] \leq 2$, $R > 0$, $\mathbb{E}\left[\bar{\tau}_i\right] = \tau_i$, and $\bar{\tau}_i \geq 0$ a.s.. We assume that the sampled times $\{\bar{\tau}_i\}$ and the stochastic random variables $\{\xi_i\}$ are statistically independent. Moreover, $\tau_1 \leq \cdots \leq \tau_n$ (w.l.o.g.).

This assumption includes a wide range of distributions, including truncated normal, any bounded distributions, all sub-Gaussian distributions, as well as Gamma and exponential distributions (Vershynin, 2018). The latter distribution is commonly used to model time delays in real computational systems (e.g., (Mitliagkas et al., 2016; Dutta et al., 2018)).

Theorem 3.2 (Time Complexity under Random Computation Model)**.** *Under the setting of Theorem 2.1, additionally assume that Assumption 3.1 holds. Then, the time complexity of Algorithm 3 to find an ε–stationary point is at most*

$$T_{\text{rand}} := \sum_{k=0}^{K-1} \max_{i \in [m]} \bar{\tau}_{k,i} \tag{5}$$

seconds, where $\bar{\tau}_{k,i}$ is a (τ_i, R)–sub-exponential random variable (independent copy). Moreover, we have

$$\mathbb{E}\left[T_{\text{rand}}\right] = \mathcal{O}\left(\frac{L\Delta}{\varepsilon}\left(\tau_m + R\log(n)\right)\max\left\{1, \frac{\sigma^2}{m\varepsilon}\right\}\right) \tag{6}$$

for all $m \in [n]$ in Algorithm 3.

We obtain exactly the same result as in Theorem 2.3, with the only difference being an additional "noise" term $R\log(n)$, and the result now holds on average. After establishing the following lower bound, we show that the R term does not dominate and that the result is nearly optimal with $m = \arg\min_{m \in [n]} \left[\tau_m \max\left\{1, \sigma^2/m\varepsilon\right\}\right]$ in a number of relevant examples, both in theory and in practice.

Remark. In this section, we implicitly assume that the workers are not allowed to stop calculations because modern frameworks do not allow this, and the only option is to "kill" the running computation process on the GPU and start the code again, which would lead to even larger delays. We refer to (Maranjyan et al., 2025a), where the authors consider scenarios in which it is possible.

Lower bound. For Assumption 3.1, to the best of our knowledge, there is no lower bound in the literature. Thus, we prove a new lower bound to understand the optimality gap of (6).

Theorem 3.3 (Lower Bound under Random Computation Model)**.** *For all $\Delta, L, \varepsilon, R, \sigma > 0$, and time delays satisfying Assumption 3.1 such that*[4] $\tau_1 \max\left\{1, \sigma^2/n\varepsilon\right\} \geq \bar{c}R\max\left\{1, \sigma/\sqrt{\varepsilon}\right\}\log(n)$*, where $\bar{c}$ is a universal constant, there exists a smooth function f with L–Lipschitz gradients, and $f(0) - f^* \leq \Delta$, and an oracle that returns unbiased stochastic gradients with σ^2–bounded gradient variance such that the expected time of any zero-respecting (see Definition H.1) algorithm that does not stop computations is $\Omega\left(T_{\text{optimal}}\right)$ seconds (from (2)) to get an ε-stationary point.*

[4]The inequality is satisfied if, for instance, ε is small enough, R is small enough, or σ is large enough.

Corollary 3.4. *Under the setting of Theorem 3.2, m-Synchronous SGD (Algorithm 3) is nearly optimal, on average, at least in the regimes where $\tau_1 \max\left\{1, \sigma^2/n\varepsilon\right\} \geq \Omega(R \max\left\{1, \sigma/\sqrt{\varepsilon}\right\} \log(n))$ and[5] $\tau_1 \geq \tilde{\Omega}(R)$:*

$$\mathbb{E}\left[T_{\text{rand}}\right] = \mathcal{O}\left(T_{\text{optimal}} \times \log(n+1)\right)$$

if m is a minimizer of $\min_{m \in [n]} \left[\tau_m \max\left\{1, \frac{\sigma^2}{m\varepsilon}\right\}\right]$.

Hence, even Synchronous SGD (Algorithm 1) is nearly optimal in the listed regimes if the computation times are random and the means $\{\tau_i\}$ are the same.

Theoretical examples. The standard examples are the truncated normal and exponential distributions. For instance, if $\bar{\tau}_i$ is from the truncated normal distribution, that is, $\bar{\tau}_i \sim \mathcal{N}(\mu_i, \varsigma^2)$ conditional on $\bar{\tau}_i \geq 0$ for any $\mu_i, \varsigma \geq 0$, then it satisfies Assumption 3.1 with $R = \mathcal{O}(\varsigma)$ (Barreto et al., 2025). Moreover, for the truncated normal distribution, $\tau_i = \Omega(\max\{\mu_i, \varsigma\})$. Therefore, $R = \mathcal{O}(\tau_i)$ for all $i \in [n]$, R does not dominate in (6), and the result (6) *is nearly optimal* due to Corollary 3.4 when, for instance, ε is small.

Consider another example where $\bar{\tau}_i$ is from the exponential distribution $\text{Exp}(\lambda)$ for all workers. In this case, $R = \Theta(1/\lambda)$ and $\tau_i = \mathbb{E}\left[\bar{\tau}_i\right] = 1/\lambda$ for all $i \in [n]$ (Vershynin, 2018). Therefore, $(6) = \tilde{\mathcal{O}}\left(\tau_1 L\Delta/\varepsilon \max\left\{1, \sigma^2/n\varepsilon\right\}\right)$ for $m = n$, and even though the computation times are random, *the time complexity of Synchronous SGD (Algorithm 1) is nearly optimal for all $\lambda > 0$, when, for instance, ε is small.* See more examples in Section H.1.

Practical simulations. In practice, with NanoGPT (Karpathy, 2022), we also observe that $R \log n$ is negligibly small compared to the mean. See details in Section N.

4 On the Optimal Selection of Active Workers

In order to obtain nearly optimal time complexities for m-Synchronous SGD (Algorithm 3) in Sections 2 and 3, we have to minimize

$$g(m) := \tau_m \max\left\{1, \frac{\sigma^2}{m\varepsilon}\right\} \tag{7}$$

for all $m \in [n]$, where m is the number of active workers and $\{\tau_m\}$ is a non-decreasing sequence.

Proposition 4.1. *Consider the mapping $g(m)$ from (7). Then, $g(m)$ is minimized with $m \leq \min\left\{\left\lceil \sigma^2/\varepsilon \right\rceil, n\right\}$. If $\sigma^2/\varepsilon \leq 1$, then the optimal $m = 1$. Otherwise, if $\sigma^2/\varepsilon > 1$, then $\frac{\sigma^2 h(m)}{\varepsilon} \leq g(m) \leq \frac{2\sigma^2 h(m)}{\varepsilon}$ for all $m \leq \min\left\{\left\lceil \sigma^2/\varepsilon \right\rceil, n\right\}$, where*

$$h(m) := \frac{\tau_m}{m}. \tag{8}$$

Proposition 4.1 says that it is necessary and sufficient to find a minimizer of $h(m)$ for all $m \leq \min\left\{\left\lceil \sigma^2/\varepsilon \right\rceil, n\right\}$ to obtain the minimal value of $g(m)$ for all $m \in [n]$ (up to a factor of 2). This proposition provides a simple choice of m and identifies regimes where even Synchronous SGD is nearly optimal. For instance, suppose that $\tau_{m+1} \leq \tau_m\left(1 + 1/m\right)$, meaning that τ_m is a non-decreasing sequence that increases by at most a factor of $1 + 1/m$. One can easily show that the sequences $\tau_m = \tau_1 \times m$ (linear increase) and $\tau_m = \tau_1 \times m^\alpha$ (sublinear increase) satisfy this condition for any $\alpha \in [0, 1]$. In this case, since $h(m)$ is non-increasing, we can take $m = \min\left\{\left\lceil \sigma^2/\varepsilon \right\rceil, n\right\}$ as a minimizer of $h(m)$, and use this m in Algorithms 3. We can generalize the previous result:

Proposition 4.2. *Assume that*

$$\tau_m = \tau_1 m^\alpha + \delta_m \tag{9}$$

for any $\alpha \in [0, 1]$ and δ_m such that $0 \leq \delta_m \leq \delta$. Then, the choice $m = \min\left\{\left\lceil \sigma^2/\varepsilon \right\rceil, n\right\}$ is optimal in (7) when $\min\left\{\left\lceil \sigma^2/\varepsilon \right\rceil, n\right\} \geq \left(\delta/\tau_1\right)^{1/\alpha}$. Thus, for τ_m satisfying (9) and large enough σ^2/ε, even Synchronous SGD (Algorithm 1) is nearly optimal in Corollary 3.4 since $m = n$.

[5] the inequalities hold, for example, when ε is small and R is small compared to τ_1, e.g., when $\bar{\tau}_i \sim \mathcal{N}(\mu_i, \varsigma^2)$ truncated to $\bar{\tau}_i \geq 0$, or when $\bar{\tau}_i$ follow $\text{Exp}(\lambda)$. See also Section H.1.

Thus, if $\{\tau_m\}$ follows the power law (9) with $0 \leq \alpha \leq 1$ and bounded $\{\delta_m\}$, then it is sufficient to take $m = \min\left\{\left\lceil \sigma^2/\varepsilon \right\rceil, n\right\}$ when either n or σ^2/ε is large, *meaning that even Synchronous SGD (Algorithm 1) is nearly optimal in this case when $\sigma^2/\varepsilon \geq n$, despite the fact that $\tau_n \gg \tau_1$.*

Notice that the regime when $\tau_m \approx \tau_1 m^\alpha$ and $\alpha > 1$ is not very interesting since (2) reduces to $\Theta\left(\tau_1 \max\left\{L\Delta/\varepsilon, \sigma^2 L\Delta/m\varepsilon^2\right\}\right)$, which is the time complexity of the fastest worker, and multiple workers are therefore not needed since $\{\tau_i\}$ grows too fast.

5 Universal Computation Model

Here, we consider another way to generalize Assumption 2.2:

Assumption 5.1 (Tyurin (2025))**.** For each worker $i \in [n]$, we relate a non-negative Riemann integrable (continuous almost everywhere) *computation power* $v_i : \mathbb{R}_+ \to \mathbb{R}_+$. The number of stochastic gradients that worker i can compute between times t_0 and t_1 is

$$N_i(t_0, t_1) := \left\lfloor \int_{t_0}^{t_1} v_i(\tau) d\tau \right\rfloor. \tag{10}$$

For instance, assume that the computation times of workers are fixed: $v_i(\tau) = v_i \in \mathbb{R}_+$. In this case, $N_i(t_0, t_1) = \lfloor v_i \times (t_1 - t_0) \rfloor$, and if worker i starts calculating at time t_0, then it will compute one gradient after $t_0 + 1/v_i$ seconds because $N_i(t_0, t_0 + 1/v_i) = 1$, two gradients after $t_0 + 2/v_i$ seconds, and so forth. In this example, Assumption 5.1 reduces to Assumption 2.2 with $\tau_i \equiv 1/v_i$.

However, Assumption 5.1 allows us to use virtually any computation power, capturing virtually all possible computational scenarios. For instance, it is possible that $v_i(t) > 0$ for all $t \leq \bar{t}$ and $v_i(t) = 0$ for all $t > \bar{t}$, meaning that worker i is not available after time $\bar{t}$. Moreover, v_i can have arbitrary fluctuations (see Figures 4 and 5), and, unlike Assumption 3.1, it supports non-stationary trends. See more examples in Sections C.1, 5.1, and C.2. Tyurin (2025) proved the following lower bound.

Theorem 5.2 (Informal Lower Bound under Universal Computation Model; Tyurin (2025))**.** *Let Assumptions 1.1, 1.2, 1.3, and 5.1 hold. With high probability, it is impossible to find an ε–stationary point faster than $1/2 \times \underline{t}_{\underline{K}}$ seconds, where the sequence $\{\underline{t}_k\}$ is defined as*

$$\underline{t}_k = \min\left\{t : \sum_{i=1}^{n} N_i(\underline{t}_{k-1}, t) \geq c_2 \left\lceil \frac{\sigma^2}{\varepsilon} \right\rceil\right\}, \tag{11}$$

where $\underline{t}_0 = 0$, $\underline{K} := \left\lceil c_1 \times \frac{L\Delta}{\varepsilon} \right\rceil$, and c_1 and c_2 are universal constants.

This lower bound $\Omega(\underline{t}_{\underline{K}})$ is achieved by the Rennala SGD and Ringmaster ASGD asynchronous methods (up to constant factors) (Tyurin and Richtárik, 2023; Maranjyan et al., 2025b). The result is non-explicit, and, in general, to find $\underline{t}_{\underline{K}}$, one has to first find $\underline{t}_1$, $\underline{t}_2$, and so forth, either theoretically or numerically. Under Assumption 5.1, we can prove the following theorem for m-Synchronous SGD:

Theorem 5.3 (Time Complexity under Universal Computation Model)**.** *Under the setting of Theorem 2.1, additionally assume that Assumption 5.1 holds. Then, for all $m \in [n]$, the time complexity of m-Synchronous SGD (Algorithm 3) is $\bar{t}_{\bar{K}}$ seconds, where the sequence $\{\bar{t}_k\}$ is defined as*

$$\bar{t}_{k+1} = \min\left\{t \geq 0 : \max_{S \subseteq [n], |S| = m} \min_{i \in S} N_i(\bar{t}_k, t) = 2\right\}, \tag{12}$$

$\bar{t}_0 = 0$, *and* $\bar{K} := \left\lceil 16 \max\left\{L\Delta/\varepsilon, \sigma^2 L\Delta/m\varepsilon^2\right\}\right\rceil$.

Theorem 5.3 is also non-explicit. In general, Theorem 5.3 does not match the lower bound in Theorem 5.2 (see Section M); however, there are several theoretical and practical examples where it does. We consider examples with constant powers and numerically generated powers in Section C.

5.1 Example: Optimization with stragglers

One important example is when all workers have the same computational power, but at most pn of them can be idle at any moment (e.g., $p = 0.1$ and at most 10% are stragglers; see possible visualization in Figure 4).

Assumption 5.4. Under Assumption 5.1, for some fixed $p \in [0, 1)$, assume that $\forall t \geq 0$, there exists a subset S_t of at least size $(1-p)n$ such that $v_i(t) = v \in \mathbb{R}_+ \; \forall i \in S_t$, and $v_i(t) \leq v \; \forall i \notin S_t$.

Notice that for all $t \geq 0$, we do not make any assumptions about the subset S_t; *the subsets can be fixed, random, or even adversarial to the algorithm.* Clearly, in this case, a lower bound is

$$\Omega\left(\frac{1}{v}\max\left\{\frac{L\Delta}{\varepsilon}, \frac{\sigma^2 L\Delta}{n\varepsilon^2}\right\}\right), \tag{13}$$

in the best possible case when all workers participate. We now show that the time complexity of m-Synchronous SGD (Algorithm 3) is optimal and matches (13) when $p < 0.4$:

Theorem 5.5 (Time Complexity under Partial Participation)**.** *Under the setting of Theorem 2.1, additionally assume that Assumption 5.4 holds. Then, the time complexity of Algorithm 3 is optimal for all $p < 0.4$. For all $p < 0.4$ and any $m \in \{\lfloor n/5 \rfloor, \ldots, \lfloor (1-2p)n \rfloor\}$, the time complexity is*

$$\mathcal{O}\left(\frac{1}{v}\max\left\{\frac{L\Delta}{\varepsilon}, \frac{\sigma^2 L\Delta}{n\varepsilon^2}\right\}\right) \tag{14}$$

Comparing (13) and (14), one can see that the complexity of m-Synchronous SGD is tight up to a constant factor when $p < 0.4$, which is a practical regime in the training of machine learning models, where it is very unlikely that more than 40% of workers (GPUs) fail at the same moment in time. Algorithm 3 does not need the sets of participating workers; it is sufficient to fix any $m \in \{\lfloor n/5 \rfloor, \ldots, \lfloor (1-2p)n \rfloor\}$. In practice, of course, one should take $m = \lfloor (1-2p)n \rfloor$ to maximize the number of active workers. Notice that it is not necessarily workers 1 through m that participate; the set of active workers may vary. Thus, even under the general partial participation assumption, it is sufficient to use m-Synchronous SGD to obtain the optimal theoretical time complexity.

6 When Asynchronicity Might Be Needed

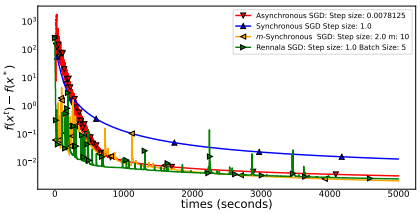
In the previous sections, we explained and proved that m-Synchronous SGD and Synchronous SGD are almost as fast as asynchronous methods in many scenarios. At the same time, we should acknowledge that synchronous methods are not a silver bullet and that there are several scenarios and corner cases where asynchronous methods might be needed.

Rapid changes in computation times. In general, it is possible that the upper bound $\bar{t}_{\bar{K}}$ from Theorem 5.3 is asymptotically larger than the lower bound $\underline{t}_{\underline{K}}$ from Theorem 5.2. Indeed, consider an example when the computation times of workers are the same and equal τ at the beginning. In this case, we choose $m = n$ in Algorithm 3. However, suddenly, the first worker becomes infinitely fast after a few iterations: the new computation time is $\tilde{\tau}_1 \approx 0$, while the other computation times remain the same, $\tilde{\tau}_i = \tau$ for all $i > 1$. In this case, the time complexity of m-Synchronous SGD does not change, since it depends only on τ because $m = n$, and the first worker calculates only one gradient, even though it is extremely fast. At the same time, the asynchronous methods will immediately adapt to the sudden speed of the first worker and leverage it. This is an example when the asynchronicity helps. See the formal proof in Section M. However, this scenario is rare and might be arguably impractical.

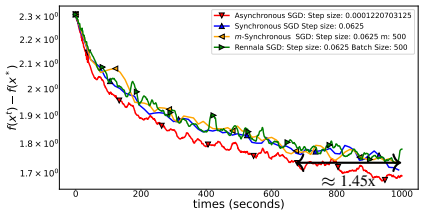
Large variance. Recall Theorem 3.2 from Section 3, where we prove the time complexity (6) for m-Synchronous SGD. One potential limitation of this result is that R can be large. Informally, the R term comes from the fact that it is possible that $\mathbb{E}\left[\max_{i\in[m]} \bar{\tau}_{k,i}\right] \approx R \gg \tau_i$ due to the $\max_{i\in[m]}$ operation; at the same time, the asynchronous Rennala SGD method might be more robust to noise since, unlike m-Synchronous SGD, the method waits for several stochastic gradients, and thus the total waiting time averages the noise, potentially leading to a better asymptotic dependence on R.

Heterogeneous setup. Under Assumptions 2.2 and 5.1, the heterogeneous setup with asynchronous methods is also well-explored (Tyurin and Richtárik, 2023; Tyurin, 2025; Maranjyan and Richtárik, 2025). In particular, consider stochastic optimization problem of minimizing a function: $\min_{x\in\mathbb{R}^d}\left\{f(x) := 1/n \sum_{i=1}^n f_i(x)\right\}$ where $f_i : \mathbb{R}^d \to \mathbb{R}$ and f is a smooth function (Assumption 1.1). Unlike the previous homogeneous setup, here we assume that only worker i can compute stochastic gradients $\nabla f_i(x;\xi)$ at any point $x \in \mathbb{R}^d$ with $\xi \sim \mathcal{D}_i$, where $\mathcal{D}_i$ is some distribution such that $\nabla f_i(x;\xi)$ is an unbiased estimate of $\nabla f_i(x)$ and σ^2-variance bounded. Under

[5] In the proof of the lower bound, c_1 and c_2 are universal but small constants. For a fair comparison of $\bar{t}_{\bar{K}}$ and $\underline{t}_{\underline{K}}$, we take $c_1 = 16$ and $c_2 = 1$, which can be obtained for Rennala SGD.



(a) Quadratic optimization with $n = 10^3$ and $\tau_i = \sqrt{i}$.



(b) CIFAR10 with NN and $\bar{\tau}_i \sim \text{Unif}(0.5, 1.5)$.

Assumption 2.2, it was shown that the optimal time complexity is

$$\Theta\left(\tau_n \frac{L\Delta}{\varepsilon} + \left(\frac{1}{n}\sum_{i=1}^{n} \tau_i\right) \frac{\sigma^2 L\Delta}{n\varepsilon^2}\right), \tag{15}$$

achieved by Malenia SGD (Tyurin and Richtárik, 2023), an async method similar to Rennala SGD.

We want to emphasize that classical Asynchronous SGD (Algorithm 2) cannot converge in general without the participation of all workers, since it is always possible to "put all information about f" on a worker that does not participate (Mishchenko et al., 2022; Tyurin and Richtárik, 2023). For the same reason, we cannot implement Algorithm 3 with $m < n$, since in the heterogeneous setting the participation of all workers is required because, in the worst case, potentially all information about f can be adversarially placed in the workers that are ignored. Thus, it might be the case that the asynchronicity in Malenia SGD is still needed to achieve the optimal time complexity (15) and improve (1).

However, the gap between τ_n and $\frac{1}{n}\sum_{i=1}^{n} \tau_i$ is not as bad as one might think. While it can be arbitrarily large (e.g., $\tau_1 = \cdots = \tau_{n-1} \approx 0$ and $\tau_n \gg 0$), one can show that $\frac{1}{n}\sum_{i=1}^{n} \tau_i \geq \frac{1}{2}\tau_{\lfloor n/2 \rfloor}$, and $\tau_n / \left(\frac{1}{n}\sum_{i=1}^{n} \tau_i\right) = \mathcal{O}(\tau_n / \tau_{\lfloor n/2 \rfloor})$. In many scenarios where the τ_i grow reasonably fast, the gap can be a constant factor. For instance, if $\tau_m = \tau_1 m^\alpha$ for all $m \geq 1$ and some $0 \leq \alpha \leq 4$, then $\tau_n / \left(\frac{1}{n}\sum_{i=1}^{n} \tau_i\right) = \mathcal{O}(1)$, meaning that even if τ_m grows as fast as $\tau_1 m^4$, the gap between Malenia SGD and Synchronous SGD is only a constant factor.

7 Experimental Highlights

In Section O, we conducted numerical experiments on quadratic optimization tasks and small-scale machine learning problems in controlled heterogeneous computation environments, and on NanoGPT training in a real environment. All our experiments concur with our theoretical results, and we observe that either Synchronous SGD or m-Synchronous SGD performs well under heterogeneous computation times and converges comparably to the optimal Asynchronous SGD and Rennala SGD methods. In particular, in Figure 3a, we run the methods on a quadratic optimization task with $n = 1000$, assuming that worker i requires $\tau_i = \sqrt{i}$ seconds to compute one stochastic gradient, and ignoring other factors, including communication times. This experiment matches the discussion in Sections 1 and 2, where we see that Synchronous SGD is slow due to the stragglers with large τ_i; however, m-Synchronous SGD with $m = 10$, effectively ignoring stragglers, has the same convergence speed as the optimal methods despite computing one stochastic gradient per worker in every iteration. In Figure 3b, we evaluate on CIFAR-10 (Krizhevsky et al., 2009) under a setup with random computation times and equal means. The results show that Synchronous SGD converges as fast as Rennala SGD and is at most a factor of $1.45\times$ slower than Asynchronous SGD, as theoretically predicted in Section 3. In Section 8, we explain why a constant-factor gap of 1–$2\times$ is not an issue in real applications and environments.

8 Conclusion

In this paper, we revisit the classical Synchronous SGD method and its robust variant, m-Synchronous SGD, which can be interpreted as a method with a backup-worker strategy, and compare them with modern asynchronous optimization methods. To summarize, we show that synchronous methods are nearly optimal (i) under the fixed computation model (Assumption 2.2); (ii) under the random computation model (Assumption 3.1), at least when σ^2/ε is large and the times are truncated normal

or exponential (see also Section H.1); (iii) under partial participation (Assumption 5.4); and (iv) in several numerical examples (Section C.2). Our work provides theoretical evidence for the popularity of synchronous optimization methods in modern training.

One might argue that asynchronous methods can still be better by a constant or a $\log n$ factor under Assumptions 2.2, 3.1, 5.4, and in Figure 3b. We agree with this. However, in our NanoGPT experiment in a real homogeneous environment, with unstable computation times and custom implementations, we observe that the convergence rates are similar (see Section O.5). Moreover, in practice, engineers implement various optimization and caching tricks to speed up synchronization (Sergeev and Balso, 2018), which can potentially mitigate the constant and $\log n$ factor losses relative to asynchronous methods. On top of all this, Asynchronous SGD is not all-reduce or communication-friendly (Tyurin and Sivtsov, 2026), unlike synchronous methods.

Acknowledgements

The work was supported by the grant for research centers in the field of AI provided by the Ministry of Economic Development of the Russian Federation in accordance with the agreement 000000C313925P4F0002 and the agreement №139-10-2025-033.

References

Arjevani, Y., Carmon, Y., Duchi, J. C., Foster, D. J., Sekhari, A., and Sridharan, K. (2020). Second-order information in non-convex stochastic optimization: Power and limitations. In *Conference on Learning Theory*, pages 242–299. PMLR.

Arjevani, Y., Carmon, Y., Duchi, J. C., Foster, D. J., Srebro, N., and Woodworth, B. (2022). Lower bounds for non-convex stochastic optimization. *Mathematical Programming*, pages 1–50.

Barreto, M., Marchal, O., and Arbel, J. (2025). Optimal sub-gaussian variance proxy for truncated gaussian and exponential random variables. *Statistics & Probability Letters*, page 110555.

Boucheron, S., Lugosi, G., and Bousquet, O. (2003). Concentration inequalities. In *Summer school on machine learning*, pages 208–240. Springer.

Carmon, Y., Duchi, J. C., Hinder, O., and Sidford, A. (2020). Lower bounds for finding stationary points i. *Mathematical Programming*, 184(1):71–120.

Carmon, Y., Duchi, J. C., Hinder, O., and Sidford, A. (2021). Lower bounds for finding stationary points ii: first-order methods. *Mathematical Programming*, 185(1):315–355.

Chen, J., Pan, X., Monga, R., Bengio, S., and Jozefowicz, R. (2016). Revisiting distributed synchronous SGD. *arXiv preprint arXiv:1604.00981*.

Cohen, A., Daniely, A., Drori, Y., Koren, T., and Schain, M. (2021). Asynchronous stochastic optimization robust to arbitrary delays. *Advances in Neural Information Processing Systems*, 34:9024–9035.

Dutta, S., Joshi, G., Ghosh, S., Dube, P., and Nagpurkar, P. (2018). Slow and stale gradients can win the race: Error-runtime trade-offs in distributed SGD. In *International Conference on Artificial Intelligence and Statistics*, pages 803–812. PMLR.

Feyzmahdavian, H. R., Aytekin, A., and Johansson, M. (2016). An asynchronous mini-batch algorithm for regularized stochastic optimization. *IEEE Transactions on Automatic Control*, 61(12):3740–3754.

Grattafiori, A., Dubey, A., Jauhri, A., Pandey, A., Kadian, A., Al-Dahle, A., Letman, A., Mathur, A., Schelten, A., Vaughan, A., et al. (2024). The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Jordan, K., Jin, Y., Boza, V., Jiacheng, Y., Cesista, F., Newhouse, L., and Bernstein, J. (2024). Muon: An optimizer for hidden layers in neural networks.

Karpathy, A. (2022). NanoGPT.

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Koloskova, A., Stich, S. U., and Jaggi, M. (2022). Sharper convergence guarantees for Asynchronous SGD for distributed and federated learning. *Advances in Neural Information Processing Systems*, 35:17202–17215.

Konečný, J., McMahan, H. B., Yu, F. X., Richtárik, P., Suresh, A. T., and Bacon, D. (2016). Federated learning: Strategies for improving communication efficiency. *arXiv preprint arXiv:1610.05492*.

Krizhevsky, A., Hinton, G., et al. (2009). Learning multiple layers of features from tiny images. Technical report, University of Toronto, Toronto.

Lan, G. (2020). *First-order and stochastic optimization methods for machine learning*. Springer.

LeCun, Y., Cortes, C., and Burges, C. (2010). MNIST handwritten digit database. *ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist*, 2.

Lian, X., Huang, Y., Li, Y., and Liu, J. (2015). Asynchronous parallel stochastic gradient for nonconvex optimization. *Advances in Neural Information Processing Systems*, 28.

Maranjyan, A., Omar, O. S., and Richtárik, P. (2025a). MindFlayer SGD: Efficient parallel SGD in the presence of heterogeneous and random worker compute times. In *Conference on Uncertainty in Artificial Intelligence*.

Maranjyan, A. and Richtárik, P. (2025). Ringleader ASGD: The first Asynchronous SGD with optimal time complexity under data heterogeneity. *arXiv preprint arXiv:2509.22860*.

Maranjyan, A., Tyurin, A., and Richtárik, P. (2025b). Ringmaster ASGD: The first Asynchronous SGD with optimal time complexity. In *International Conference on Machine Learning*.

McMahan, H. B., Moore, E., Ramage, D., and y Arcas, B. A. (2016). Federated learning of deep networks using model averaging. *arXiv preprint arXiv:1602.05629*, 2:2.

Mishchenko, K., Bach, F., Even, M., and Woodworth, B. (2022). Asynchronous SGD beats minibatch SGD under arbitrary delays. *arXiv preprint arXiv:2206.07638*.

Mitliagkas, I., Zhang, C., Hadjis, S., and Ré, C. (2016). Asynchrony begets momentum, with an application to deep learning. In *2016 54th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pages 997–1004. IEEE.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al. (2019). Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Rajbhandari, S., Rasley, J., Ruwase, O., and He, Y. (2020). Zero: Memory optimizations toward training trillion parameter models. In *SC20: International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 1–16. IEEE.

Recht, B., Re, C., Wright, S., and Niu, F. (2011). Hogwild!: A lock-free approach to parallelizing stochastic gradient descent. *Advances in Neural Information Processing Systems*, 24.

Sergeev, A. and Balso, M. D. (2018). Horovod: Fast and easy distributed deep learning in TensorFlow. *arXiv preprint arXiv:1802.05799*.

Shoeybi, M., Patwary, M., Puri, R., LeGresley, P., Casper, J., and Catanzaro, B. (2019). Megatron-lm: Training multi-billion parameter language models using model parallelism. *arXiv preprint arXiv:1909.08053*.

Sra, S., Yu, A. W., Li, M., and Smola, A. (2016). Adadelay: Delay adaptive distributed stochastic optimization. In *Artificial Intelligence and Statistics*, pages 957–965. PMLR.

Stich, S. U. and Karimireddy, S. P. (2020). The error-feedback framework: SGD with delayed gradients. *Journal of Machine Learning Research*, 21(237):1–36.

Tsitsiklis, J., Bertsekas, D., and Athans, M. (1986). Distributed asynchronous deterministic and stochastic gradient optimization algorithms. *IEEE Transactions on Automatic Control*, 31(9):803–812.

Tyurin, A. (2025). Tight time complexities in parallel stochastic optimization with arbitrary computation dynamics. In *13th International Conference on Learning Representations*.

Tyurin, A., Pozzi, M., Ilin, I., and Richtárik, P. (2024). Shadowheart SGD: Distributed asynchronous SGD with optimal time complexity under arbitrary computation and communication heterogeneity. *Advances in Neural Information Processing Systems*, 37.

Tyurin, A. and Richtárik, P. (2023). Optimal time complexities of parallel stochastic optimization methods under a fixed computation model. *Advances in Neural Information Processing Systems*.

Tyurin, A. and Richtárik, P. (2024). On the optimal time complexities in decentralized stochastic asynchronous optimization. *Advances in Neural Information Processing Systems*, 37.

Tyurin, A. and Sivtsov, D. (2026). Birch SGD: A tree graph framework for local and asynchronous SGD methods. In *International Conference on Learning Representations*.

Vershynin, R. (2018). *High-dimensional probability: An introduction with applications in data science*, volume 47. Cambridge university press.

Contents

1 Introduction **1**

1.1 Contributions 3

2 Synchronous Methods Are Nearly Optimal under Fixed Computation **4**

3 Random Computation Model **5**

4 On the Optimal Selection of Active Workers **6**

5 Universal Computation Model **7**

5.1 Example: Optimization with stragglers 7

6 When Asynchronicity Might Be Needed **8**

7 Experimental Highlights **9**

8 Conclusion **9**

A Notations **15**

B Analysis with Communication Times **15**

C Universal Computation Model: More Examples **18**

C.1 Example: different constant powers 18

C.2 Numerical examples 18

D Proof of Theorem 2.3 **19**

E Proof of Theorem 3.2 **20**

F Proof of Theorem B.3 **21**

G Proof of Theorem B.4 **22**

H Proof of Theorem 3.3 **24**

H.1 More theoretical examples 26

I Proof of Proposition 4.1 **27**

J Proof of Theorems 5.3 **27**

K Proof of Theorem 5.5 **28**

L Proof of Proposition 4.2 **28**

M A Theoretical Example Where Asynchronicity Is Needed **29**

N NanoGPT Benchmark **30**

O Extra Experiments and Details **31**

O.1 Quadratic optimization problem with heterogeneous times 31

O.2 Quadratic optimization with increasing number of workers 33

O.3 Quadratic optimization with random computational times 33

O.4 Small-Scale Machine Learning Task 36

O.5 NanoGPT experiments 40

A Notations

Table 1: List of notations used throughout the paper.

Notation	Meaning
$\mathbb{R}_+$	Denotes $[0, \infty)$.
$[n]$	Denotes a finite set $\{1, \ldots, n\}$.
$\|x\|$	Euclidean norm of a vector x.
$g = \mathcal{O}(f)$	There exists $C > 0$ such that $g(z) \leq C\, f(z)$ for all $z \in \mathcal{Z}$.
$g = \Omega(f)$	There exists $C > 0$ such that $g(z) \geq C\, f(z)$ for all $z \in \mathcal{Z}$.
$g = \Theta(f)$	There exist $C_1, C_2 > 0$ such that $C_1 f(z) \leq g(z) \leq C_2 f(z)$ for all $z \in \mathcal{Z}$.
$\tilde{\mathcal{O}}, \tilde{\Omega}$, and $\tilde{\Theta}$	The same as $\mathcal{O}$, Ω, and Θ, but up to logarithmic factors.
$\mathbb{E}[\cdot]$	Full expectation.

B Analysis with Communication Times

In the main part, we analyze the methods assuming that communication times are negligible. We now extend the results from Sections 2 and 3 to scenarios where communication takes times. The main conclusions remain unchanged; however, the resulting time complexity becomes more involved, so for clarity, we defer the full results and analysis to the appendix.

Assumption B.1. For all $i \in [n]$, the i^{th} worker requires $\bar{\tau}_i$ seconds to calculate a stochastic gradient and $\bar{\kappa}_i$ seconds to send/receive a vector to/from the server, where $\bar{\tau}_i$ is a (τ_i, R)–sub-exponential random variable and $\bar{\kappa}_i$ is a (κ_i, R)–sub-exponential random variable. We assume that the sampled times $\{\bar{\tau}_i\}$, $\{\bar{\kappa}_i\}$, and the stochastic random variables $\{\xi_i\}$ are statistically independent.

Instead of Algorithm 3, we analyze its minor modification with mini-batching, which helps to reduce the number of synchronizations and improve communication efficiency.

Algorithm 4 m-Synchronous SGD with mini-batching

1: **Input:** point $x^0 \in \mathbb{R}^d$, stepsize γ, # of workers $m \in [n]$, batch size B
2: **for** $k = 0, 1, \ldots, K-1$ **do**
3: Each idle worker starts computing stochastic gradients at x^k
4: Aggregate from the first m workers in this iteration:

$$g^k = \frac{1}{m} \sum_{i \in S_k} \frac{1}{B} \sum_{j=1}^{B} \nabla f(x^k; \xi_{ij}^k), \;\; S_k \subseteq [n], \;\; |S_k| = m$$

5: Update $x^{k+1} = x^k - \gamma g^k$
6: Start a non-blocking broadcast of x^{k+1} to all workers
7: Discard any gradients for iteration k that arrive after the update
8: **end for**

Notice that Algorithm 4 reduces to Algorithm 3 when $B = 1$. We now prove the following iteration complexity result, an auxiliary result:

Theorem B.2 (Iteration Complexity; (Lan, 2020))**.** *Under Assumptions 1.1, 1.2, and 1.3, let* $\gamma = \min\left\{\frac{1}{2L}, \frac{\varepsilon m B}{4L\sigma^2}\right\}$ *in Algorithm 4. Then* $\frac{1}{K}\sum_{k=0}^{K-1} \mathbb{E}\left[\left\|\nabla f(x^k)\right\|^2\right] \leq \varepsilon$ *after*

$$K := \left\lceil 16 \max\left\{\frac{L\Delta}{\varepsilon}, \frac{\sigma^2 L \Delta}{Bm\varepsilon^2}\right\}\right\rceil \tag{16}$$

iterations, where $m \in [n]$ *and* $B \geq 1$ *are arbitrary numbers.*

We now extend Theorem 3.2:

Algorithm 5 Minimizing the Objective $T_{\text{det}}(m)$

Require: Workers $i \in [n]$ with parameters (κ_i, τ_i), ratio σ^2/ε
Ensure: Optimal m

```
OPT = +∞
S* = ∅
Let K_1 < K_2 < ··· < K_L be the distinct values among {κ_i : i ∈ [n]}
for ℓ = 1, ..., L do
    P_ℓ = {i ∈ [n] : κ_i ≤ K_ℓ}
    Sort P_ℓ by increasing τ_i: τ_{j_1} ≤ τ_{j_2} ≤ ··· ≤ τ_{j_{|P_ℓ|}}.
    for m = 1, ..., |P_ℓ| do
        β_m = max{1, σ^2/(mε)}
        V = max{ max_{p∈[m]} κ_{j_p}, β_m τ_{j_m} }
        if V < OPT then
            OPT = V
            S* = {j_1, ..., j_m}
        end if
    end for
end for
return m = |S*|
```

Theorem B.3 (Time Complexity)**.** *Under the setting of Theorem B.2, additionally assume that Assumption B.1 holds. Then, the time complexity of Algorithm 4 to find an ε–stationary point is at most*

$$T_{\text{rand}} := \min_{S\subseteq[n],|S|=m} \left[\sum_{k=0}^{K-1}\left(\max_{i\in S}\sum_{j=1}^{B}\bar{\tau}_{k,ij} + \max_{i\in S}\bar{\kappa}_{k,i1} + \max_{i\in S}\bar{\kappa}_{k,i2}\right)\right] \tag{17}$$

seconds, where $\bar{\tau}_{k,ij}$ are (τ_i, R)–sub-exponential random variables (independent copies) and $\bar{\kappa}_{k,i1}$ and $\bar{\kappa}_{k,i2}$ are (κ_i, R)–sub-exponential random variables (independent copies). Moreover, we have

$$\begin{aligned}\mathbb{E}\left[T_{\text{rand}}\right] = \mathcal{O}\Bigg(\frac{L\Delta}{\varepsilon}\Bigg(&\min_{S\subseteq[n],|S|=m}\left[\max_{i\in S}B\tau_i + \max_{i\in S}\kappa_i\right] \\ &+ R\max\left\{\log(2n), \sqrt{B\log(2n)}\right\}\Bigg)\max\left\{1, \frac{\sigma^2}{Bm\varepsilon}\right\}\Bigg),\end{aligned} \tag{18}$$

for all $m \in [n]$ and $B \geq 1$ in Algorithm 4. Finally, taking $B = \max\left\{1, \left\lceil\frac{\sigma^2}{m\varepsilon}\right\rceil\right\}$,

$$\mathbb{E}\left[T_{\text{rand}}\right] = \mathcal{O}\left(T_{\text{det}}(m) + \frac{L\Delta}{\varepsilon}R\max\left\{\log(2n), \sqrt{\max\left\{1, \frac{\sigma^2}{m\varepsilon}\right\}\log(2n)}\right\}\right) \tag{19}$$

for all $m \in [n]$, where

$$T_{\text{det}}(m) := \frac{L\Delta}{\varepsilon}\min_{S\subseteq[n],|S|=m}\left[\max\left\{\max_{i\in S}\kappa_i, \max_{i\in S}\tau_i\max\left\{1, \frac{\sigma^2}{m\varepsilon}\right\}\right\}\right].$$

Similarly to Theorem 3.2, we get the deterministic term plus an additional "noise" term proportional to R. In the deterministic term, we have a free parameter $m \in [n]$. In order to find an optimal m, we can use Algorithm 5 and prove the following result:

Theorem B.4 (Time Complexity with a Good Choice of m)**.** *Consider the setting of Theorem B.3. Using Algorithm 5, we can find $m^* \in [n]$ that minimizes $T_{\text{det}}(m)$. Moreover, $\min_{m\in[n]} T_{\text{det}}(m) = \mathcal{O}\left(\log(n+1)T_{\text{optimal}}\right)$, and*

$$\mathbb{E}\left[T_{\text{rand}}\right] = \mathcal{O}\left(\log(n+1)T_{\text{optimal}} + \frac{L\Delta}{\varepsilon}R\max\left\{\log(2n), \sqrt{\max\left\{1, \frac{\sigma^2}{m^*\varepsilon}\right\}\log(2n)}\right\}\right)$$

for $m = m^*$ *and* $B = \max\left\{1, \left\lceil \frac{\sigma^2}{m\varepsilon} \right\rceil\right\}$, *where*

$$T_{\text{optimal}} = \tilde{\Theta}\left(\frac{L\Delta}{\varepsilon} \min_{m\in[n]} \max\left\{\max\{\kappa_{\pi_m}, \tau_{\pi_m}\}, \frac{\sigma^2}{\varepsilon}\left(\sum_{i=1}^{m} \frac{1}{\tau_{\pi_i}}\right)^{-1}\right\}\right)$$

is the optimal time complexity in the centralized setup with deterministic computation and communication times (Tyurin et al., 2024; Tyurin and Richtárik, 2024) and π *is a permutation that sorts* $\{\max\{\kappa_i, \tau_i\}\}_{i\in[n]} : \max\{\kappa_{\pi_1}, \tau_{\pi_1}\} \leq \cdots \leq \max\{\kappa_{\pi_n}, \tau_{\pi_n}\}$.

Theorem B.4 shows that the time complexity of Algorithm 4 is nearly optimal up to a logarithmic factor when the times are non-random (e.g., $R = 0$ in Assumption B.1), a result similar to Theorem 2.3. In more general setting, when $R > 0$, Algorithm 4 achieves $\mathcal{O}\left(\log(n+1)T_{\text{optimal}}\right)$ when $\min_{m\in[n]} T_{\text{det}}(m)$ dominates $\frac{L\Delta}{\varepsilon} R \max\left\{\log(2n), \sqrt{\max\left\{1, \frac{\sigma^2}{m^*\varepsilon}\right\}\log(2n)}\right\}$, which is true when ε is small, $\{\kappa_i\}$ are large, σ is large, or R is small, a result similar to Corollary 3.4.

C Universal Computation Model: More Examples

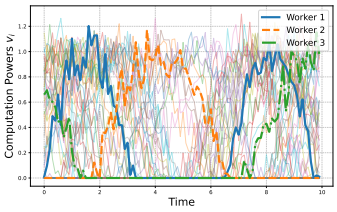


Figure 4: Computation powers for 50 workers, where every worker chaotically stops participating in the computation process (its computation power is zero for some time intervals). We generate worker powers as $v_i(t_k) = \max\{\sin(a_i t_k + s_i) + \varepsilon_{ik},\, 0\}, \varepsilon_{ik} \sim \mathcal{N}(0, 0.1^2), s_i \sim \text{Unif}(0, 2\pi), a_i \sim \text{Unif}(0.5, 1)$, on a uniform grid $t_k = 0.1k$, and define $v_i(t)$ by linear interpolation.

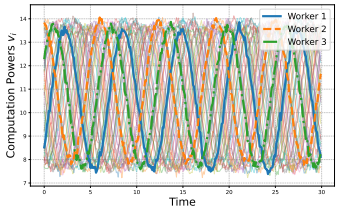


Figure 5: Computation powers for 50 workers, where every worker has a periodic computation power. We generate worker powers $v_i(t_k) = \max\{s_i + 3\sin(t_k + \phi_i) + \varepsilon_{ik},\, 0.1\}, \varepsilon_{ik} \sim \mathcal{N}(0, 0.1^2), \quad s_i \sim \text{Unif}(10.5, 11.0), \phi_i \sim \text{Unif}(0, 2\pi)$, on a uniform grid $t_k = 0.1k$, and define $v_i(t)$ by linear interpolation.

In this section, we consider some examples of universal computation models from Section 5.

C.1 Example: different constant powers

In the case of constant powers, when $v_i(t) = v_i \in \mathbb{R}_+$ for all $t \geq 0$, Theorem 5.3 reduces to Theorem 2.3. Indeed, $N_i(t_0, t_1) = \lfloor (t_1 - t_0) v_i \rfloor$ and $\max_{S \subseteq [n], |S|=m} \min_{i \in S} N_i(\bar{t}_k, t) = \max_{S \subseteq [n], |S|=m} \min_{i \in S} \lfloor (t - \bar{t}_k) v_i \rfloor$. W.l.o.g., assume that $v_1 \geq \cdots \geq v_n$, then $\max_{S \subseteq [n], |S|=m} \min_{i \in S} N_i(\bar{t}_k, \bar{t}_{k+1}) = 2$ for $\bar{t}_{k+1} = \bar{t}_k + \frac{2}{v_m} = \frac{2(k+1)}{v_m}$. Due to Theorem 5.3, the method converges after $\bar{t}_{\bar{K}} = \mathcal{O}\left(1/v_m \max\left\{L\Delta/\varepsilon, \sigma^2 L\Delta/m\varepsilon^2\right\}\right)$ seconds, and we get the same result as in Theorem 2.3 up to a constant factor if we take $m = \arg\min_{m \in [n]} \left[1/v_m \cdot \max\left\{1, \sigma^2/m\varepsilon\right\}\right]$ and recall that $\tau_m \equiv 1/v_m$ in the scenario with the fixed powers. Similarly, one can show that the lower bound in Theorem 5.2 is $\underline{t}_{\underline{K}} = (2)$ which corresponds to the optimal time complexity (2). Here, as expected, $\bar{t}_{\bar{K}}$ is tight up to logarithmic factors for an appropriate choice of m.

C.2 Numerical examples

Since $\bar{t}_{\bar{K}}$ and $\underline{t}_{\underline{K}}$ are non-explicit, we generate several practical scenarios and *numerically* evaluate the gap $\bar{t}_{\bar{K}}/\underline{t}_{\underline{K}}$ between the lower bound and the time complexity of m-Synchronous SGD (Algorithm 3).

We start with Figure 4, where every worker participates in the optimization process chaotically. We numerically evaluate[6] $\bar{t}_{\bar{K}}$ and $\underline{t}_{\underline{K}}$ from Theorems 5.3 and 5.2 in two noise regimes. We fix $n = 50$ and $L\Delta/\varepsilon = 1$ and show that $\bar{t}_{\bar{K}}/\underline{t}_{\underline{K}} \leq 1.52$ for $\sigma^2/\varepsilon = 100$ and $m = 15$, and that $\bar{t}_{\bar{K}}/\underline{t}_{\underline{K}} \leq 1.85$ for $\sigma^2/\varepsilon = 1000$ and $m = 14$. Thus, the difference is only a factor of 1.85. In Figure 5, we consider a slightly different scenario with periodic computation powers. We fix $n = 50$ and $L\Delta/\varepsilon = 1$ and show that $\bar{t}_{\bar{K}}/\underline{t}_{\underline{K}} \leq 1.11$ for $\sigma^2/\varepsilon = 100$ and $m = 49$, and that $\bar{t}_{\bar{K}}/\underline{t}_{\underline{K}} \leq 1.37$ for $\sigma^2/\varepsilon = 1000$ and $m = 50$.

Notice that the computation powers in Figures 4 and 5 are very chaotic, and intuitively, the time complexity of m-Synchronous SGD should be much worse than the time complexity of asynchronous methods. Nevertheless, with a proper choice of m in Algorithm 3, it is possible to obtain a time complexity that is worse by at most a factor of 2. See our discussion in Section 8 why a minor constant factor is not a problem in practice.

D Proof of Theorem 2.3

Theorem 2.3 (Time Complexity under Fixed Computation Model)**.** *Under the setting of Theorem 2.1, additionally assume that Assumption 2.2 holds. Then, the time complexity of Algorithm 3 to find an ε–stationary point is*

$$T_{\text{sync}} := \frac{16L\Delta}{\varepsilon} \min_{m\in[n]} \left[\tau_m \max\left\{1, \frac{\sigma^2}{m\varepsilon}\right\}\right] \tag{3}$$

if m is a minimizer of $\min_{m\in[n]} \left[\tau_m \max\left\{1, \frac{\sigma^2}{m\varepsilon}\right\}\right]$. More importantly, the result is nearly optimal:

$$T_{\text{sync}} = \mathcal{O}\left(T_{\text{optimal}} \times \log(n+1)\right). \tag{4}$$

Proof. The first term in the theorem follows from the fact that every iteration takes at most τ_m seconds; thus, the total time is at most $\tau_m \times 16 \max\left\{\frac{L\Delta}{\varepsilon}, \frac{\sigma^2 L\Delta}{m\varepsilon^2}\right\}$ seconds due to Theorem 2.1.

Let $\frac{\sigma^2}{\varepsilon} \leq 1$. Then

$$\min_{m\in[n]} \left[\tau_m \max\left\{\frac{L\Delta}{\varepsilon}, \frac{\sigma^2 L\Delta}{m\varepsilon^2}\right\}\right] = \tau_1 \frac{L\Delta}{\varepsilon}$$

and

$$T_{\text{optimal}} := \Theta\left(\min_{m\in[n]} \left[\left(\frac{1}{m}\sum_{i=1}^{m}\frac{1}{\tau_i}\right)^{-1}\left(\max\left\{\frac{L\Delta}{\varepsilon}, \frac{\sigma^2 L\Delta}{m\varepsilon^2}\right\}\right)\right]\right) = \Theta\left(\tau_1 \frac{L\Delta}{\varepsilon}\right).$$

since both terms are minimized with $m = 1$. Thus, the theorem is correct. Further we assume that $\frac{\sigma^2}{\varepsilon} > 1$.

Let us define m^* as the smallest minimum of $\min_{m\in[n]} \left[\tau_m \max\left\{1, \frac{\sigma^2}{m\varepsilon}\right\}\right]$. Then

$$\tau_m \max\left\{1, \frac{\sigma^2}{m\varepsilon}\right\} \geq \tau_{m^*} \max\left\{1, \frac{\sigma^2}{m^*\varepsilon}\right\} \tag{20}$$

for all $m \in [n]$.

The optimal time complexity is

$$T_{\text{optimal}} := \Theta\left(\min_{m\in[n]} \left[\left(\frac{1}{m}\sum_{i=1}^{m}\frac{1}{\tau_i}\right)^{-1}\left(\max\left\{\frac{L\Delta}{\varepsilon}, \frac{\sigma^2 L\Delta}{m\varepsilon^2}\right\}\right)\right]\right). \tag{21}$$

Let us define $\bar{m}$ as the smallest minimizer in (21). Then

$$T_{\text{optimal}} = \Theta\left(\left(\frac{1}{\bar{m}}\sum_{i=1}^{\bar{m}}\frac{1}{\tau_i}\right)^{-1}\left(\max\left\{\frac{L\Delta}{\varepsilon}, \frac{\sigma^2 L\Delta}{\bar{m}\varepsilon^2}\right\}\right)\right). \tag{22}$$

and $\bar{m} \leq \left\lceil \frac{\sigma^2}{\varepsilon} \right\rceil \leq \frac{2\sigma^2}{\varepsilon}$ since $\left(\frac{1}{m}\sum_{i=1}^{m}\frac{1}{\tau_i}\right)^{-1}$ is a non-decreasing sequence as a function of m and $\frac{\sigma^2}{\varepsilon} > 1$. Moreover, $\bar{m} \geq \max\{m^* - 1, 1\}$. It is true if $m^* = 1$. Assuming $m^* > 1$, let us consider two cases to prove $\bar{m} \geq \max\{m^* - 1, 1\}$:

i) if $1 \leq \frac{\sigma^2}{m^*\varepsilon}$, then

$$g(m) := \left(\frac{1}{m}\sum_{i=1}^{m}\frac{1}{\tau_i}\right)^{-1}\left(\max\left\{\frac{L\Delta}{\varepsilon}, \frac{\sigma^2 L\Delta}{m\varepsilon^2}\right\}\right) = \left(\sum_{i=1}^{m}\frac{1}{\tau_i}\right)^{-1}\left(\frac{\sigma^2 L\Delta}{\varepsilon^2}\right)$$

for all $m \leq m^*$, and $g(m)$ can only potentially decrease if we take $m > m^* \geq \max\{m^* - 1, 1\}$.

ii) if $1 > \frac{\sigma^2}{m^*\varepsilon}$, then $1 < \frac{\sigma^2}{(m^*-1)\varepsilon}$ because m^* is the smallest minimum of $\min_{m\in[n]} \left[\tau_m \max\left\{1, \frac{\sigma^2}{m\varepsilon}\right\}\right]$.

Similarly to the previous case, since $1 < \frac{\sigma^2}{(m^*-1)\varepsilon}$, we can conclude that $\bar{m} \geq m^* - 1 \geq \max\{m^* - 1, 1\}$.

Due to (20), for all $m \in [\bar{m}]$,

$$\frac{2\tau_m\sigma^2}{m\varepsilon} \overset{m\leq\bar{m}\leq\frac{2\sigma^2}{\varepsilon}}{\geq} \tau_m \max\left\{1, \frac{\sigma^2}{m\varepsilon}\right\} \geq \tau_{m^*} \max\left\{1, \frac{\sigma^2}{m^*\varepsilon}\right\} \geq \frac{\tau_{m^*}\sigma^2}{m^*\varepsilon}$$

and

$$\tau_m \geq \frac{\tau_{m^*} m}{2m^*}.$$

Substituting this bound to (22), we get

$$T_{\text{optimal}} = \Omega\left(\left(\frac{m^*}{\tau_{m^*}\bar{m}}\sum_{i=1}^{\bar{m}}\frac{1}{i}\right)^{-1}\left(\max\left\{\frac{L\Delta}{\varepsilon}, \frac{\sigma^2 L\Delta}{\bar{m}\varepsilon^2}\right\}\right)\right).$$

Using the well-known inequality $\sum_{i=1}^{\bar{m}} \frac{1}{i} \leq 1 + \log\bar{m} \leq 2\log(n+1)$,

$$T_{\text{optimal}} = \Omega\left(\frac{\tau_{m^*}\bar{m}}{m^*\log(n+1)}\left(\max\left\{\frac{L\Delta}{\varepsilon}, \frac{\sigma^2 L\Delta}{\bar{m}\varepsilon^2}\right\}\right)\right) = \Omega\left(\frac{1}{\log(n+1)} \times \tau_{m^*}\left(\max\left\{\frac{\bar{m}L\Delta}{m^*\varepsilon}, \frac{\sigma^2 L\Delta}{m^*\varepsilon^2}\right\}\right)\right).$$

It is left to recall that $\bar{m} \geq \max\{m^* - 1, 1\} \geq \frac{m^*}{2}$ to get

$$T_{\text{optimal}} = \Omega\left(\frac{1}{\log(n+1)} \times \tau_{m^*}\left(\max\left\{\frac{L\Delta}{\varepsilon}, \frac{\sigma^2 L\Delta}{m^*\varepsilon^2}\right\}\right)\right) = \Omega\left(\frac{1}{\log(n+1)} \times \min_{m\in[n]}\left[\tau_m \max\left\{\frac{L\Delta}{\varepsilon}, \frac{\sigma^2 L\Delta}{m\varepsilon^2}\right\}\right]\right)$$

and the result of the theorem. □

E Proof of Theorem 3.2

Theorem 3.2 (Time Complexity under Random Computation Model)**.** *Under the setting of Theorem 2.1, additionally assume that Assumption 3.1 holds. Then, the time complexity of Algorithm 3 to find an ε–stationary point is at most*

$$T_{\text{rand}} := \sum_{k=0}^{K-1} \max_{i\in[m]} \bar{\tau}_{k,i} \tag{5}$$

seconds, where $\bar{\tau}_{k,i}$ is a (τ_i, R)–sub-exponential random variable (independent copy). Moreover, we have

$$\mathbb{E}\left[T_{\text{rand}}\right] = \mathcal{O}\left(\frac{L\Delta}{\varepsilon}\left(\tau_m + R\log(n)\right)\max\left\{1, \frac{\sigma^2}{m\varepsilon}\right\}\right) \tag{6}$$

for all $m \in [n]$ in Algorithm 3.

Proof. We run Algorithm 3 at most

$$K := \left\lceil 16\max\left\{\frac{L\Delta}{\varepsilon}, \frac{\sigma^2 L\Delta}{m\varepsilon^2}\right\}\right\rceil$$

iterations. Let t_k be the time when the k^{th} iteration is finished. Workers $1, \ldots, m$ finish computing the first stochastic gradients after at most

$$\max_{i\in[m]} \bar{\tau}_{0,i} \geq t_1$$

seconds. Using mathematical induction, assume that $\sum_{k=0}^{K-1} \max_{i\in[m]} \bar{\tau}_{k,i} \geq t_K$. Iteration $K+1$ starts at time t_K. Let $\bar{k}_i \in \{0, \ldots, K\}$ be the last iteration when worker i finishes before the end of this iteration. Then worker i starts calculating the first stochastic gradient after time t_k, after at most

$$t_{\bar{k}_i} + \sum_{k=\bar{k}_i}^{K-1} \bar{\tau}_{k,i}$$

seconds. Therefore, workers $1, \ldots, m$ finish their first stochastic gradient after time t_k, after at most

$$\max_{i\in[m]}\left(t_{\bar{k}_i}+\sum_{k=\bar{k}_i}^{K-1}\bar{\tau}_{k,i}+\bar{\tau}_{K,i}\right)\leq\max_{i\in[m]}\left(\sum_{k=0}^{\bar{k}_i-1}\max_{i\in[m]}\bar{\tau}_{k,i}+\sum_{k=\bar{k}_i}^{K}\bar{\tau}_{k,i}\right)$$
$$\leq\max_{i\in[m]}\left(\sum_{k=0}^{\bar{k}_i-1}\max_{i\in[m]}\bar{\tau}_{k,i}+\sum_{k=\bar{k}_i}^{K}\max_{i\in[m]}\bar{\tau}_{k,i}\right)=\sum_{k=0}^{K}\max_{i\in[m]}\bar{\tau}_{k,i}$$

seconds, meaning that $t_{K+1}\leq\sum_{k=0}^{K}\max_{i\in[m]}\bar{\tau}_{k,i}$.

It is left to prove the second part of the theorem. We have

$$\mathbb{E}\left[\max_{i\in[m]}\left(\bar{\tau}_{k,i}-\tau_i\right)\right]=\frac{1}{\lambda}\mathbb{E}\left[\log\left(\exp\left(\lambda\max_{i\in[m]}\left(\bar{\tau}_{k,i}-\tau_i\right)\right)\right)\right]$$

for all $\lambda>0$. Since $\log x$ is concave,

$$\mathbb{E}\left[\max_{i\in[m]}\left(\bar{\tau}_{k,i}-\tau_i\right)\right]\leq\frac{1}{\lambda}\log\left(\mathbb{E}\left[\exp\left(\lambda\max_{i\in[m]}\left(\bar{\tau}_{k,i}-\tau_i\right)\right)\right]\right)=\frac{1}{\lambda}\log\left(\mathbb{E}\left[\max_{i\in[m]}\exp\left(\lambda\left(\bar{\tau}_{k,i}-\tau_i\right)\right)\right]\right)$$
$$\leq\frac{1}{\lambda}\log\left(\mathbb{E}\left[\sum_{i=1}^{m}\exp\left(\lambda\left(\bar{\tau}_{k,i}-\tau_i\right)\right)\right]\right)\leq\frac{1}{\lambda}\log\left(\sum_{i=1}^{m}\mathbb{E}\left[\exp\left(\lambda\left|\bar{\tau}_{k,i}-\tau_i\right|\right)\right]\right).$$

Taking $\lambda=\frac{1}{R}$, we get

$$\mathbb{E}\left[\max_{i\in[m]}\left(\bar{\tau}_{k,i}-\tau_i\right)\right]\leq R\log\left(\sum_{i=1}^{m}\mathbb{E}\left[\exp\left(\left|\bar{\tau}_{k,i}-\tau_i\right|/R\right)\right]\right)\leq R\log(2m)$$

since $\bar{\tau}_{k,i}$ is a (τ_i,R)–sub-exponential random variable. It is left to notice that

$$\max_{i\in[m]}\bar{\tau}_{k,i}\leq\max_{i\in[m]}\left(\bar{\tau}_{k,i}-\tau_i\right)+\max_{i\in[m]}\tau_i.$$

□

F Proof of Theorem B.3

Theorem B.3 (Time Complexity)**.** *Under the setting of Theorem B.2, additionally assume that Assumption B.1 holds. Then, the time complexity of Algorithm 4 to find an ε–stationary point is at most*

$$T_{\text{rand}}:=\min_{S\subseteq[n],|S|=m}\left[\sum_{k=0}^{K-1}\left(\max_{i\in S}\sum_{j=1}^{B}\bar{\tau}_{k,ij}+\max_{i\in S}\bar{\kappa}_{k,i1}+\max_{i\in S}\bar{\kappa}_{k,i2}\right)\right]\tag{17}$$

seconds, where $\bar{\tau}_{k,ij}$ are (τ_i,R)–sub-exponential random variables (independent copies) and $\bar{\kappa}_{k,i1}$ and $\bar{\kappa}_{k,i2}$ are (κ_i,R)–sub-exponential random variables (independent copies). Moreover, we have

$$\mathbb{E}\left[T_{\text{rand}}\right]=\mathcal{O}\left(\frac{L\Delta}{\varepsilon}\left(\min_{S\subseteq[n],|S|=m}\left[\max_{i\in S}B\tau_i+\max_{i\in S}\kappa_i\right]\right.\right.$$
$$\left.\left.+R\max\left\{\log(2n),\sqrt{B\log(2n)}\right\}\right)\max\left\{1,\frac{\sigma^2}{Bm\varepsilon}\right\}\right),\tag{18}$$

for all $m\in[n]$ and $B\geq 1$ in Algorithm 4. Finally, taking $B=\max\left\{1,\left\lceil\frac{\sigma^2}{m\varepsilon}\right\rceil\right\}$,

$$\mathbb{E}\left[T_{\text{rand}}\right]=\mathcal{O}\left(T_{\text{det}}(m)+\frac{L\Delta}{\varepsilon}R\max\left\{\log(2n),\sqrt{\max\left\{1,\frac{\sigma^2}{m\varepsilon}\right\}\log(2n)}\right\}\right)\tag{19}$$

for all $m\in[n]$, where

$$T_{\text{det}}(m):=\frac{L\Delta}{\varepsilon}\min_{S\subseteq[n],|S|=m}\left[\max\left\{\max_{i\in S}\kappa_i,\max_{i\in S}\tau_i\max\left\{1,\frac{\sigma^2}{m\varepsilon}\right\}\right\}\right].$$

Proof. We fix any $S \subseteq [n]$ such that $|S| = m$. Then, the proof of (17) is the same in the proof of Theorem 3.2. There are only two differences: i) every worker calculates B stochastic gradients instead of one stochastic gradient; ii) worker i sends aggregated stochastic gradients, which takes $\bar{\kappa}_{k,i1}$ seconds and receives a new iterate from the server, which takes $\bar{\kappa}_{k,i2}$ seconds. Then, it is left to use that $S \subseteq [n]$ is an arbitrary subset such that $|S| = m$.

We now prove (18), which also almost repeats the proof of Theorem 3.2. Notice that

$$T_{\text{rand}} \leq \sum_{k=0}^{K-1} \left(\max_{i \in S} B\tau_i + 2 \max_{i \in S} \kappa_i \right) + \sum_{k=0}^{K-1} \left(\max_{i \in S} \sum_{j=1}^{B} (\bar{\tau}_{k,ij} - \tau_i) + \max_{i \in S} (\bar{\kappa}_{k,i1} - \kappa_i) + \max_{i \in S} (\bar{\kappa}_{k,i2} - \kappa_i) \right)$$

for all $S \subseteq [n]$ such that $|S| = m$. Similarly to the proof of Theorem 3.2, one can easily show that $\mathbb{E}\left[\max_{i \in S} (\bar{\kappa}_{k,i1} - \kappa_i) + \max_{i \in S} (\bar{\kappa}_{k,i2} - \kappa_i) \right] \leq 2R \log(2n)$. Also,

$$\mathbb{E}\left[\max_{i \in S} \sum_{j=1}^{B} (\bar{\tau}_{k,ij} - \tau_i) \right] \leq \frac{1}{\lambda} \log \left(\sum_{i=1}^{m} \mathbb{E}\left[\exp\left(\lambda \sum_{j=1}^{B} (\bar{\tau}_{k,ij} - \tau_i) \right) \right] \right).$$

Since the random variables are i.i.d.,

$$\mathbb{E}\left[\max_{i \in S} \sum_{j=1}^{B} (\bar{\tau}_{k,ij} - \tau_i) \right] \leq \frac{1}{\lambda} \log \left(\sum_{i=1}^{m} \left(\mathbb{E}\left[\exp\left(\lambda (\bar{\tau}_{k,i1} - \tau_i) \right) \right] \right)^B \right).$$

Since $\bar{\tau}_{k,i1}$ is a sub-exponential random variable, there exist a universal constant $c_1 \geq 1$ such that

$$\mathbb{E}\left[\max_{i \in S} \sum_{j=1}^{B} (\bar{\tau}_{k,ij} - \tau_i) \right] \leq \frac{1}{\lambda} \log \left(m \exp\left(c_1^2 \lambda^2 R^2 B \right) \right) = \frac{\log m}{\lambda} + c_1^2 \lambda R^2 B$$

for all $\lambda \leq \frac{1}{c_1 R}$ (Vershynin, 2018). Taking $\lambda = \min\left\{ \frac{1}{c_1 R}, \frac{\sqrt{\log(2m)}}{c_1 R \sqrt{B}} \right\}$,

$$\mathbb{E}\left[\max_{i \in S} \sum_{j=1}^{B} (\bar{\tau}_{k,ij} - \tau_i) \right] = \mathcal{O}\left(R \max\left\{ \log(2m), \sqrt{B \log(2m)} \right\} \right).$$

Thus,

$$\mathbb{E}\left[T_{\text{rand}}\right] = \mathcal{O}\left(\frac{L\Delta}{\varepsilon} \left(\left[\max_{i \in S} B\tau_i + \max_{i \in S} \kappa_i \right] + R \max\left\{ \log(2n), \sqrt{B \log(2n)} \right\} \right) \max\left\{ 1, \frac{\sigma^2}{Bm\varepsilon} \right\} \right)$$

for all $S \subseteq [n]$ such that $|S| = m$. It is left to take the min over all $S \subseteq [n]$ such that $|S| = m$. □

G Proof of Theorem B.4

Theorem B.4 (Time Complexity with a Good Choice of m)**.** *Consider the setting of Theorem B.3. Using Algorithm 5, we can find $m^* \in [n]$ that minimizes $T_{\text{det}}(m)$. Moreover, $\min_{m \in [n]} T_{\text{det}}(m) = \mathcal{O}\left(\log(n+1) T_{\text{optimal}}\right)$, and*

$$\mathbb{E}\left[T_{\text{rand}}\right] = \mathcal{O}\left(\log(n+1) T_{\text{optimal}} + \frac{L\Delta}{\varepsilon} R \max\left\{ \log(2n), \sqrt{\max\left\{ 1, \frac{\sigma^2}{m^* \varepsilon} \right\} \log(2n)} \right\} \right)$$

for $m = m^*$ and $B = \max\left\{ 1, \left\lceil \frac{\sigma^2}{m\varepsilon} \right\rceil \right\}$, where

$$T_{\text{optimal}} = \tilde{\Theta}\left(\frac{L\Delta}{\varepsilon} \min_{m \in [n]} \max\left\{ \max\{\kappa_{\pi_m}, \tau_{\pi_m}\}, \frac{\sigma^2}{\varepsilon} \left(\sum_{i=1}^{m} \frac{1}{\tau_{\pi_i}} \right)^{-1} \right\} \right)$$

is the optimal time complexity in the centralized setup with deterministic computation and communication times (Tyurin et al., 2024; Tyurin and Richtárik, 2024) and π is a permutation that sorts $\{\max\{\kappa_i, \tau_i\}\}_{i\in[n]} : \max\{\kappa_{\pi_1}, \tau_{\pi_1}\} \leq \cdots \leq \max\{\kappa_{\pi_n}, \tau_{\pi_n}\}$.

Proof. Define

$$\widehat{T}_{\text{det}}(m) := \min_{S\subseteq[n],|S|=m} \max\left\{\max_{i\in S}\kappa_i, \beta_m \max_{i\in S}\tau_i\right\}, \qquad \beta_m := \max\left\{1, \frac{\sigma^2}{m\varepsilon}\right\}.$$

For each $\ell \in [L]$,

$$P_\ell = \{i \in [n] : \kappa_i \leq K_\ell\},$$

which also defined in Algorithm 5. After sorting P_ℓ by increasing τ_i, write

$$\tau_{j_1} \leq \tau_{j_2} \leq \cdots \leq \tau_{j_{|P_\ell|}}.$$

For every $m \leq |P_\ell|$, define $S_{\ell,m} := \{j_1, \ldots, j_m\}$. The value computed by the algorithm in every inner loop of Algorithm 5 is

$$V_{\ell,m} := \max\left\{\max_{p\in[m]}\kappa_{j_p}, \beta_m\tau_{j_m}\right\} = \max\left\{\max_{i\in S_{\ell,m}}\kappa_i, \beta_m \max_{i\in S_{\ell,m}}\tau_i\right\}.$$

Therefore $V_{\ell,m}$ is exactly the objective value on the set $S_{\ell,m}$. Hence $\min_{m\in[n]}\widehat{T}_{\text{det}}(m) \leq \text{OPT}$.

It remains to prove the opposite inequality. Fix any $m \in [n]$ and any set $S \subseteq [n]$ with $|S| = m$. Let $K_\ell = \max_{i\in S}\kappa_i$. for some ℓ. Since $\kappa_i \leq K_\ell$ for all $i \in S$, we have $S \subseteq P_\ell$. Therefore $|P_\ell| \geq m$. Since $j_1, \ldots, j_m$ are the m workers in P_ℓ with the smallest values of τ_i, we have

$$\tau_{j_m} \leq \max_{i\in S}\tau_i.$$

Moreover, because $S_{\ell,m} \subseteq P_\ell$, we have

$$\max_{p\in[m]}\kappa_{j_p} \leq K_\ell = \max_{i\in S}\kappa_i.$$

Thus,

$$V_{\ell,m} = \max\left\{\max_{p\in[m]}\kappa_{j_p}, \beta_m\tau_{j_m}\right\} \leq \max\left\{\max_{i\in S}\kappa_i, \beta_m \max_{i\in S}\tau_i\right\}.$$

Since Algorithm 5 minimizes over all such pairs (ℓ, m), we obtain

$$\text{OPT} \leq \max\left\{\max_{i\in S}\kappa_i, \beta_m \max_{i\in S}\tau_i\right\}.$$

Taking the minimum over all $S \subseteq [n]$ with $|S| = m$, and then over all $m \in [n]$, gives $\text{OPT} \leq \min_{m\in[n]}\widehat{T}_{\text{det}}(m)$. Combining the two inequalities, we conclude that $\text{OPT} = \min_{m\in[n]}\widehat{T}_{\text{det}}(m)$. It left to note that $T_{\text{det}}(m) = \frac{L\Delta}{\varepsilon}\widehat{T}_{\text{det}}(m)$ for all $m \in [n]$.

Now we prove the second part of the theorem. Recall that

$$T_{\text{optimal}} = \frac{L\Delta}{\varepsilon}\min_{m\in[n]}\max\left\{\max\{\kappa_{\pi_m}, \tau_{\pi_m}\}, \frac{\sigma^2}{\varepsilon}\left(\sum_{i=1}^{m}\frac{1}{\tau_{\pi_i}}\right)^{-1}\right\},$$

up to constant factors, where π sorts $\{\max\{\kappa_i, \tau_i\}\}_{i\in[n]}$. Let $\bar{m}$ be an optimal choice and define

$$a_{\bar{m}} := \max\{\kappa_{\pi_{\bar{m}}}, \tau_{\pi_{\bar{m}}}\}, A_{\bar{m}} := \sum_{i=1}^{\bar{m}}\frac{1}{\tau_{\pi_i}}, \bar{S} := \{\pi_1, \ldots, \pi_{\bar{m}}\}.$$

Then

$$T_{\text{optimal}} = \frac{L\Delta}{\varepsilon}\max\left\{a_{\bar{m}}, \frac{\sigma^2}{\varepsilon}A_{\bar{m}}^{-1}\right\}.$$

Now reorder the elements of $\bar{S}$ so that $\tau_{q_1} \leq \tau_{q_2} \leq \cdots \leq \tau_{q_{\bar{m}}}$, $\{q_1, \ldots, q_{\bar{m}}\} = \bar{S}$. For each $j \in [\bar{m}]$, define $S_j := \{q_1, \ldots, q_j\}$. Then

$$\max_{i \in S_j} \tau_i = \tau_{q_j}, \qquad \max_{i \in S_j} \kappa_i \leq a_{\bar{m}}.$$

Choose $j^* \in \arg\min_{j \in [\bar{m}]} \frac{\tau_{q_j}}{j}$. Note that

$$\frac{\tau_{q_{j^*}}}{j^*} \leq \frac{H_{\bar{m}}}{A_{\bar{m}}}, \qquad H_{\bar{m}} := \sum_{j=1}^{\bar{m}} \frac{1}{j}.$$

Indeed,

$$A_{\bar{m}} = \sum_{j=1}^{\bar{m}} \frac{1}{\tau_{q_j}} = \sum_{j=1}^{\bar{m}} \frac{j/\tau_{q_j}}{j} \leq \left(\max_{j \in [\bar{m}]} \frac{j}{\tau_{q_j}} \right) \sum_{j=1}^{\bar{m}} \frac{1}{j} = \frac{H_{\bar{m}}}{\min_{j \in [\bar{m}]} \tau_{q_j}/j}.$$

Using the feasible choice S_{j^*} in the definition of T_{det}, we obtain

$$\begin{aligned}
\min_{m \in [n]} T_{\text{det}}(m) &\leq T_{\text{det}}(j^*) \\
&\leq \frac{L\Delta}{\varepsilon} \max\left\{ \max_{i \in S_{j^*}} \kappa_i, \max_{i \in S_{j^*}} \tau_i \max\left\{1, \frac{\sigma^2}{j^*\varepsilon}\right\} \right\} \\
&= \frac{L\Delta}{\varepsilon} \max\left\{ \max_{i \in S_{j^*}} \kappa_i, \tau_{q_{j^*}}, \frac{\sigma^2 \tau_{q_{j^*}}}{j^*\varepsilon} \right\}.
\end{aligned}$$

Since $S_{j^*} \subseteq \bar{S}$, we have

$$\max_{i \in S_{j^*}} \kappa_i \leq a_{\bar{m}}, \qquad \tau_{q_{j^*}} \leq a_{\bar{m}}.$$

Therefore,

$$\min_{m \in [n]} T_{\text{det}}(m) \leq \frac{L\Delta}{\varepsilon} \max\left\{ a_{\bar{m}}, \frac{\sigma^2 \tau_{q_{j^*}}}{j^*\varepsilon} \right\} \leq \frac{L\Delta}{\varepsilon} \max\left\{ a_{\bar{m}}, \frac{\sigma^2}{\varepsilon} \frac{H_{\bar{m}}}{A_{\bar{m}}} \right\} \leq H_{\bar{m}} \frac{L\Delta}{\varepsilon} \max\left\{ a_{\bar{m}}, \frac{\sigma^2}{\varepsilon} A_{\bar{m}}^{-1} \right\}.$$

By optimality of $\bar{m}$, this yields $\min_{m \in [n]} T_{\text{det}}(m) \leq H_{\bar{m}}\, T_{\text{optimal}}$. Finally, since $H_{\bar{m}} \leq 2\log(n+1)$, we get

$$\min_{m \in [n]} T_{\text{det}}(m) \leq 2\log(n+1) T_{\text{optimal}}.$$

□

H Proof of Theorem 3.3

Theorem 3.3 (Lower Bound under Random Computation Model)**.** *For all $\Delta, L, \varepsilon, R, \sigma > 0$, and time delays satisfying Assumption 3.1 such that*[7] $\tau_1 \max\left\{1, \sigma^2/n\varepsilon\right\} \geq \bar{c} R \max\left\{1, \sigma/\sqrt{\varepsilon}\right\} \log(n)$*, where $\bar{c}$ is a universal constant, there exists a smooth function f with L–Lipschitz gradients, and $f(0) - f^* \leq \Delta$, and an oracle that returns unbiased stochastic gradients with σ^2–bounded gradient variance such that the expected time of any zero-respecting (see Definition H.1) algorithm that does not stop computations is $\Omega\left(T_{\text{optimal}}\right)$ seconds (from (2)) to get an ε-stationary point.*

Proof. We are slightly concise in our descriptions, since the proof largely repeats the ideas from (Arjevani et al., 2020; Carmon et al., 2021; Arjevani et al., 2022; Tyurin and Richtárik, 2023, 2024; Tyurin et al., 2024). We analyze zero-respecting algorithms:

Definition H.1 (Zero-respecting algorithm)**.** At any moment of time or iteration, given the previously computed stochastic gradients $\{\nabla f(\bar{x}^k)\}_{0 \leq k \leq \bar{k}}$, the algorithm can only produce a query point $x \in \mathbb{R}^d$ such that

$$\text{supp}(x) \subseteq \bigcup_{k=0}^{\bar{k}} \text{supp}\left(\nabla f(\bar{x}^k)\right),$$

where $\text{supp}(v) := \{i \in [d] \mid v_i \neq 0\}$ denotes the support of a vector v.

[7] The inequality is satisfied if, for instance, ε is small enough, R is small enough, or σ is large enough.

Basically, the proofs reduce to the analysis of the concentration of random variables. In particular, Tyurin and Richtárik (2023) showed that the time required to find an ε–stationary point is lower bounded by

$$\sum_{k=1}^{T} \min_{i\in[n]} \tau_i \eta_{ki} \tag{23}$$

if the computation times satisfy Assumption 2.2, where $T = \Theta\left(\frac{L\Delta}{\varepsilon}\right)$ and η_{ki}, conditioned on $\{\eta_{k'i}\}_{i\in[n],k'<k}$, satisfies the geometric distribution with parameter $p = \Theta\left(\min\{\frac{\varepsilon}{\sigma^2}, 1\}\right)$ for all $k \geq 1$ and $i \in [n]$.

Then,

$$\mathbb{E}\left[\sum_{k=1}^{T} \min_{i\in[n]} \tau_i \eta_{ki}\right] \geq \Omega\left(\min_{m\in[n]}\left[\left(\frac{1}{m}\sum_{i=1}^{m}\frac{1}{\tau_i}\right)^{-1}\left(\max\left\{\frac{L\Delta}{\varepsilon}, \frac{\sigma^2 L\Delta}{m\varepsilon^2}\right\}\right)\right]\right), \tag{24}$$

which follows from Theorem 6.4 (Tyurin and Richtárik, 2023) since $\mathbb{E}[\mu] \geq t \times \mathbb{P}(\mu \geq t)$ for all non-negative μ random variables.

However, under Assumption 3.1, we have to analyze

$$T_{\text{lower}} := \sum_{k=1}^{T} \min_{i\in[n]} \sum_{j=1}^{\eta_{ki}} \bar{\tau}_{kij}, \tag{25}$$

where $\{\bar{\tau}_{kij}\}$ are i.i.d. random variables that satisfy Assumption 3.1. The lower bound (25) reduces to (23), when $\bar{\tau}_{kij} = \tau_i$ almost surely. Similarly to (Tyurin and Richtárik, 2023), the lower bound follows from the fact that workers have to "uncover" T coordinates of the "worst-case" function (Carmon et al., 2020) sequentially, and the time to discover one coordinate is lower bounded by $\min_{i\in[n]} \sum_{j=1}^{\eta_{ki}} \bar{\tau}_{kij}$ since each worker calculate stochastic gradients in parallel and wait for the moment when one of them draw a "lucky" stochastic gradient. See details in (Arjevani et al., 2022; Tyurin and Richtárik, 2023).

It is left to bound $\mathbb{E}[T_{\text{lower}}]$. Notice that

$$\mathbb{E}[T_{\text{lower}}] \geq \mathbb{E}\left[\sum_{k=1}^{T} \min_{i\in[n]} \tau_i \eta_{ki}\right] + \mathbb{E}\left[\sum_{k=1}^{T} \min_{i\in[n]} \sum_{j=1}^{\eta_{ki}} (\bar{\tau}_{kij} - \tau_i)\right] = \mathbb{E}\left[\sum_{k=1}^{T} \min_{i\in[n]} \tau_i \eta_{ki}\right] - \mathbb{E}\left[\sum_{k=1}^{T} \max_{i\in[n]} \sum_{j=1}^{\eta_{ki}} (\tau_i - \bar{\tau}_{kij})\right]. \tag{26}$$

Using (24), we can bound the first term. Consider one of the elements from the second term. Using simple algebra,

$$\begin{aligned}
&\mathbb{E}\left[\max_{i\in[n]} \sum_{j=1}^{\eta_{ki}} (\tau_i - \bar{\tau}_{kij})\right] = \frac{1}{\lambda}\mathbb{E}\left[\lambda \max_{i\in[n]} \sum_{j=1}^{\eta_{ki}} (\tau_i - \bar{\tau}_{kij})\right] \\
&= \frac{1}{\lambda}\log\exp\left(\mathbb{E}\left[\lambda \max_{i\in[n]} \sum_{j=1}^{\eta_{ki}} (\tau_i - \bar{\tau}_{kij})\right]\right) \leq \frac{1}{\lambda}\log\mathbb{E}\left[\exp\left(\lambda \max_{i\in[n]} \sum_{j=1}^{\eta_{ki}} (\tau_i - \bar{\tau}_{kij})\right)\right],
\end{aligned}$$

for all $\lambda > 0$, where we use Jensen's inequality. Next,

$$\mathbb{E}\left[\max_{i\in[n]} \sum_{j=1}^{\eta_{ki}} (\tau_i - \bar{\tau}_{kij})\right] \leq \frac{1}{\lambda}\log\left(\mathbb{E}\left[\sum_{i=1}^{n}\exp\left(\lambda \sum_{j=1}^{\eta_{ki}} (\tau_i - \bar{\tau}_{kij})\right)\right]\right) = \frac{1}{\lambda}\log\left(\sum_{i=1}^{n}\mathbb{E}\left[\exp\left(\lambda \sum_{j=1}^{\eta_{ki}} (\tau_i - \bar{\tau}_{kij})\right)\right]\right). \tag{27}$$

Next,

$$\mathbb{E}\left[\exp\left(\lambda \sum_{j=1}^{\eta_{ki}} (\tau_i - \bar{\tau}_{kij})\right)\right] = \mathbb{E}\left[\mathbb{E}\left[\prod_{j=1}^{\eta_{ki}} \exp\left(\lambda(\tau_i - \bar{\tau}_{kij})\right) \middle| \eta_{ki}\right]\right].$$

By construction, η_{ki} is the smallest index in a sequence of random Bernoulli variables that is constructed independently of $\bar{\tau}_{kij}$. Moreover, $\bar{\tau}_{kij}$ are i.i.d. due to Assumption 3.1. Thus,

$$\mathbb{E}\left[\exp\left(\lambda\sum_{j=1}^{\eta_{ki}}(\tau_i-\bar{\tau}_{kij})\right)\right]=\mathbb{E}\left[\prod_{j=1}^{\eta_{ki}}\mathbb{E}\left[\exp\left(\lambda(\tau_i-\bar{\tau}_{kij})\right)\right]\right].$$

Since $\bar{\tau}_{kij}$ is sub-exponential and $\mathbb{E}\left[\bar{\tau}_{kij}\right]=\tau_i$, there exist a universal constant $c_1\geq 1$ such that

$$\mathbb{E}\left[\exp\left(\lambda\sum_{j=1}^{\eta_{ki}}(\tau_i-\bar{\tau}_{kij})\right)\right]\leq\mathbb{E}\left[\prod_{j=1}^{\eta_{ki}}e^{c_1^2\lambda^2R^2}\right]=\mathbb{E}\left[e^{c_1^2\lambda^2R^2\eta_{ki}}\right]$$

for all $\lambda\leq\frac{1}{c_1R}$ (Vershynin, 2018). Now, since $\eta_{ki}\sim\text{Geom}(p)$, we get

$$\mathbb{E}\left[\exp\left(\lambda\sum_{j=1}^{\eta_{ki}}(\tau_i-\bar{\tau}_{kij})\right)\right]\leq\mathbb{E}\left[e^{c_1^2\lambda^2R^2\eta_{ki}}\right]=\frac{pe^{c_1^2\lambda^2R^2}}{1-(1-p)e^{c_1^2\lambda^2R^2}}$$

for all $\lambda>0$ such that $\lambda\leq\frac{1}{c_1R}$ and $c_1^2\lambda^2R^2<-\log(1-p)$, where we take the moment-generating function (MGF) of η_{ki}, which is only defined for $c_1^2\lambda^2R^2<-\log(1-p)$. Taking $\lambda=\frac{\sqrt{p}}{2c_1R}$, we ensure that $\lambda\leq\frac{1}{c_1R}$ and $c_1^2\lambda^2R^2<-\log(1-p)$. For this choice of λ,

$$\mathbb{E}\left[\exp\left(\lambda\sum_{j=1}^{\eta_{ki}}(\tau_i-\bar{\tau}_{kij})\right)\right]\leq 4$$

because $1-(1-p)e^{c_1^2\lambda^2R^2}=1-(1-p)e^{\frac{p}{4}}\geq 1-(1-p)(1+\frac{p}{2})\geq\frac{p}{2}$ for all $p\in(0,1]$. Substituting this inequality and our choice of λ to (27),

$$\mathbb{E}\left[\max_{i\in[n]}\sum_{j=1}^{\eta_{ki}}(\tau_i-\bar{\tau}_{kij})\right]\leq\frac{2c_1R}{\sqrt{p}}\log(4n).$$

Using (24), (26), $T=\Theta\left(\frac{L\Delta}{\varepsilon}\right)$, $p=\Theta\left(\min\left\{1,\frac{\varepsilon}{\sigma^2}\right\}\right)$, and the last inequality,

$$\mathbb{E}\left[T_{\text{lower}}\right]\geq\Omega\left(\min_{m\in[n]}\left[\left(\frac{1}{m}\sum_{i=1}^{m}\frac{1}{\tau_i}\right)^{-1}\left(\max\left\{\frac{L\Delta}{\varepsilon},\frac{\sigma^2L\Delta}{m\varepsilon^2}\right\}\right)\right]\right)-\Theta\left(\frac{RL\Delta}{\varepsilon}\max\left\{1,\frac{\sigma}{\sqrt{\varepsilon}}\right\}\log(4n)\right).\tag{28}$$

Since

$$\min_{m\in[n]}\left[\left(\frac{1}{m}\sum_{i=1}^{m}\frac{1}{\tau_i}\right)^{-1}\left(\max\left\{\frac{L\Delta}{\varepsilon},\frac{\sigma^2L\Delta}{m\varepsilon^2}\right\}\right)\right]\geq\tau_1\max\left\{\frac{L\Delta}{\varepsilon},\frac{\sigma^2L\Delta}{n\varepsilon^2}\right\}$$

and we assume that

$$\tau_1\max\left\{1,\frac{\sigma^2}{n\varepsilon}\right\}\geq\bar{c}\times R\max\left\{1,\frac{\sigma}{\sqrt{\varepsilon}}\right\}\log(n),$$

where $\bar{c}$ is universal constant, then the term with Ω dominates the term with Θ in (28) and we get

$$\mathbb{E}\left[T_{\text{lower}}\right]\geq\Omega\left(\min_{m\in[n]}\left[\left(\frac{1}{m}\sum_{i=1}^{m}\frac{1}{\tau_i}\right)^{-1}\left(\max\left\{\frac{L\Delta}{\varepsilon},\frac{\sigma^2L\Delta}{m\varepsilon^2}\right\}\right)\right]\right).$$

□

H.1 More theoretical examples

Here we present additional examples for Section 3.

Chi-square distributions. Suppose that $\bar{\tau}_i \sim \chi^2_{k_i}$ (chi-square distributions) with $k_i \in \mathbb{N}$ for all $i \in [n]$. Then $\mathbb{E}[\bar{\tau}_i] = \tau_i = k_i$, and the chi-square distribution is (τ_i, R_i)–sub-exponential with $R_i = \mathcal{O}(\sqrt{k_i})$. As long as $\min_{i\in[n]} k_i \geq \max_{i\in[n]} \sqrt{k_i}$, we have $R \equiv \max_{i\in[n]} R_i = \mathcal{O}(\tau_i)$ for all $i \in [n]$. Then, due to Corollary 3.4, m-Synchronous SGD is nearly optimal if, for instance, ε is small.

Gamma distribution. Suppose that $\bar{\tau}_i \sim \text{Gamma}(a_i, b_i)$ with $a_i, b_i > 0$. In this case, $\mathbb{E}[\bar{\tau}_i] = \tau_i = a_i b_i$, and $\bar{\tau}_i$ is (τ_i, R_i)–sub-exponential random variable with $R_i = \mathcal{O}(\max\{\sqrt{a_i}, 1\}b_i)$ (Boucheron et al., 2003). Then, due to Corollary 3.4, m-Synchronous SGD is nearly optimal when $\min_{i\in[n]} a_i b_i \geq \max_{i\in[n]} \max\{\sqrt{a_i}, 1\}b_i$ and ε is small, where the former true if, for instance, $\{a_i\}$ are large.

Shifted exponential distribution. Assume that $\bar{\eta}_i$ is from the exponential distribution $\text{Exp}(\lambda_i)$ for all $i \in [n]$. We define the shifted exponential random variable as $\bar{\tau}_i = \mu_i + \bar{\eta}_i$ for some $\mu_i \geq 0$. In this case, we can take $R = \Theta(\max_{i\in[n]}(1/\lambda_i))$ and $\tau_i = \mu_i + 1/\lambda_i$ for all $i \in [n]$. Thus, as long as $\max_{i\in[n]}(1/\lambda_i) \leq \min_{i\in[n]}(\mu_i + 1/\lambda_i)$, m-Synchronous SGD is nearly optimal when, for instance, ε is small.

I Proof of Proposition 4.1

Proof. Consider

$$g(m) := \tau_m \max\left\{1, \frac{\sigma^2}{m\varepsilon}\right\}$$

for all $m \in [n]$. For all $m \geq \left\lceil \frac{\sigma^2}{\varepsilon} \right\rceil$, $g(m) = \tau_m$, which is non-decreasing. Thus, it is sufficient to restrict the function to $m \leq \min\left\{\left\lceil \sigma^2/\varepsilon \right\rceil, n\right\}$ to find its minimizer.

For all $m \leq \min\left\{\left\lceil \sigma^2/\varepsilon \right\rceil, n\right\}$, if $\frac{\sigma^2}{\varepsilon} \leq 1$, then $m = 1$. Let $\frac{\sigma^2}{\varepsilon} > 1$. If $m \leq \min\left\{\left\lceil \sigma^2/\varepsilon \right\rceil, n\right\}$, then clearly $m \leq \left\lceil \sigma^2/\varepsilon \right\rceil$. Notice that

$$\tau_m \frac{\sigma^2}{m\varepsilon} \leq \tau_m \max\left\{1, \frac{\sigma^2}{m\varepsilon}\right\} \leq 2\tau_m \frac{\sigma^2}{m\varepsilon}$$

since $m \leq \left\lceil \sigma^2/\varepsilon \right\rceil \leq 2\sigma^2/\varepsilon$. □

J Proof of Theorems 5.3

Theorem 5.3 (Time Complexity under Universal Computation Model)**.** *Under the setting of Theorem 2.1, additionally assume that Assumption 5.1 holds. Then, for all $m \in [n]$, the time complexity of m-Synchronous SGD (Algorithm 3) is $\bar{t}_{\bar{K}}$ seconds, where the sequence $\{\bar{t}_k\}$ is defined as*

$$\bar{t}_{k+1} = \min\left\{t \geq 0 : \max_{S\subseteq[n],|S|=m} \min_{i\in S} N_i(\bar{t}_k, t) = 2\right\}, \tag{12}$$

$\bar{t}_0 = 0$, *and* $\bar{K} := \left\lceil 16 \max\left\{L\Delta/\varepsilon, \sigma^2 L\Delta/m\varepsilon^2\right\}\right\rceil$.

Proof. Algorithm 3 starts at time $\bar{t}_0 = t_0 = 0$. Let us define t_k as the first time moment when Algorithm 3 finishes the k_{th} iteration. Then, if

$$\bar{t}_1 = \min\left\{t \geq 0 : \max_{S\subseteq[n],|S|=m} \min_{i\in S} N_i(\bar{t}_0, t) = 2\right\},$$

then $\bar{t}_1 \geq t_1$ because it is sufficient for the algorithm to wait for the fastest m workers. Using mathematical induction, assume that $\bar{t}_k \geq t_k$. The $k+1^{\text{th}}$ iteration starts at time t_k, and the algorithm waits for the first m workers to finish their calculations. In the worst case, worker i has to finish calculating the previous stochastic gradient (requested before time t_k and not used), and start calculating the new stochastic gradient after time t_k. Thus, worker i will finish calculating the first stochastic gradient started after t_k after $\hat{t}$ seconds such that $N_i(\bar{t}_k, \hat{t}) \geq 2$, and therefore

$$\bar{t}_{k+1} = \min\left\{t \geq 0 : \max_{S\subseteq[n],|S|=m} \min_{i\in S} N_i(\bar{t}_k, t) = 2\right\}$$

is an upper bound for t_{k+1}. □

K Proof of Theorem 5.5

Theorem 5.5 (Time Complexity under Partial Participation)**.** *Under the setting of Theorem 2.1, additionally assume that Assumption 5.4 holds. Then, the time complexity of Algorithm 3 is optimal for all $p < 0.4$. For all $p < 0.4$ and any $m \in \{\lfloor n/5 \rfloor, \ldots, \lfloor (1-2p)n \rfloor\}$, the time complexity is*

$$\mathcal{O}\left(\frac{1}{v}\max\left\{\frac{L\Delta}{\varepsilon}, \frac{\sigma^2 L\Delta}{n\varepsilon^2}\right\}\right) \tag{14}$$

Proof. In Assumption 5.4, we assume that, for all $t \geq 0$, there exists a subset S_t of at least size $(1-p)n$ such that $v_i(t) = v \in \mathbb{R}_+$ for all $i \in S_t$, $v_i(t) \leq v$ for all $i \notin S_t$. Using Theorem 5.3, we now derive an explicit formula for the time complexity of m-Synchronous SGD (Algorithm 3).

Notice that

$$\sum_{i=1}^{n}\int_{\bar{t}_k}^{t} v_i(t)dt = \int_{\bar{t}_k}^{t}\sum_{i=1}^{n} v_i(t)dt \geq (1-p)nv(t-t_k)$$

for all $t \geq \bar{t}_k$. At the same time,

$$\int_{\bar{t}_k}^{t} v_i(t)dt \leq (t-\bar{t}_k)v$$

for all $t \geq \bar{t}_k$ and $i \in [n]$. For any fixed $t > \bar{t}_k$, let $\bar{n}$ be the number of workers such that

$$\int_{\bar{t}_k}^{t} v_i(t)dt < \frac{1}{2} \times (t-\bar{t}_k)v.$$

Then

$$(n-\bar{n})v(t-\bar{t}_k) + \frac{1}{2}\bar{n}v(t-\bar{t}_k) \geq (1-p)nv(t-\bar{t}_k) \Leftrightarrow \bar{n} \leq 2pn.$$

Therefore, for all $t > \bar{t}_k$, at least $n - 2pn = (1-2p)n$ workers satisfy the inequality

$$\int_{\bar{t}_k}^{t} v_i(t)dt \geq \frac{1}{2}(t-\bar{t}_k)v.$$

Thus, for all $t > \bar{t}_k$, $p < 0.4$, and $m \in \{\lfloor n/5 \rfloor, \ldots, \lfloor (1-2p)n \rfloor\}$, there exists a set S of size m such that

$$\min_{i \in S} N_i(\bar{t}_k, t) \geq \frac{1}{2}(t-\bar{t}_k)v,$$

meaning that $\bar{t}_{k+1} \leq \bar{t}_k + \frac{4}{v}$. Finally,

$$\bar{t}_{\bar{K}} = \mathcal{O}\left(\frac{1}{v}\max\left\{\frac{L\Delta}{\varepsilon}, \frac{\sigma^2 L\Delta}{m\varepsilon^2}\right\}\right).$$

It is left to notice that $m = \Theta(n)$ for all $m \in \{\lfloor n/5 \rfloor, \ldots, \lfloor (1-2p)n \rfloor\}$ and $p < 0.4$. □

L Proof of Proposition 4.2

Proof. We consider $\tau_m = \tau_1 m^\alpha + \delta_m$ under the conditions $0 \leq \alpha \leq 1$ and $0 \leq \delta_m \leq \delta$. In this case, we have to find m that minimizes

$$h(m) := \frac{\tau_m}{m}$$

for all $m \leq \min\left\{\left\lceil \sigma^2/\varepsilon \right\rceil, n\right\}$ due to Proposition 4.1. We have

$$h(m) = \frac{\tau_1 m^\alpha + \delta_m}{m} = \Theta\left(\frac{\tau_1 m^\alpha}{m}\right)$$

for all $m \geq \left(\frac{\delta}{\tau_1}\right)^{1/\alpha}$. Thus, $m = \min\left\{\left\lceil \sigma^2/\varepsilon \right\rceil, n\right\}$ is a minimizer on the interval $\left[\left(\frac{\delta}{\tau_1}\right)^{1/\alpha}, \min\left\{\left\lceil \sigma^2/\varepsilon \right\rceil, n\right\}\right]$ and

$$h(m) = \Theta\left(\frac{\tau_1}{m^{1-\alpha}}\right)$$

for $m = \min\left\{\left\lceil \sigma^2/\varepsilon \right\rceil, n\right\}$.

For all $m \leq \left(\frac{\delta}{\tau_1}\right)^{1/\alpha}$,

$$h(m) \geq \frac{\tau_1}{m^{1-\alpha}} \geq \frac{\tau_1^{1/\alpha}}{\delta^{(1-\alpha)/\alpha}}.$$

Combining both cases, we can conclude that $m = \min\left\{\left\lceil \sigma^2/\varepsilon \right\rceil, n\right\}$ minimizes h when $m \geq \left(\frac{\delta}{\tau_1}\right)^{1/\alpha}$ (either n or σ^2/ε is large). □

M A Theoretical Example Where Asynchronicity Is Needed

In this section, we formalize the example from Section 6. In order to formalize it, we can define the computation powers as

$$v_i(t) = v$$

for all $i > 1$ and $t \geq 0$, and

$$v_1(t) = \begin{cases} v, & t \leq \bar{t} := \frac{1}{v}, \\ \infty, & t > \bar{t} \end{cases}$$

for the first worker. All workers except the first one have constant computational power. The first worker has the constant computation power for the first $\bar{t}$ seconds, and then becomes infinitely fast.

In this example, knowing only the behavior of the first worker for at most the first $\bar{t}$ seconds, the best choice of m in Algorithm 3 is $m = n$, and the time complexity is

$$\Theta\left(\frac{1}{v}\max\left\{\frac{L\Delta}{\varepsilon}, \frac{\sigma^2 L\Delta}{n\varepsilon^2}\right\}\right) \tag{29}$$

because the algorithm always waits for all the workers, while the first one quickly computes one stochastic gradient and then is idle during each iteration.

At the same time, according to Theorem 5.2, the lower bound and the time complexity of the optimal asynchronous methods can be much faster. Indeed, notice that

$$\begin{aligned} \underline{t}_1 &= \min\left\{t \,:\, \sum_{i=1}^{n}\left\lfloor \int_0^t v_i(\tau)d\tau \right\rfloor \geq c_2\left\lceil\frac{\sigma^2}{\varepsilon}\right\rceil\right\} \\ &= \min\left\{t \,:\, (n-1)\lfloor tv\rfloor + \mathbf{1}[t \leq \bar{t}] \times \lfloor tv\rfloor + \mathbf{1}[t > \bar{t}] \times \infty \geq c_2\left\lceil\frac{\sigma^2}{\varepsilon}\right\rceil\right\}, \end{aligned}$$

where we assume that $0 \times \infty = 0$. Therefore, $\underline{t}_1 \leq 2\bar{t}$, and

$$\underline{t}_2 = \min\left\{t \,:\, \sum_{i=1}^{n} N_i(\underline{t}_1, t) \geq c_2\left\lceil\frac{\sigma^2}{\varepsilon}\right\rceil\right\} \leq \min\left\{t \,:\, \sum_{i=1}^{n} N_i(2\bar{t}, t) \geq c_2\left\lceil\frac{\sigma^2}{\varepsilon}\right\rceil\right\} \leq 2\bar{t} + \delta$$

for all $\delta > 0$ because $\sum\limits_{i=1}^{n} N_i(2\bar{t}, 2\bar{t}+\delta) \geq N_1(2\bar{t}, 2\bar{t}+\delta) = \infty$. Using the same reasoning, we can conclude that

$$\underline{t}_K \leq \Theta\left(\bar{t} + \delta\frac{L\Delta}{\varepsilon}\right).$$

Taking $\delta = \frac{\varepsilon\bar{t}}{L\Delta}$,

$$\underline{t}_K \leq \Theta\left(\bar{t}\right) = \Theta\left(\frac{1}{v}\right),$$

which can arbitrarily smaller than (29).

N NanoGPT Benchmark

In this section, we numerically evaluate the computation times for the NanoGPT model to understand the difference between the noise factor R and the mean computation time τ_1 in Assumption 3.1. We use the NanoGPT model (Karpathy, 2022) with `vocab_size=50304`, `block_size=512`, `n_layer=6`, `n_head=6`, and `n_embd=384`. We use `Tesla V100-SXM3-32GB`. We first run 10 warmup steps and then 200 backpropagation steps to estimate the mean and the variance. Empirically, we observe that the mean is $\approx$ 72.2 ms, while the parameter R is[8] $\approx$ 0.6. Thus, $R \log n \approx 0.6 \log(1,000,000) = 8.2 \ll 71.2$ even if the number of devices is $n = 1,000,000$. In Figure 6, we also present a Q–Q plot, where the computation times are plotted against a normal distribution; we see that the computation times are visually close to normal.

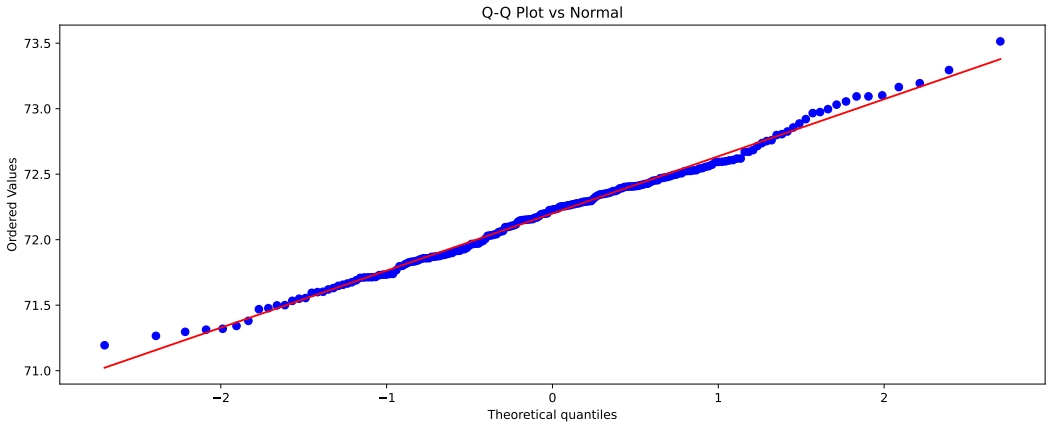


Figure 6: Q-Q Plot of the data from Section N versus a normal distribution.

[8] we numerically found the smallest R such that $\frac{1}{200} \sum_{j=1}^{200} \mathbb{E}\left[\exp(|\bar{\tau}_j - \tilde{\tau}| / R)\right] = 2$, where $\bar{\tau}_j$ are the recorder times and $\tilde{\tau}$ is the estimated mean.

O Extra Experiments and Details

The code was prepared in Python 3 and executed on a machine with 52 CPUs (Intel(R) Xeon(R) Gold 6278C @ 2.60,GHz).

We conduct three types of experiments: (i) we consider a synthetic quadratic optimization task and controlled computation times, where we verify our theoretical results and compare the methods; (ii) then, we verify different computation times on small-scale real machine learning problems; (iii) finally, we implement a custom version in PyTorch and compare Syncrounous SGD and Asyncrounous SGD on the NanoGPT training task.

We first compare the methods on synthetic quadratic optimization problems in which the noise in the computation times of stochastic gradients can be precisely controlled. Consider the problem

$$f(x) = \frac{1}{2}x^\top \mathbf{A} x - b^\top x$$

for all $x \in \mathbb{R}^d$ and take $d = 1000$,

$$\mathbf{A} = \frac{1}{4}\begin{pmatrix} 2 & -1 & & 0 \\ -1 & \ddots & \ddots & \\ & \ddots & \ddots & -1 \\ 0 & & -1 & 2 \end{pmatrix} \in \mathbb{R}^{d\times d} \quad \text{and,} \quad b = \frac{1}{4}\begin{bmatrix} -1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \in \mathbb{R}^d.$$

We define function $\text{prog}(x) := \max\{i \geq 0 \,|\, x_i \neq 0\}$ $\quad (x_0 \equiv 0)$ and consider the following stochastic gradients:

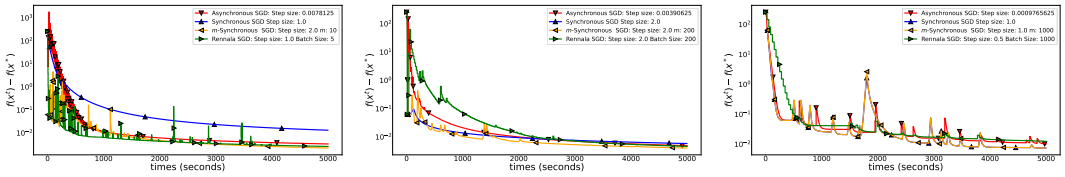
$$[\nabla f(x;\xi)]_j := \nabla_j f(x)\left(1 + \mathbf{1}[j > \text{prog}(x)]\left(\frac{\xi}{p} - 1\right)\right) \quad \forall x \in \mathbb{R}^d, \tag{30}$$

where $\xi \sim \text{Bernoulli}(p)$ for all $i \in [n]$, and $p \in (0, 1]$ is a parameter controlling the noise level: the smaller p the stronger the noise in the stochastic gradients. We use $[x]_j$ to denote the j-th component of a vector $x \in \mathbb{R}^d$. In our experiments, we initialize the optimization at $x^0 = [\sqrt{d}, 0, \ldots, 0]^\top$. In all methods, we performed a grid search over the step size $\gamma \in \left\{2^{-16}, \ldots, 2^4\right\}$. In Rennala SGD, we tune the batch size parameter from the set $\{1, 5, 10, 15, 20, \ldots, n\}$. In m-Synchronous, we also tune the number of active workers m from the set $\{1, 5, 10, 15, 20, \ldots, n\}$.

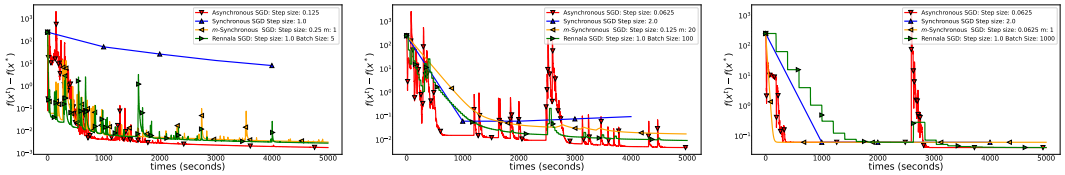
O.1 Quadratic optimization problem with heterogeneous times

In Figure 7, we consider the fixed computation model (Assumption 2.2) with the number of workers $n = 1000$. In each row of Figure 7, we consider one computational scenario, and in each column, we fix the probability from (30). The main observation is that m-Synchronous SGD with a proper m, Asynchronous SGD, and Rennala SGD are competitive, in accordance with Section 2.

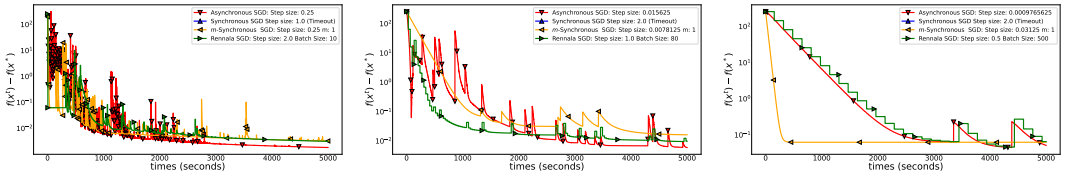
At the same time, Synchronous SGD does not match other methods in some of the plots. Consider the first row, when worker i requires $\tau_i = \sqrt{i}$ seconds. Recall that Proposition 4.2 implies that the optimal number of workers in Algorithm 3 is $m = \min\left\{\left\lceil \sigma^2/\varepsilon \right\rceil, n\right\}$. Consequently, the number of active workers should increase as the gradient noise grows, until the moment when $\sigma^2/\varepsilon \geq n$. In the first row, we see exactly the predicted behavior: when $\sigma^2/\varepsilon \ll n$ (Fig. 7a), Synchronous SGD is slower because the optimal $m \ll n$. However, once we start increasing the variance by taking a smaller p, we can see that the gap between Synchronous SGD and other methods decreases since the optimal choice of m becomes n. In the second row, we see a similar behavior. In the third row, when worker i requires $\tau_i = i^{1.2}$ seconds, the optimal choice of m is 1 due to Proposition 4.1, and Synchronous SGD with all participating workers cannot match other methods.



(a) $p = 0.01, n = 10^3, \tau_i = \sqrt{i}$ (b) $p = 0.001, n = 10^3, \tau_i = \sqrt{i}$ (c) $p = 0.0001, n = 10^3, \tau_i = \sqrt{i}$



(d) $p = 0.01, n = 10^3, \tau_i = i$ (e) $p = 0.001, n = 10^3, \tau_i = i$ (f) $p = 0.0001, n = 10^3, \tau_i = i$

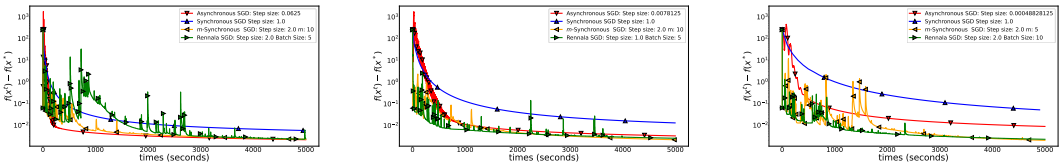


(g) $p = 0.01, n = 10^3, \tau_i = i^{1.2}$ (h) $p = 0.001, n = 10^3, \tau_i = i^{1.2}$ (i) $p = 0.0001, n = 10^3, \tau_i = i^{1.2}$

Figure 7: Comparison of the methods on the quadratic optimization problem under different computation time scenarios and noise levels.

O.2 Quadratic optimization with increasing number of workers

In the previous section, we fix the number of workers to $n = 1000$. In this section, we compare the behavior of methods with $n \in \{100, 1000, 10000\}$. In Figure 8, we observe that m-Synchronous, as expected, is robust to the increasing number of workers, while the gap between Synchronous SGD and other methods increases.



(a) $n = 100$ (b) $n = 1000$ (c) $n = 10000$

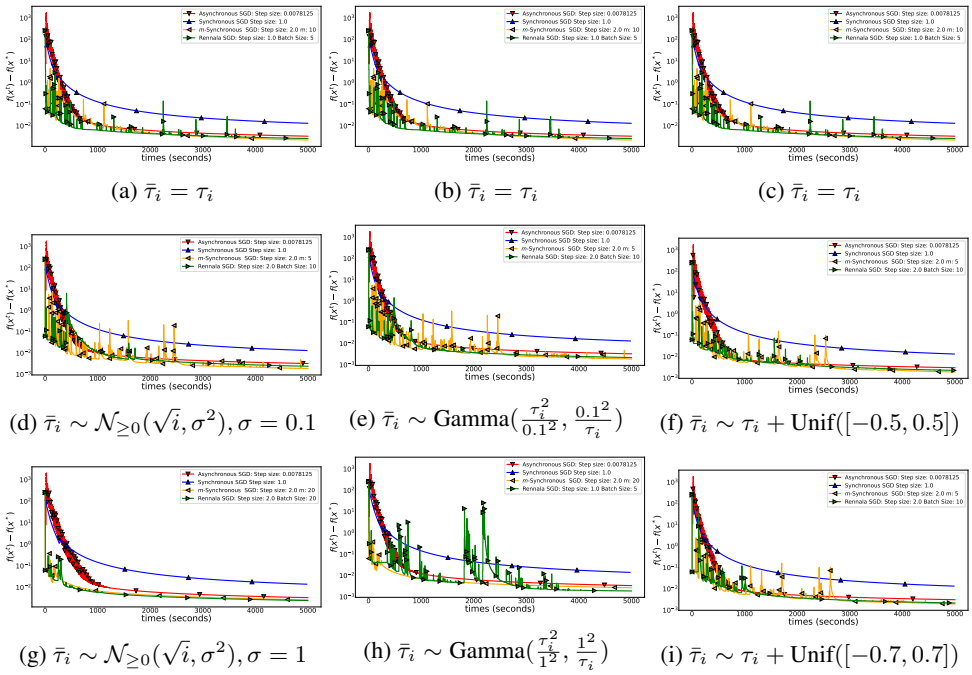
Figure 8: Comparison of the methods on the quadratic optimization problem with different number of workers ($p = 0.01, \tau_i = \sqrt{i}$).

O.3 Quadratic optimization with random computational times

In this section, we consider an experiment setup similar to Section O.1, with the only difference that we examine the performance of optimization methods when computational times are modeled as random variables drawn from different probability distributions.

1. First, we assume that random computational times are drawn from a normal distribution with parameter μ_i and variance σ^2, truncated to the non-negative half-line $[0, \infty)$, which we denote as $\mathcal{N}_{\geq 0}(\mu_i, \sigma^2)$. ($\mathbb{E}[\bar{\tau}_i] \neq \mu_i$, in general).
2. Next, we consider $\bar{\tau}_i$ drawn from a Gamma distribution with shape parameter $k = \tau_i^2/\sigma^2$ and scale parameter $\theta = \sigma^2/\tau_i$. Under this parametrization, the mean and variance are $\mathbb{E}[\bar{\tau}_i] = k\theta = \tau_i$ and $\mathrm{Var}(\bar{\tau}_i) = k\theta^2 = \sigma^2$, accordingly.
3. Finally, we take $\bar{\tau}_i$ drawn from a uniform distribution $\mathrm{Unif}[\tau_i - \sigma, \tau_i + \sigma]$.

We present the results in Figures 9, 10, and 11. In the first row of Figure 9, as expected, we get the same result as in Figure 7a. In all noise regimes, we observe that m-Synchronous SGD is robust and that its performance is comparable to that of the asynchronous methods. We arrive at the same conclusion in Figure 10 when we change τ_i to $\tau_i = i$. Figure 11 also concur. However, in Figure 11, even Synchronous SGD with all participating workers has a good performance when the means are the same, i.e., $\tau_i = 1$, which supports our theoretical results.

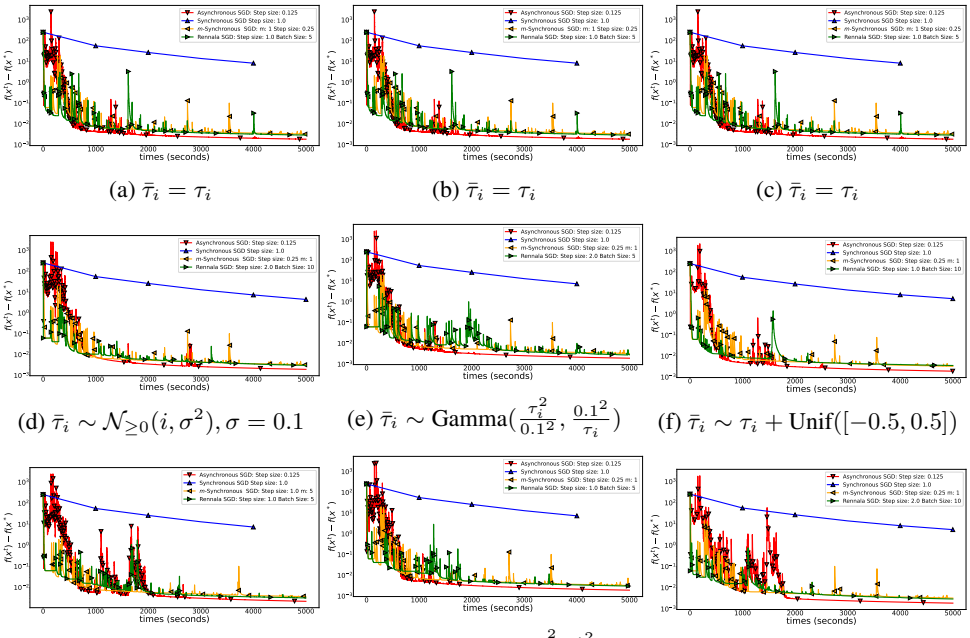


(a) $\bar{\tau}_i = \tau_i$ (b) $\bar{\tau}_i = \tau_i$ (c) $\bar{\tau}_i = \tau_i$

(d) $\bar{\tau}_i \sim \mathcal{N}_{\geq 0}(\sqrt{i}, \sigma^2), \sigma = 0.1$ (e) $\bar{\tau}_i \sim \text{Gamma}(\frac{\tau_i^2}{0.1^2}, \frac{0.1^2}{\tau_i})$ (f) $\bar{\tau}_i \sim \tau_i + \text{Unif}([-0.5, 0.5])$

(g) $\bar{\tau}_i \sim \mathcal{N}_{\geq 0}(\sqrt{i}, \sigma^2), \sigma = 1$ (h) $\bar{\tau}_i \sim \text{Gamma}(\frac{\tau_i^2}{1^2}, \frac{1^2}{\tau_i})$ (i) $\bar{\tau}_i \sim \tau_i + \text{Unif}([-0.7, 0.7])$

Figure 9: Comparison of the methods on a quadratic optimization problem with different types of random computation times. We take $p = 0.01$, $n = 1000$, and $\tau_i = \sqrt{i}$. Three noise regimes are considered: (i) no noise in the first row; (ii) weak noise in the second row; and (iii) stronger noise in the third row. In the first column, we sample computation times from the truncated normal distribution. In the second column, from the gamma distribution. In the third column, from the uniform distribution.

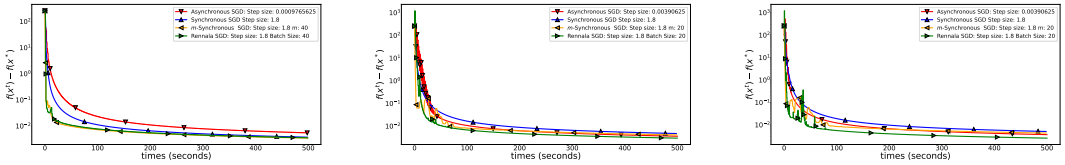


(a) $\bar{\tau}_i = \tau_i$ (b) $\bar{\tau}_i = \tau_i$ (c) $\bar{\tau}_i = \tau_i$

(d) $\bar{\tau}_i \sim \mathcal{N}_{\geq 0}(i, \sigma^2), \sigma = 0.1$ (e) $\bar{\tau}_i \sim \text{Gamma}(\frac{\tau_i^2}{0.1^2}, \frac{0.1^2}{\tau_i})$ (f) $\bar{\tau}_i \sim \tau_i + \text{Unif}([-0.5, 0.5])$

(g) $\bar{\tau}_i \sim \mathcal{N}_{\geq 0}(i, \sigma^2), \sigma = 1$ (h) $\bar{\tau}_i \sim \text{Gamma}(\frac{\tau_i^2}{1^2}, \frac{1^2}{\tau_i})$ (i) $\bar{\tau}_i \sim \tau_i + \text{Unif}([-0.7, 0.7])$

Figure 10: Comparison of the methods on a quadratic optimization problem with different types of random computation times. We take $p = 0.01$, $n = 1000$, and $\tau_i = i$. Three noise regimes are considered: (i) no noise in the first row; (ii) weak noise in the second row; and (iii) stronger noise in the third row. In the first column, we sample computation times from the truncated normal distribution. In the second column, from the gamma distribution. In the third column, from the uniform distribution.



(a) $\bar{\tau}_i = 1$ (b) $\bar{\tau}_i \sim 1 + \text{Unif}([-0.5, 0.5])$ (c) $\bar{\tau}_i \sim 1 + \text{Unif}([-0.7, 0.7])$

Figure 11: Comparison of the methods on the quadratic optimization problem with the same mean $\tau_i = 1$ and uniform noise ($p = 0.01, \tau_i = 1, n = 1000$). In these experiments, for all methods, we choose the best step size from the set $\left\{2^{-16}, \ldots, 2^{-2}, 2^{-1}, 1, 1.2, 1.6, 1.8, 2\right\}$.

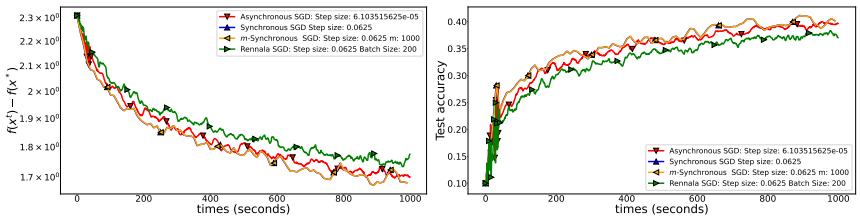
O.4 Small-Scale Machine Learning Task

In this section, we train a two-layer neural network (NN), Linear(input_dim, 32) $\to$ ReLU $\to$ Linear(32, num_classes), with the logistic loss on the *CIFAR-10* dataset (Krizhevsky et al., 2009) and *MNIST* (LeCun et al., 2010) to compare the performance of the methods on tasks where workers compute stochastic gradients via uniformly random data sampling with a sample size of 128.

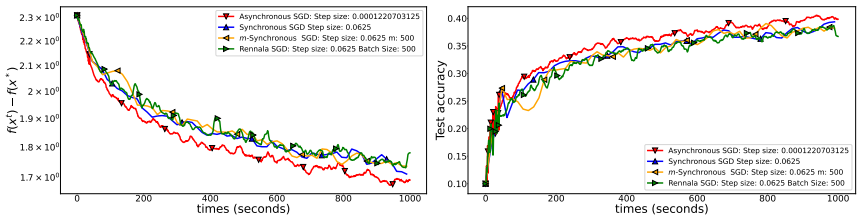
We compare two computational scenarios:
i) $\bar{\tau}_i \sim \text{Unif}(1-\sigma, 1+\sigma)$, the computations times are random variables from the uniform distribution with equal means;
ii) $\bar{\tau}_i \sim \mathcal{N}_{\geq 0}(\mu_i; \sigma^2)$, $\mu_i = \sqrt{i}$, the computations times are random variables from the truncated normal distribution.

The results for CIFAR10 are presented in Figure 12, we consider the first computational scenario $\bar{\tau}_i \sim \text{Unif}(1-\sigma, 1+\sigma)$ and vary σ to check the methods' robustness to these variations. When $\sigma = 0.5$ and $\sigma = 0.7$, all methods are comparable, with Asynchronous SGD being faster by at most $\approx 2\times$ times, which is expected in practical tasks. One important observation is that even Synchronous SGD is as fast as the optimal Rennala SGD method. Notice that a $2\times$ factor can be "caught up" through more efficient implementations and communication (see Section 8, Section O.5, and (Sergeev and Balso, 2018)).

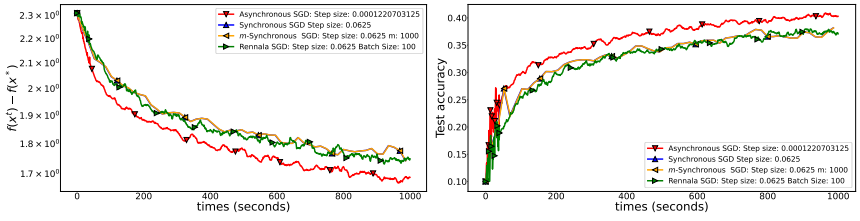
In Figure 13, we consider the second computational scenario $\bar{\tau}_i \sim \mathcal{N}_{\geq 0}(\sqrt{i}; \sigma^2)$ and also vary σ. The conclusions are the same as in the previous computation scenario, with the only main difference that m-Synchronous SGD is more robust than Synchronous SGD when $\tau_i = \sqrt{i}$ are heterogeneous. In Figures 14 and 15, we repeat the same experiments on MNIST, and observe that the behavior and conclusions are consistent across the datasets.



$\bar{\tau}_i = 1$ $\qquad$ $\bar{\tau}_i = 1$

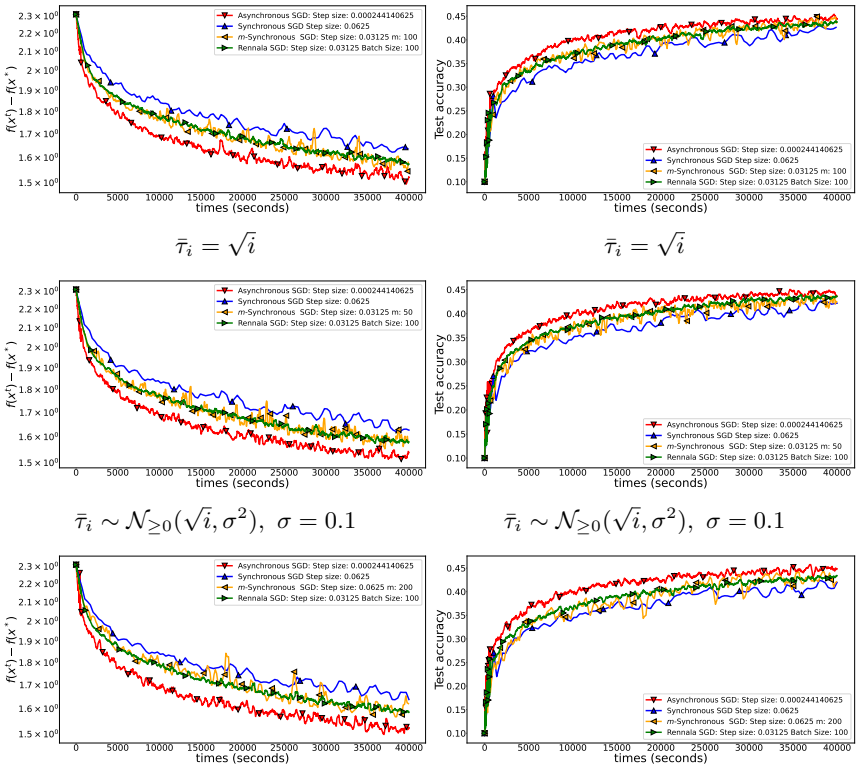


$\bar{\tau}_i \sim 1 + \text{Unif}([-0.5, 0.5])$ $\qquad$ $\bar{\tau}_i \sim 1 + \text{Unif}([-0.5, 0.5])$



$\bar{\tau}_i \sim 1 + \text{Unif}([-0.7, 0.7])$ $\qquad$ $\bar{\tau}_i \sim 1 + \text{Unif}([-0.7, 0.7])$

Figure 12: Experiments with the CIFAR-10 dataset ($n = 1000$). Each row corresponds to a fixed noise level. The first column corresponds to function values, while the second column corresponds to accuracies.

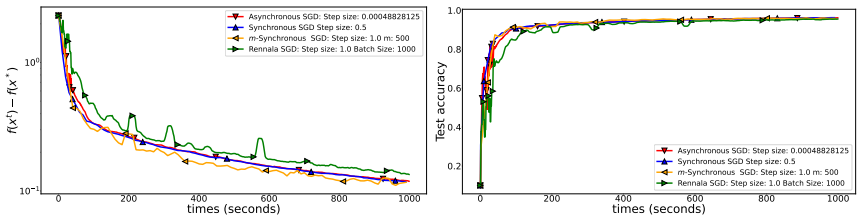


$\bar{\tau}_i = \sqrt{i}$ $\bar{\tau}_i = \sqrt{i}$

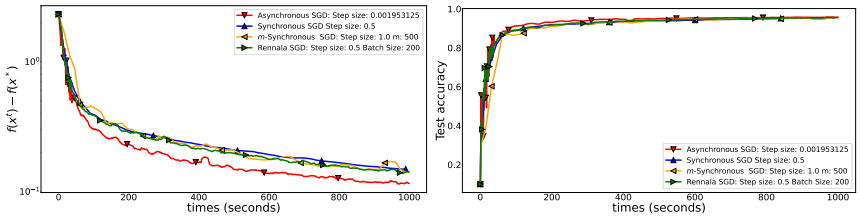
$\bar{\tau}_i \sim \mathcal{N}_{\geq 0}(\sqrt{i}, \sigma^2),\ \sigma = 0.1$ $\bar{\tau}_i \sim \mathcal{N}_{\geq 0}(\sqrt{i}, \sigma^2),\ \sigma = 0.1$

$\bar{\tau}_i \sim \mathcal{N}_{\geq 0}(\sqrt{i}, \sigma^2),\ \sigma = 1$ $\bar{\tau}_i \sim \mathcal{N}_{\geq 0}(\sqrt{i}, \sigma^2),\ \sigma = 1$

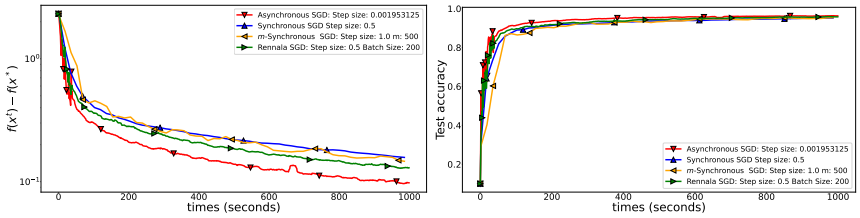
Figure 13: Experiments with the CIFAR-10 dataset ($n = 1000$). Each row corresponds to a fixed noise level. The first column corresponds to function values, while the second column corresponds to accuracies.



$\bar{\tau}_i = 1$ $\bar{\tau}_i = 1$

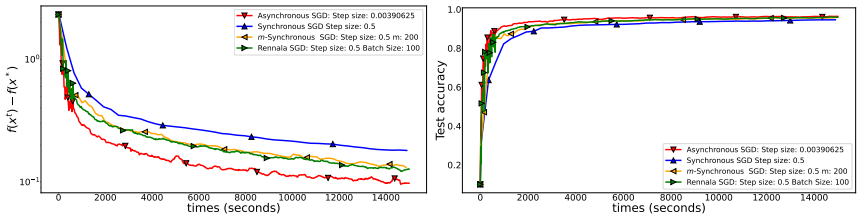


$\bar{\tau}_i \sim 1 + \text{Unif}([-0.5, 0.5])$ $\bar{\tau}_i \sim 1 + \text{Unif}([-0.5, 0.5])$

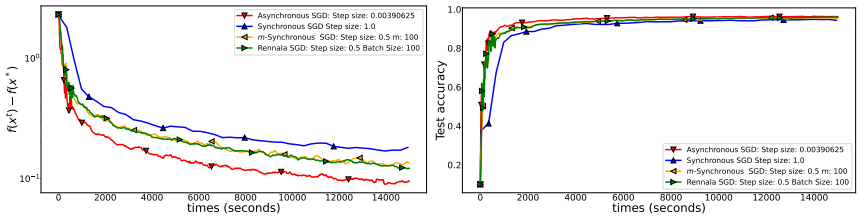


$\bar{\tau}_i \sim 1 + \text{Unif}([-0.7, 0.7])$ $\bar{\tau}_i \sim 1 + \text{Unif}([-0.7, 0.7])$

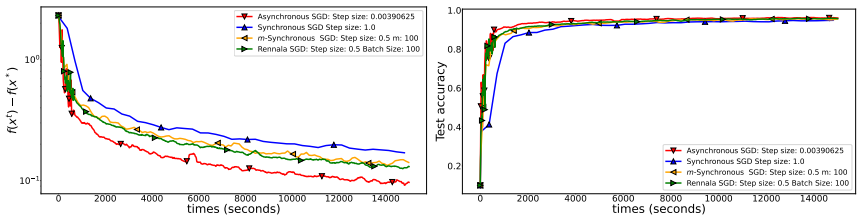
Figure 14: Experiments with the MNIST dataset ($n = 1000$). Each row corresponds to a fixed noise level. The first column corresponds to function values, while the second column corresponds to accuracies.



$\bar{\tau}_i = \sqrt{i}$ $\qquad$ $\bar{\tau}_i = \sqrt{i}$



$\bar{\tau}_i \sim \mathcal{N}_{\geq 0}(\sqrt{i}, 0.1^2)$ $\qquad$ $\bar{\tau}_i \sim \mathcal{N}_{\geq 0}(\sqrt{i}, 0.1^2)$



$\bar{\tau}_i \sim \mathcal{N}_{\geq 0}(\sqrt{i}, 1^2)$ $\qquad$ $\bar{\tau}_i \sim \mathcal{N}_{\geq 0}(\sqrt{i}, 1^2)$

Figure 15: Experiments with the MNIST dataset ($n = 1000$). Each row corresponds to a fixed noise level. The first column corresponds to function values, while the second column corresponds to accuracies.

O.5 NanoGPT experiments

In this section, we compare Synchronous SGD (Algorithm 1) and Asynchronous SGD (Algorithm 2) in the NanoGPT library by Karpathy (2022). We implement custom versions of the algorithms using `torch.distributed` from (Paszke et al., 2019). We use the same parameters as in the `train_shakespeare_char.py` configuration of the library. We run the algorithms with 4 workers operating in parallel. This is the setup with times having homogeneous means; however, the times themselves are random due to naturally unstable computation times. This setup is captured by Assumption 3.1 with equal means. In Figure 16, we present the training and validation losses versus the real measured wall-clock time, and observe that the convergence rates of the algorithms are similar. For a fair comparison, we implemented custom code without the support of code-optimized libraries such as `Horovod` (Sergeev and Balso, 2018) or `torch.nn.parallel.DistributedDataParallel`, which can potentially improve the performance of Synchronous SGD.

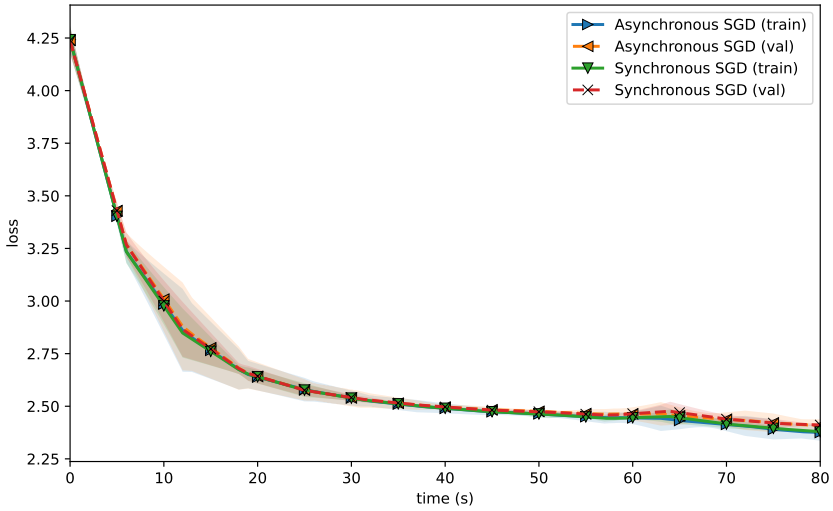


Figure 16: Training and validation losses versus measured wall-clock time for Synchronous SGD (Algorithm 1) and Asynchronous SGD (Algorithm 2) in NanoGPT, using 4 workers. We run the experiments with 3 seeds and plot the error bars.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification:

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Section 6

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

Justification: See the main part and appendix.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: Section O

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: In the supplementary materials.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: Section O

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

   Justification:

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.

- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Section O

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics https://neurips.cc/public/EthicsGuidelines?

   Answer: [Yes]

   Justification:

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: This is a theoretical work with no direct societal impacts, and any indirect effects are beyond the scope of this paper.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification:

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification:

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.

- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A]

    Justification:

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification:

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification:

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [N/A]

Justification:

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.