

# Is Going Asynchronous the Right Way of Handling Device Heterogeneity in Federated Learning?

(Handling Device Heterogeneity in Federated Learning:  
The First Optimal Parallel SGD in the Presence of Data,  
Compute & Communication Heterogeneity)



Peter Richtárik

King Abdullah University of Science and Technology  
Kingdom of Saudi Arabia

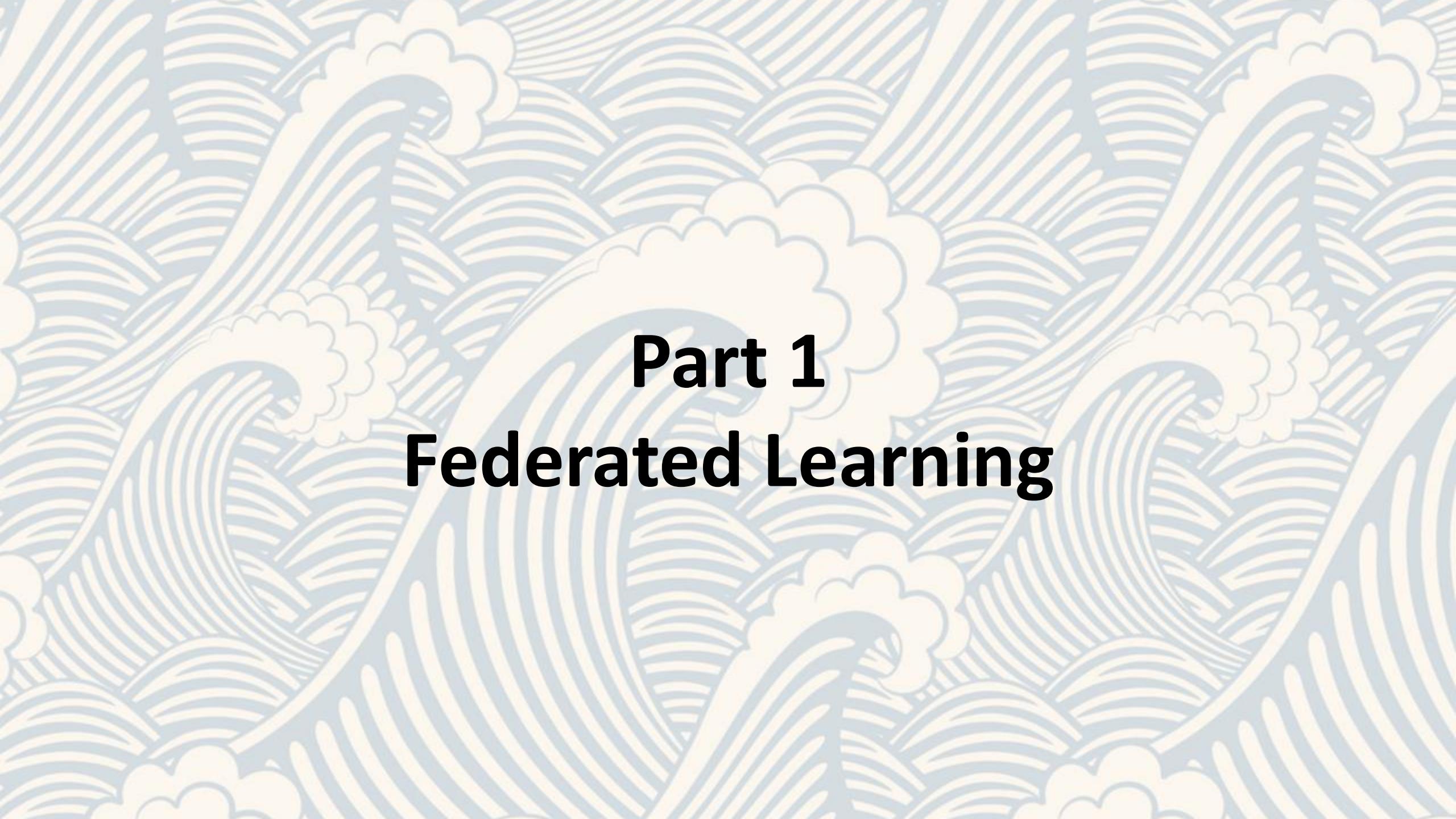


April 9, 2025 @ Federated Learning One World Seminar (FLOW)



# Optimization & Machine Learning Lab @ KAUST





# **Part 1**

# **Federated Learning**



**Jakub Konečný**



**H Brendan McMahan**

**Federated Learning  
was developed in 2015/2016 in a  
collaboration between the University  
of Edinburgh & Google**



THE UNIVERSITY  
of EDINBURGH

H. Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, Blaise Agüera y Arcas

## Communication-Efficient Learning of Deep Networks from Decentralized Data

20th International Conference on Artificial Intelligence and Statistics (AISTATS), 2017

Keith Bonawitz et al

## Practical Secure Aggregation for Federated Learning on User-Held Data

NIPS Private Multi-Party Machine Learning Workshop, 2016

### Google AI Blog

The latest from Google Research

#### Federated Learning: Collaborative Machine Learning without Centralized Training Data

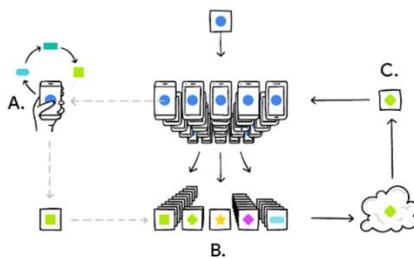
Thursday, April 6, 2017

Posted by Brendan McMahan and Daniel Ramage, Research Scientists

Standard machine learning approaches require centralizing the training data on one machine or a datacenter. And Google has built one of the most secure and robust cloud infrastructures for processing this data to make our services better. Now for models trained from user interaction with mobile devices, we're introducing an additional approach: **Federated Learning**.

Federated Learning enables mobile phones to collaboratively learn a shared prediction model while keeping all the training data on device, decoupling the ability to do machine learning from the need to store the data in the cloud. This goes beyond the use of local models that make predictions on mobile devices (like the [Mobile Vision API](#) and [On-Device Smart Reply](#)) by bringing model *training* to the device as well.

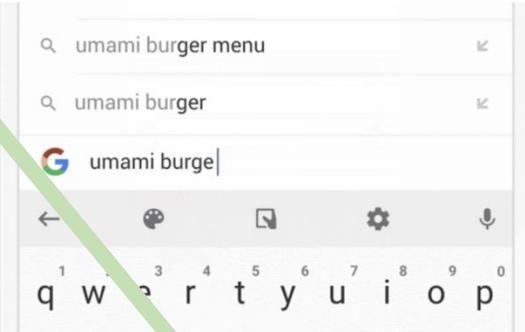
It works like this: your device downloads the current model, improves it by learning from data on your phone, and then summarizes the changes as a small focused update. Only this update to the model is sent to the cloud, using encrypted communication, where it is immediately averaged with other user updates to improve the shared model. All the training data remains on your device, and no individual updates are stored in the cloud.



Your phone personalizes the model locally, based on your usage (A). Many users' updates are aggregated (B) to form a consensus change (C) to the shared model, after which the procedure is repeated.

Federated Learning allows for smarter models, lower latency, and less power consumption, all while ensuring privacy. And this approach has another immediate benefit: in addition to providing an update to the shared model, the improved model on your phone can also be used immediately, powering experiences personalized by the way you use your phone.

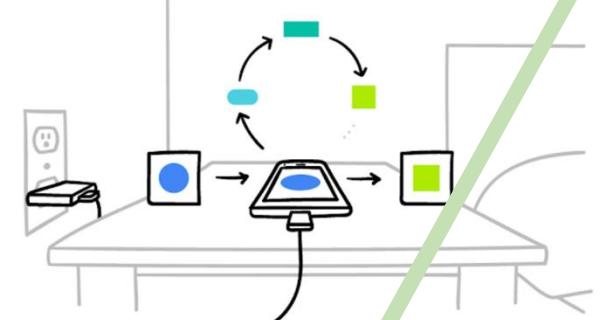
We're currently testing Federated Learning in [Gboard on Android](#), the Google Keyboard. When Gboard shows a suggested query, your phone locally stores information about the current context and whether you clicked the suggestion. Federated Learning processes that history on-device to suggest improvements to the next iteration of Gboard's query suggestion model.



To make Federated Learning possible, we had to overcome many algorithmic and technical challenges. In a typical machine learning system, an optimization algorithm like [Stochastic Gradient Descent \(SGD\)](#) runs on a large dataset partitioned homogeneously across servers in the cloud. Such highly iterative algorithms require low-latency, high-throughput connections to the training data. But in the Federated Learning setting, the data is distributed across millions of devices in a highly uneven fashion. In addition, these devices have significantly higher-latency, lower-throughput connections and are only intermittently available for training.

These bandwidth and latency limitations motivate our [Federated Averaging algorithm](#), which can train deep networks using 10-100x less communication compared to a naively federated version of SGD. The key idea is to use the powerful processors in modern mobile devices to compute higher quality updates than simple gradient steps. Since it takes fewer iterations of high-quality updates to produce a good model, training can use much less communication. As upload speeds are typically [much slower](#) than download speeds, we also developed a novel way to reduce upload communication costs up to another 100x by [compressing updates](#) using random rotations and quantization. While these approaches are focused on training deep networks, we've also [designed algorithms](#) for high-dimensional sparse linear models which excel on problems like click-through-rate prediction.

Deploying this technology to millions of heterogeneous phones running Gboard requires a sophisticated technology stack. On device training uses a miniature version of [TensorFlow](#). Careful scheduling ensures training happens only when the device is idle, plugged in, and on a free wireless connection, so there is no impact on the phone's performance.



The system then needs to communicate and aggregate those model updates in a secure, efficient, scalable, and fault-tolerant way. It's only the combination of research with this infrastructure that makes the benefits of Federated Learning possible.

Federated learning works without the need to store user data in the cloud, but we're not stopping there. We've developed a [Secure Aggregation protocol](#) that uses cryptographic techniques so a coordinating server can only decrypt the average update if 100s or 1000s of users have participated — no individual phone's update can be inspected before averaging. It's the first protocol of its kind that is practical for deep-network-sized problems and real-world connectivity constraints. We designed Federated Averaging so the coordinating server only needs the average update, which allows Secure Aggregation to be used; however the protocol is general and can be applied to other problems as well. We're working hard on a production implementation of this protocol and expect to deploy it for Federated Learning applications in the near future.

Our work has only scratched the surface of what is possible. Federated Learning can't solve all machine learning problems (for example, learning to [recognize different dog breeds](#) by training on carefully labeled examples), and for many other models the necessary training data is already stored in the cloud (like training spam filters for Gmail). So Google will continue to advance the state-of-the-art for cloud-based ML, but we are also committed to ongoing research to expand the range of problems we can solve with Federated Learning. Beyond Gboard query suggestions, for example, we hope to improve the language models that power your keyboard based on what you actually type on your phone (which can have a style all its own) and photo rankings based on what kinds of photos people look at, share, or delete.

Applying Federated Learning requires machine learning practitioners to adopt new tools and a new way of thinking: model development, training, and evaluation with no direct access to or labeling of raw data, with communication cost as a limiting factor. We believe the user benefits of Federated Learning make tackling the technical challenges worthwhile, and are publishing our work with hopes of a widespread conversation within the machine learning community.

Jakub Konečný, H. Brendan McMahan, Felix X. Yu, Peter Richtárik, Ananda Theertha Suresh, Dave Bacon

## Federated Learning: Strategies for Improving Communication Efficiency

NIPS Private Multi-Party Machine Learning Workshop, 2016

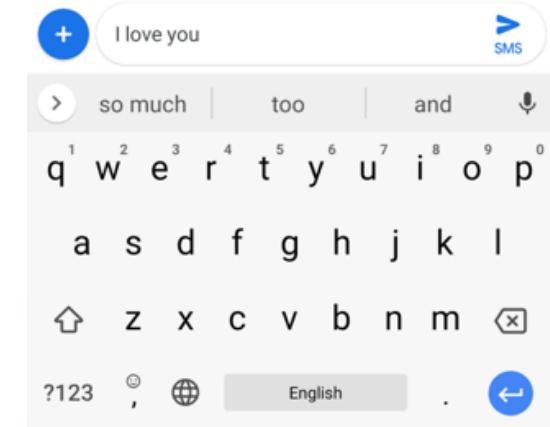
Jakub Konečný, H. Brendan McMahan, Daniel Ramage, Peter Richtárik

## Federated Optimization: Distributed Machine Learning for On-Device Intelligence

arXiv:1610.02527, 2016

# The First Federated Learning App: Next-Word Prediction

**Federated Learning** is collaborative machine learning from private data stored across a (large) number of clients/devices (e.g., hospitals, phones)



# My Team: 100+ Papers on Federated Learning



Peter Richtárik

Professor of Computer Science

King Abdullah University of Science and Technology (KAUST)

Address: Office 3145, Bldg 12, 4700 KAUST, Thuwal 23955-6900, Saudi Arabia  
E-mail: peter.richtarik@kaust.edu.sa



News   Old News   **Papers**   Talks   Video Talks   Events   Code   Team   Join   Bio   Teaching   Consulting

All papers are listed below in reverse chronological order in which they appeared online.

## Prepared in 2024

[271] Egor Shulgin and Peter Richtárik

**On the convergence of DP-SGD with adaptive clipping**

[arXiv] [method: QC-SGD, DP-QC-SGD]

[270] Igor Sokolov and Peter Richtárik

**MARINA-P: Superior performance in non-smooth federated optimization with adaptive stepsizes**

Federated Learning Paper

[arXiv] [method: MARINA-P]

[269] Artavazd Maranjyan, Abdurakhmon Sadiev, and Peter Richtárik

**Differentially private random block coordinate descent**

[arXiv] [method: DP-SKGD, DP-SkGD-BS, DP-CD]

[268] Elnur Gasanov and Peter Richtárik

**Speeding up stochastic proximal optimization in the high Hessian dissimilarity setting**

[arXiv] [method: L-SVRP]

[267] Yury Demidovich, Petr Ostroukhov, Grigory Malinovsky, Samuel Horváth, Martin Takáč, Peter Richtárik,

and Eduard Gorbunov

**Methods with local steps and random reshuffling for generally smooth non-convex federated optimization**

Federated Learning Paper

[arXiv] [method: Clip-LocalGDJ, CLERR, Clipped RR-CLI]

[266] Vladimir Malinovskii, Andrei Panferov, Ivan Ilin, Han Guo, Peter Richtárik, and Dan Alistarh

**Pushing the limits of large language model quantization via the linearity theorem**

LLM Paper

[arXiv] [method: HIGGS]

[265] Sarit Khirirat, Abdurakhmon Sadiev, Artem Riabinin, Eduard Gorbunov, and Peter Richtárik

**Error feedback under  $\|L_0\|_{L_1}$ -smoothness: normalization and momentum**

Federated Learning Paper

[arXiv] [method: ||EF21||, ||EF21-SGDM||]

[264] Wojciech Anyszka, Kaja Gruntkowska, Alexander Tyurin, and Peter Richtárik

**Tighter performance theory of FedExProx**

Federated Learning Paper

[arXiv] [method: FedExProx]

## The Next Generation Of Artificial Intelligence

Rob Toews Contributor 

I write about the big picture of artificial intelligence.

Follow

Oct 12, 2020, 09:22pm EDT

- 1. Unsupervised Learning**
- 2. Federated Learning**
- 3. Transformers**
- 4. Neural Network Compression**
- 5. Generative AI**
- 6. “System 2” Reasoning**

<https://www.forbes.com/sites/robtoews/2020/10/12/the-next-generation-of-artificial-intelligence/?sh=4d14f60159eb>

<https://www.forbes.com/sites/robtoews/2020/10/29/the-next-generation-of-artificial-intelligence-part-2/?sh=e02f2567a304>



Google



WeBank  
微众银行

IBM

SAMSUNG

Baidu 百度



# NATIONAL ARTIFICIAL INTELLIGENCE RESEARCH AND DEVELOPMENT STRATEGIC PLAN 2023 UPDATE

A Report by the

SELECT COMMITTEE ON ARTIFICIAL INTELLIGENCE  
of the  
NATIONAL SCIENCE AND TECHNOLOGY COUNCIL

May 2023



## The National Artificial Intelligence R&D Strategic Plan

### Table of Contents

<b>Executive Summary .....</b>	<b>vii</b>
<b>Introduction to the <i>National AI R&amp;D Strategic Plan: 2023 Update</i> .....</b>	<b>1</b>
AI as a National Priority.....	1
<b>Strategy 1: Make Long-Term Investments in Fundamental and Responsible AI Research .....</b>	<b>3</b>
Advancing Data-Focused Methodologies for Knowledge Discovery.....	3
Fostering Federated ML Approaches.....	4
Understanding Theoretical Capabilities and Limitations of AI.....	4
Pursuing Research on Scalable General-Purpose AI Systems.....	5
Developing AI Systems and Simulations Across Real and Virtual Environments.....	5
Enhancing the Perceptual Capabilities of AI Systems .....	5
Developing More Capable and Reliable Robots .....	6
Advancing Hardware for Improved AI.....	6
Creating AI for Improved Hardware .....	7
Embracing Sustainable AI and Computing Systems .....	8

# Federated Learning Issues & Tools

## Communication Complexity

local training

compression

stochastic approximation

variance reduction

momentum

## Data Heterogeneity

drift reduction

personalization

variable local training

## Device Heterogeneity

partial participation

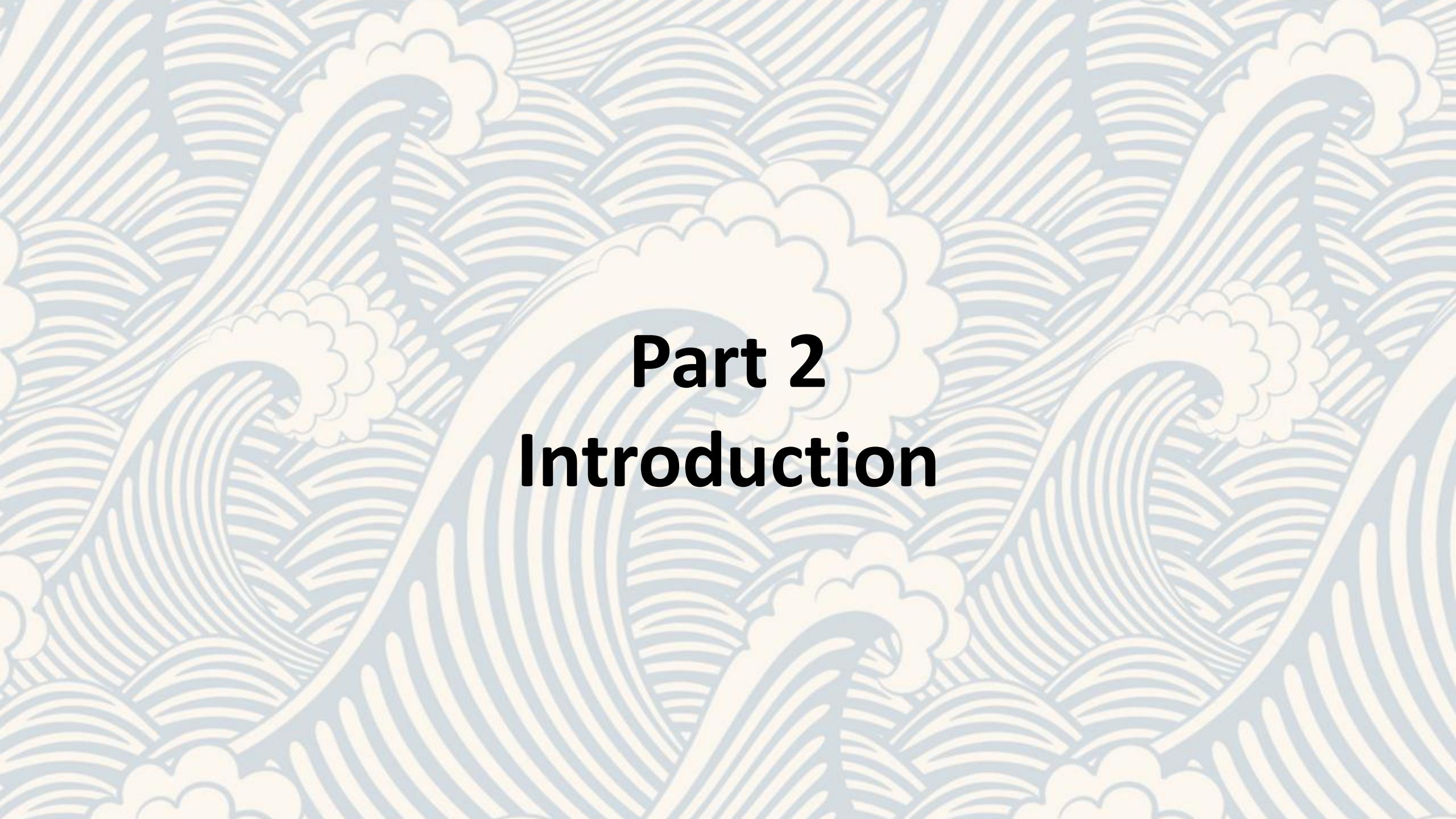
asynchronicity

## Privacy

differential privacy

homomorphic encryption

secure multiparty computation



# **Part 2**

# **Introduction**

# Optimization Problem

$$\min_{x \in \mathbb{R}^d} f(x) \stackrel{\text{def}}{=} \frac{1}{n} \sum_{i=1}^n f_i(x)$$

# model parameters / features

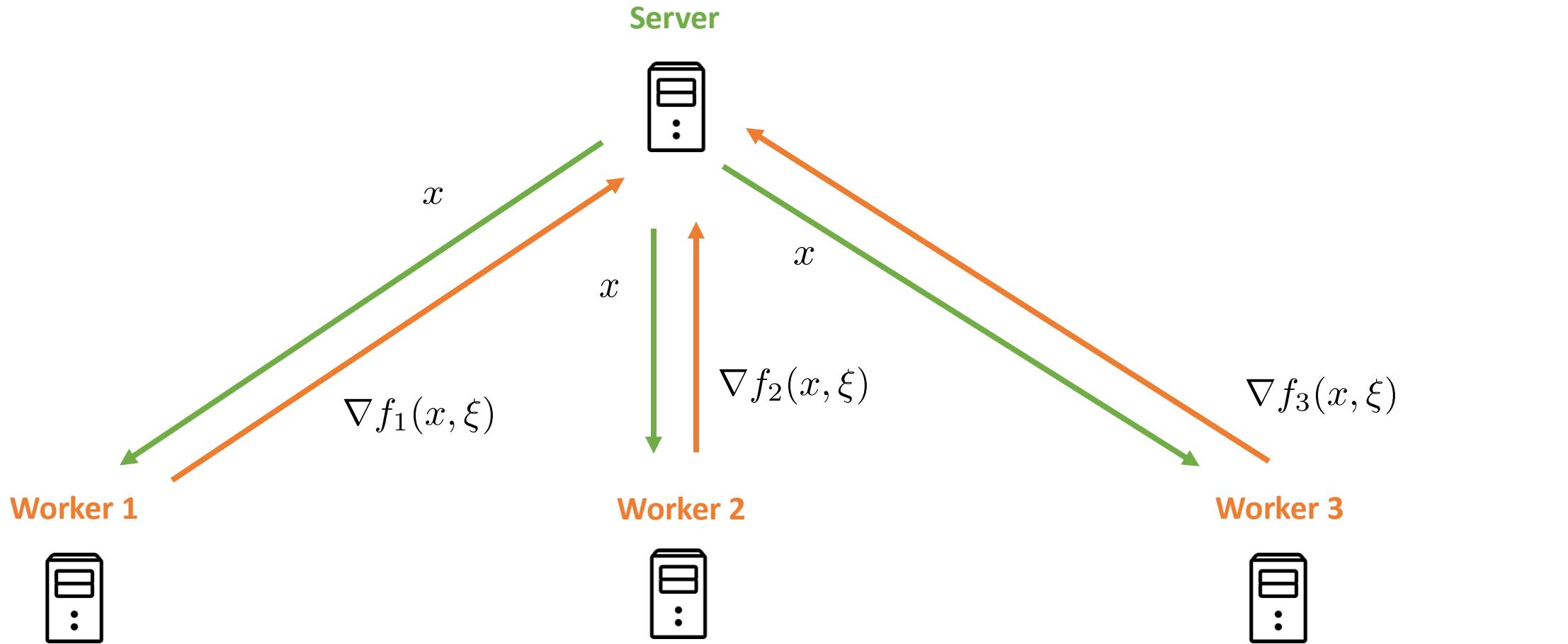
# parallel machines  
Loss on local data  $\mathcal{D}_i$  stored on machine  $i$   
 $f_i(x) := \mathbb{E}_{\xi \sim \mathcal{D}_i} [f_i(x, \xi)]$

- ! It takes  $\tau_i$  seconds for worker  $i$  to compute  $\nabla f_i(x, \xi)$ , where  $\xi \sim \mathcal{D}_i$        $0 < \tau_1 \leq \tau_2 \leq \dots \leq \tau_n$
- ! It takes  $\theta_i$  seconds for worker  $i$  to communicate  $g \in \mathbb{R}^d$  to the server

Find a (possibly random) vector  $\hat{x} \in \mathbb{R}^d$  such that  $\mathbb{E} [\|\nabla f(\hat{x})\|^2] \leq \varepsilon$

# Parallel Computing Architecture

$x$  gets updated by the server



$$f_1(x) := \mathbb{E}_{\xi \sim \mathcal{D}_1} [f_1(x, \xi)]$$

$$f_2(x) := \mathbb{E}_{\xi \sim \mathcal{D}_2} [f_2(x, \xi)]$$

$$f_3(x) := \mathbb{E}_{\xi \sim \mathcal{D}_3} [f_3(x, \xi)]$$

$\nabla f_1(x, \xi)$  compute time =  $\tau_1$  secs     $\nabla f_2(x, \xi)$  compute time =  $\tau_2$  secs     $\nabla f_3(x, \xi)$  compute time =  $\tau_3$  secs

# Three Types of Heterogeneity

<b>Data</b>	data distributions $\mathcal{D}_1, \dots, \mathcal{D}_n$ can be different
<b>Compute</b>	compute times $\tau_1, \dots, \tau_n$ are nonzero and can be different
<b>Communication</b>	communication times $\theta_1, \dots, \theta_n$ are nonzero and can be different

# Typical Assumptions

1

$$\inf f \in \mathbb{R}$$

2

$$f_i(x) := \mathbb{E}_{\xi \sim \mathcal{D}_i} [f_i(x, \xi)]$$

Gradient of local functions is Lipschitz:

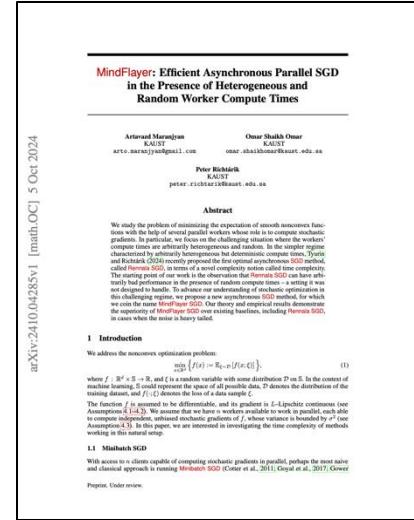
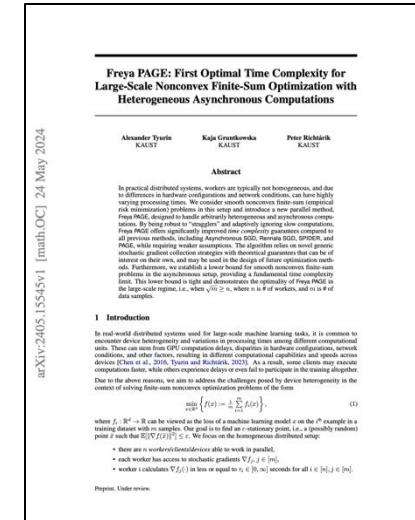
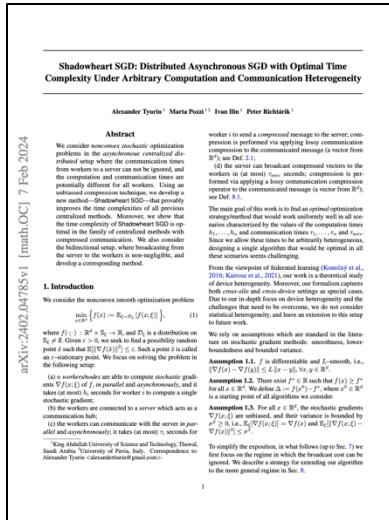
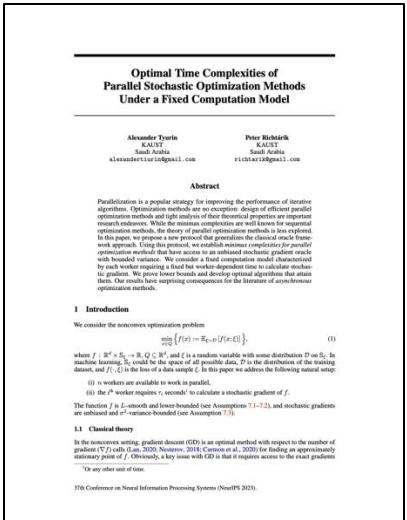
$$\max_{i \in \{1, \dots, n\}} \sup_{x \neq y} \frac{\|\nabla f_i(x) - \nabla f_i(y)\|}{\|x - y\|} \leq L$$



Stochastic gradients have bounded variance:

$$\max_{i \in \{1, \dots, n\}} \sup_{x \in \mathbb{R}^d} \mathbb{E}_{\xi \sim \mathcal{D}_i} [\|\nabla f_i(x, \xi) - \mathbb{E}_{\xi \sim \mathcal{D}_i} [\nabla f_i(x, \xi)]\|^2] \leq \sigma^2$$

# Our Papers on Optimal Parallel SGD



*First optimal  
parallel SGD under...*

# Our Papers

5/2023

Rennala SGD  
Malenia SGD  
Acc. Rennala SGD



Alexander Tyurin and P.R.

**Optimal time complexities of parallel stochastic optimization  
methods under a fixed computation model**

*NeurIPS 2023*

*... computation  
(and/or data) heterogeneity*

2/2024

Shadowheart SGD



Alexander Tyurin, Marta Pozzi, Ivan Ilin and P.R.

**Shadowheart SGD: Distributed asynchronous SGD with optimal  
time complexity under arbitrary computation and  
communication heterogeneity**

*arXiv:2402.04785, 2024*

*... communication  
(and computation) heterogeneity*

*[Rennala SGD as a special case]*

5/2024

Freya PAGE  
Freya SGD



Alexander Tyurin, Kaja Gruntkowska, and P.R.

**Freya PAGE: First optimal time complexity for large-scale  
nonconvex finite-sum optimization with heterogeneous  
asynchronous computations**

*arXiv:2405.1554, 2024*

*... computation heterogeneity for  
finite-sum problems*

in the large-scale regime:  $m \geq n^2$

5/2024

Fragile SGD, Amelie SGD  
+ accelerated variants



Alexander Tyurin and P.R.

**On the optimal time complexities in decentralized stochastic  
asynchronous optimization**

*arXiv:2405.16218, 2024*

*... computation and  
communication heterogeneity in  
the decentralized setup*

# Our Papers

10/2024

MindFlayer SGD  
Vecna SGD



Artavazd Maranjyan, Omar Shaikh Omar and P.R.

**MindFlayer: Efficient asynchronous parallel SGD in the presence of heterogeneous and random worker compute times**

*Conference on the Mathematical Theory of Deep Neural Networks (DeepMath 2024)*

*... random compute times*

1/2025

Ringmaster SGD



Artavazd Maranjyan, Alexander Tyurin and P.R.

**Ringmaster ASGD: The first asynchronous SGD with optimal time complexity**

*arXiv:2501.16168, 2025*

*1st fully asynchronous optimal parallel SGD*

2/2025

ATA



Artavazd Maranjyan, El Mehdi Saad, P.R. and Francesco Orabona

**ATA: Adaptive task allocation for efficient resource management in distributed machine learning**

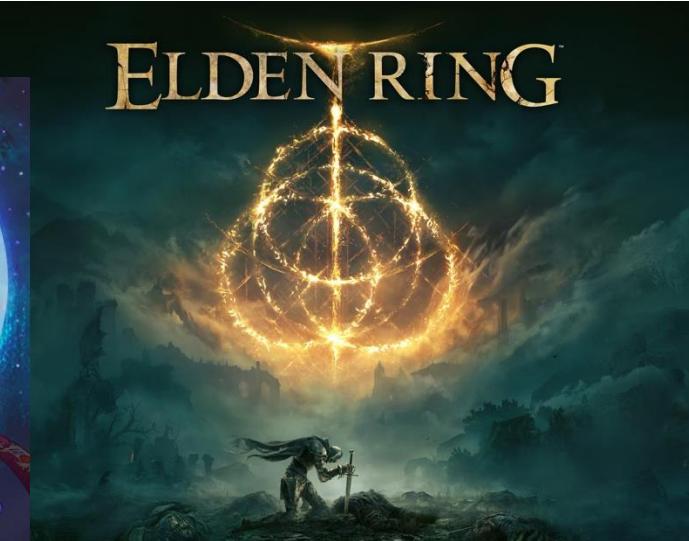
*arXiv:2502.00775, 2025*

# Peter, What About the Weird Algorithm Names?



**Rennala**

Rennala, Queen of the Full Moon is a Legend Boss in Elden Ring. Though not a demigod, Rennala is one of the shardbearers who resides in the Academy of Raya Lucaria. Rennala is a powerful sorceress, head of the Carian Royal family, and erstwhile leader of the Academy.



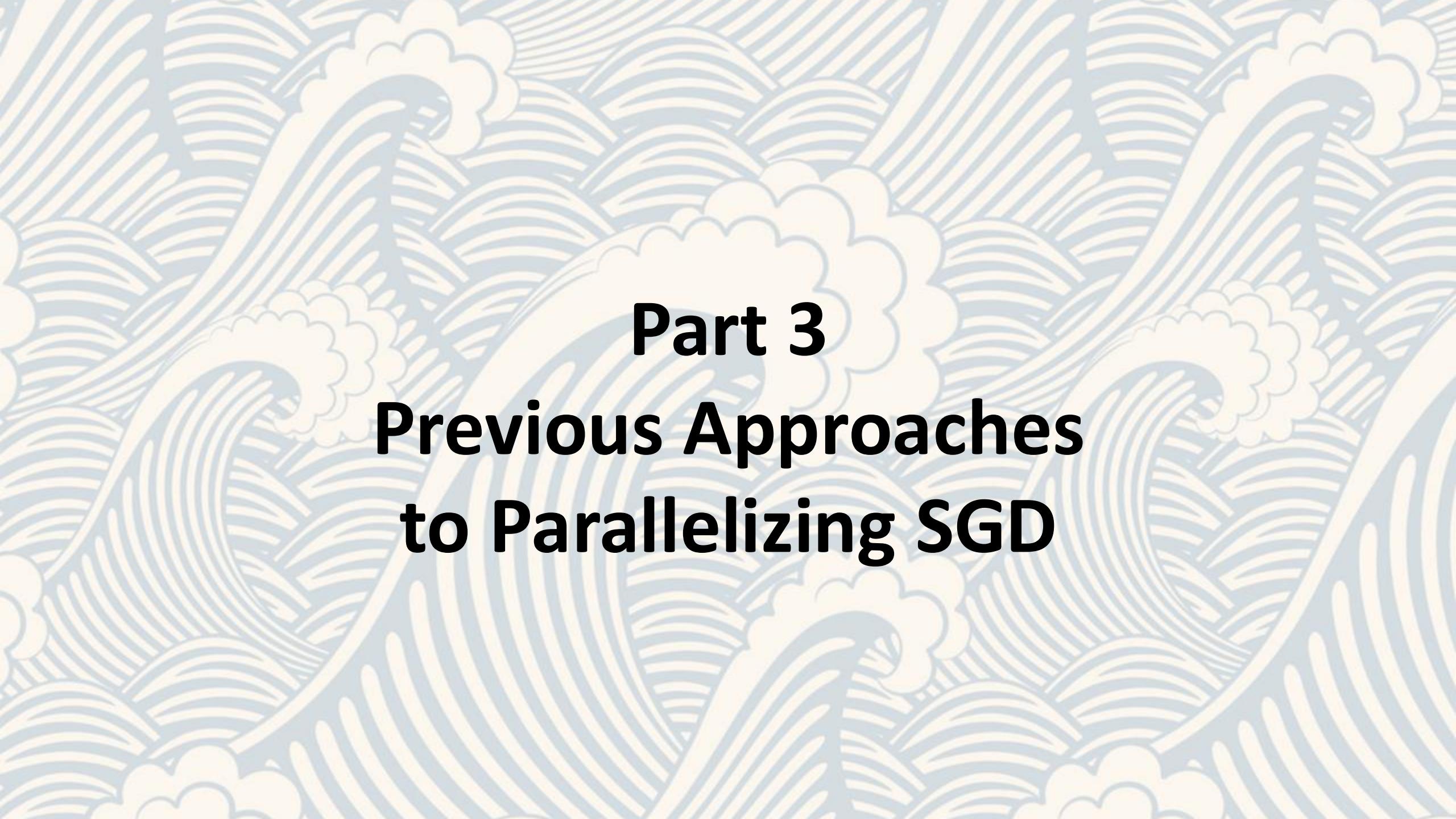
**Amelie**



**Shadowheart**

# Optimal Parallel Stochastic Gradient Methods

	Data Heterogeneity ( $D_i$ different)	Compute Heterogeneity ( $\tau_i$ different)	Communication Heterogeneity ( $\theta_i$ different)	Smooth Nonconvex	Smooth Convex	Infinite / Finite Sum?	Supports Decentralized Setup?	Optimal Time Complexity?
<b>Rennala SGD</b> Tyurin & R (NeurIPS '23)	✗	✓	0	✓		Inf	✗	✓
<b>Malenia SGD</b> Tyurin & R (NeurIPS '23)	✓	✓	0	✓		Inf	✗	✓
<b>Accelerated Rennala SGD</b> Tyurin & R (NeurIPS '23)	✗	✓	0		✓	Inf	✗	✓
<b>Shadowheart SGD</b> Tyurin, Pozzi, Ilin & R '24	✗	✓	✓	✓		Inf	✗	✓
<b>Freya PAGE</b> Tyurin, Gruntkowska & R '24	✗	✓	0	✓		Finite	✗	✓ big data regime
<b>Freya SGD</b> Tyurin, Gruntkowska & R '24	✗	✓	0	✓		Finite	✗	✗
<b>Fragile SGD</b> Tyurin & R '24	✗	✓	✓	✓		Inf	✓	nearly
<b>Amelie SGD</b> Tyurin & R '24	✓	✓	✓	✓		Inf	✓	✓



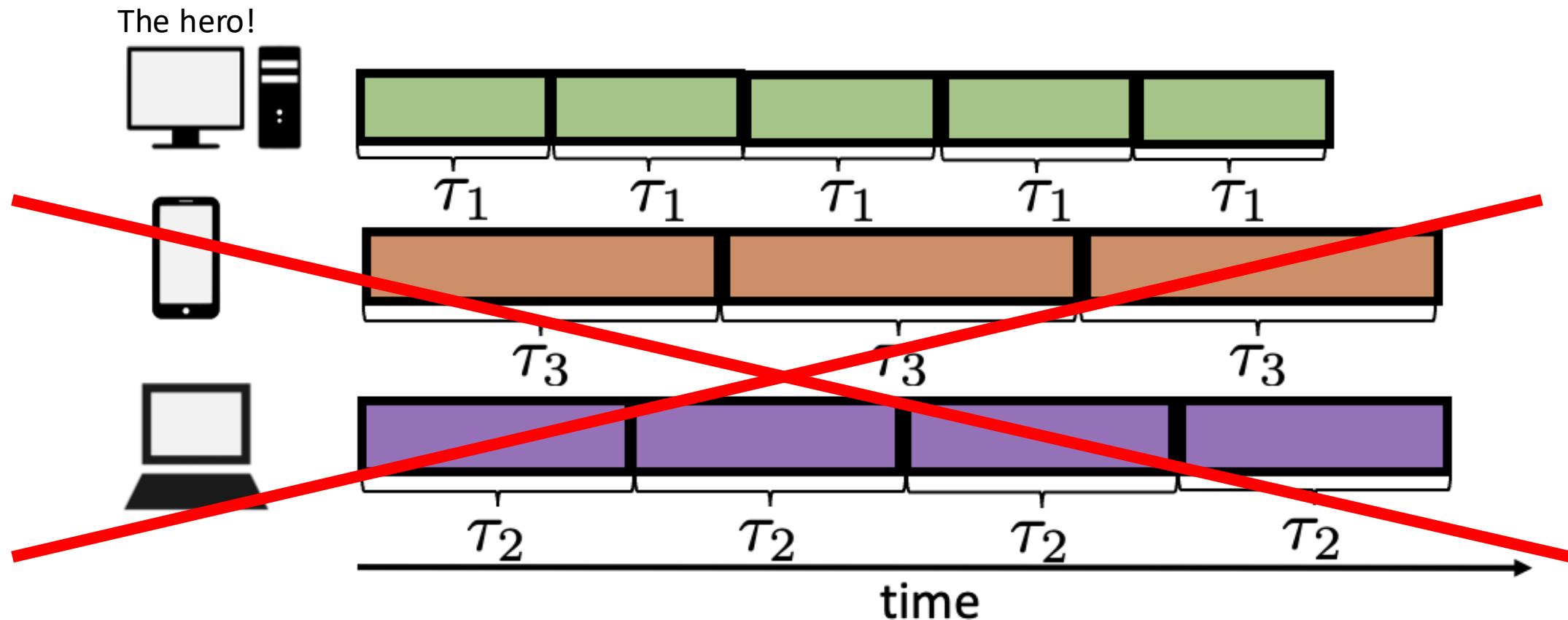
# **Part 3**

# **Previous Approaches**

# **to Parallelizing SGD**

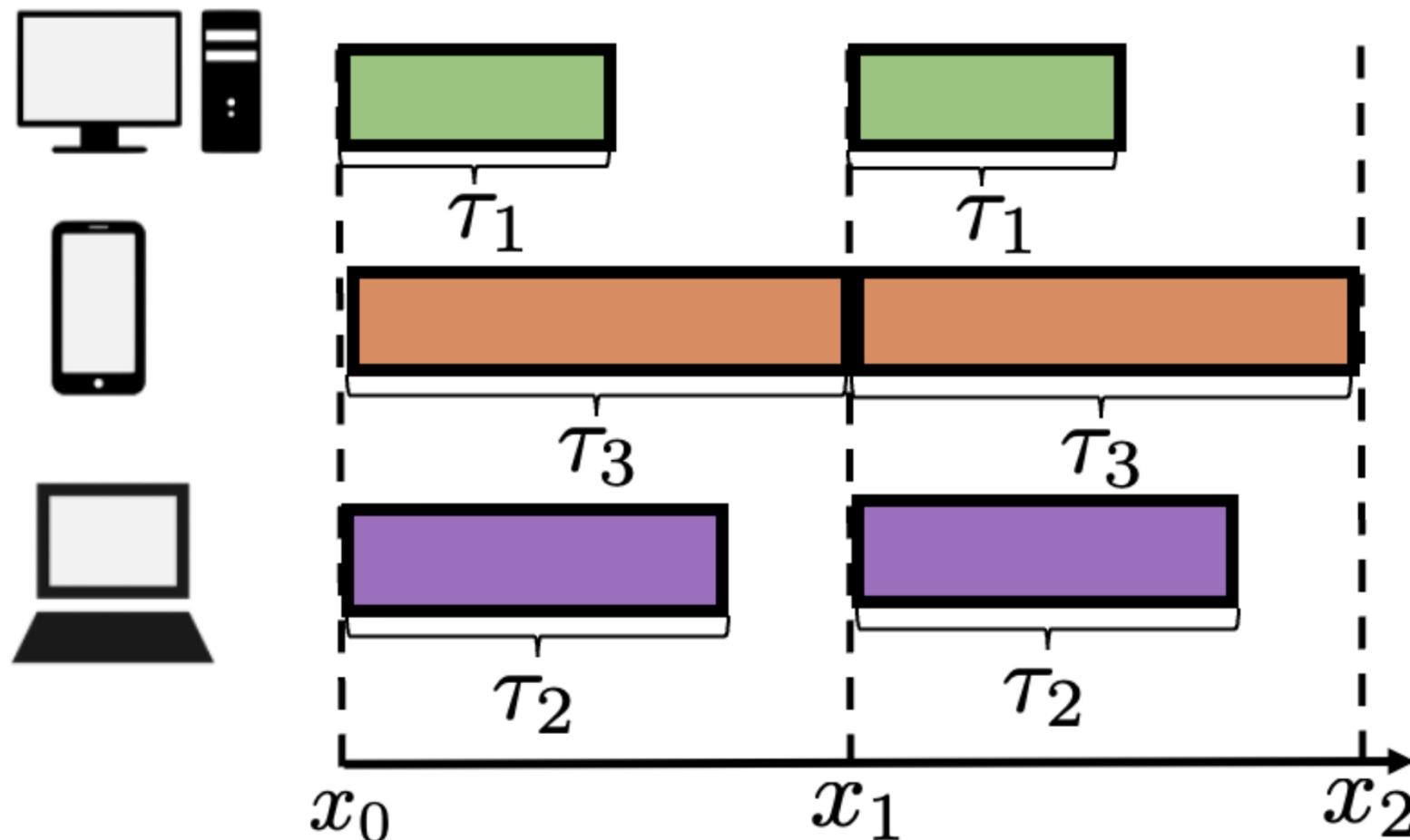
# Hero SGD

Algorithmic idea: **The fastest worker does it all!**



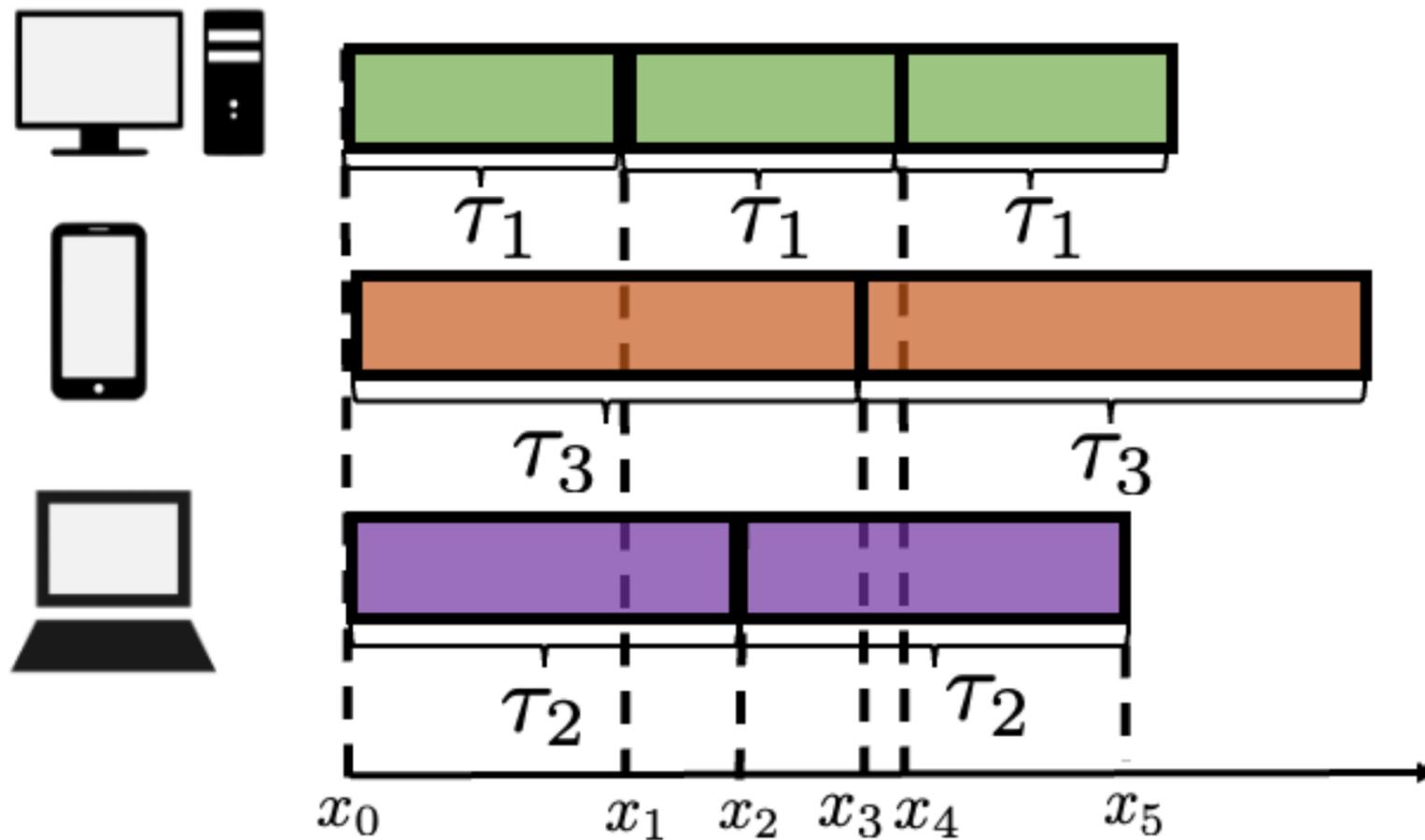
# (Fair) Minibatch SGD

Algorithmic idea: Each worker does one job only!



# Asynchronous SGD

Algorithmic idea: All workers are slaves and useful



# published in NIPS 2011

## HOGWILD!: A Lock-Free Approach to Parallelizing Stochastic Gradient Descent

Feng Niu  
leonn@cs.wisc.edu

Benjamin Recht  
brecht@cs.wisc.edu

Christopher Ré  
chrisre@cs.wisc.edu

Stephen J. Wright  
swright@cs.wisc.edu  
Computer Sciences Department  
University of Wisconsin-Madison  
Madison, WI 53706

### Abstract

Stochastic Gradient Descent (SGD) is a popular algorithm that can achieve state-of-the-art performance on a variety of machine learning tasks. Several researchers have recently proposed schemes to parallelize SGD, but all require performance-destroying memory locking and synchronization. This work aims to show using novel theoretical analysis, algorithms, and implementation that SGD can be implemented *without any locking*. We present an update scheme called HOGWILD! which allows processors access to shared memory with the possibility of overwriting each other's work. We show that when the associated optimization problem is *sparse*, meaning most gradient updates only modify small parts of the decision variable, then HOGWILD! achieves a nearly optimal rate of convergence. We demonstrate experimentally that HOGWILD! outperforms alternative schemes that use locking by an order of magnitude.

### 1 Introduction

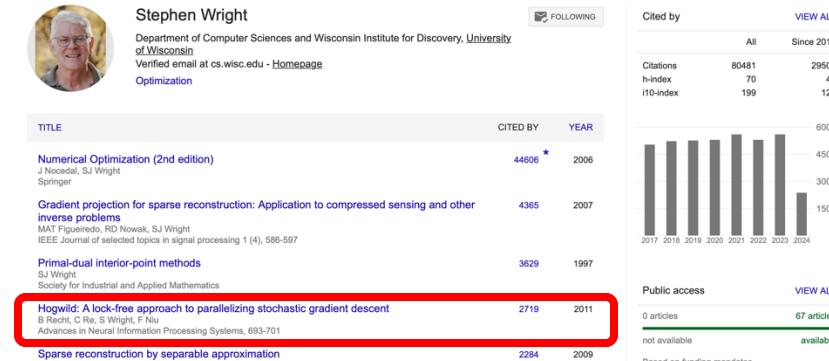
With its small memory footprint, robustness against noise, and rapid learning rates, Stochastic Gradient Descent (SGD) has proved to be well suited to data-intensive machine learning tasks [3, 5, 24]. However, SGD's scalability is limited by its inherently sequential nature; it is difficult to parallelize. Nevertheless, the recent emergence of inexpensive multicore processors and mammoth, web-scale data sets has motivated researchers to develop several clever parallelization schemes for SGD [4, 10, 12, 16, 27]. As many large data sets are currently pre-processed in a MapReduce-like parallel-processing framework, much of the recent work on parallel SGD has focused naturally on MapReduce implementations. MapReduce is a powerful tool developed at Google for extracting information from huge logs (e.g., "find all the urls from a 100TB of Web data") that was designed to ensure fault tolerance and to simplify the maintenance and programming of large clusters of machines [9]. But MapReduce is not ideally suited for online, numerically intensive data analysis. Iterative computation is difficult to express in MapReduce, and the overhead to ensure fault tolerance can result in dismal throughput. Indeed, even Google researchers themselves suggest that other systems, for example Dremel, are more appropriate than MapReduce for data analysis tasks [20].

For some data sets, the sheer size of the data dictates that one use a cluster of machines. However, there are a host of problems in which, after appropriate preprocessing, the data necessary for statistical analysis may consist of a few terabytes or less. For such problems, one can use a single inexpensive work station as opposed to a hundred thousand dollar cluster. Multicore systems have significant performance advantages, including (1) low latency and high throughput shared main memory (a processor in such a system can write and read the shared physical memory at over 12GB/s with latency in the tens of nanoseconds); and (2) high bandwidth off multiple disks (thousand-dollar RAID



Stephen Wright  
Department of Computer Sciences and Wisconsin Institute for Discovery, University of Wisconsin  
Verified email at cs.wisc.edu - Homepage  
Optimization

TITLE	CITED BY	YEAR
Numerical Optimization (2nd edition) J Nocedal, SJ Wright Springer	44606 *	2006
Gradient projection for sparse reconstruction: Application to compressed sensing and other inverse problems MAT Figueiredo, RD Nowak, SJ Wright IEEE Journal of selected topics in signal processing 1 (4), 586-597	4365	2007
Primal-dual interior-point methods SJ Wright Society for Industrial and Applied Mathematics	3629	1997
<b>Hogwild: A lock-free approach to parallelizing stochastic gradient descent</b> B Recht, C Re, S Wright, F Niu Advances in neural information processing systems, 693-701	2719	2011
Sparse reconstruction by separable approximation SJ Wright, RD Nowak, MAT Figueiredo IEEE Transactions on signal processing 57 (7), 2479-2493	2284	2009



## NeurIPS 2020 Test of Time Award

### Hogwild: A lock-free approach to parallelizing stochastic gradient descent

Authors Benjamin Recht, Christopher Re, Stephen Wright, Feng Niu

Publication date 2011

Conference Advances in Neural Information Processing Systems

Pages 693-701

Description Stochastic Gradient Descent (SGD) is a popular algorithm that can achieve state-of-the-art performance on a variety of machine learning tasks. Several researchers have recently proposed schemes to parallelize SGD, but all require performance-destroying memory locking and synchronization. This work aims to show using novel theoretical analysis, algorithms, and implementation that SGD can be implemented without any locking. We present an update scheme called Hogwild which allows processors access to shared memory with the possibility of overwriting each other's work. We show that when the associated optimization problem is sparse, meaning most gradient updates only modify small parts of the decision variable, then Hogwild achieves a nearly optimal rate of convergence. We demonstrate experimentally that Hogwild outperforms alternative schemes that use locking by an order of magnitude.

Total citations Cited by 2719

A bar chart showing the total number of citations per year from 2012 to 2024. The chart shows a sharp increase in 2011 (2719 citations) and a general decline thereafter.

Scholar articles [Hogwild: A lock-free approach to parallelizing stochastic gradient descent](#)  
B Recht, C Re, S Wright, F Niu - Advances in neural information processing systems, 2011  
Cited by 2718 Related articles All 35 versions

[Hogwild: A lock-free approach to parallelizing stochastic gradient descent](#)  
B Recht, C Re, S Wright, F Niu - Advances in neural information processing systems, 2011  
Cited by 2 Related articles

# Our Inspiration: Two Beautiful Papers

**Asynchronous SGD Beats Minibatch SGD Under Arbitrary Delays**

---

Konstantin Mishchenko   Francis Bach   Mathieu Even   Blake Woodworth

DI ENS, Ecole normale supérieure,  
Université PSL, CNRS, INRIA  
75005 Paris, France

**Abstract**

The existing analysis of asynchronous stochastic gradient descent (SGD) degrades dramatically when any delay is large, giving the impression that performance depends primarily on the delay. On the contrary, we prove much better guarantees for the same asynchronous SGD algorithm regardless of the delays in the gradients, depending instead just on the number of parallel devices used to implement the algorithm. Our guarantees are strictly better than the existing analyses, and we also argue that asynchronous SGD outperforms synchronous minibatch SGD in the settings we consider. For our analysis, we introduce a novel recursion based on “virtual iterates” and delay-adaptive stepsizes, which allow us to derive state-of-the-art guarantees for both convex and non-convex objectives.

**1 Introduction**

We consider solving stochastic optimization problems of the form

$$\min_{\mathbf{x} \in \mathbb{R}^d} \{F(\mathbf{x}) := \mathbb{E}_{\xi \sim \mathcal{D}} f(\mathbf{x}; \xi)\}, \quad (1)$$

which includes machine learning (ML) training objectives, where  $f(\mathbf{x}; \xi)$  represents the loss of a model parameterized by  $\mathbf{x}$  on the datum  $\xi$ . Depending on the application,  $\mathcal{D}$  could represent a finite dataset of size  $n$  or a population distribution. In recent years, such stochastic optimization problems have continued to grow rapidly in size, both in terms of the dimension  $d$  of the optimization variable—i.e., the number of model parameters in ML—and in terms of the quantity of data—i.e., the number of samples  $\xi_1, \dots, \xi_n \sim \mathcal{D}$  being used. With  $d$  and  $n$  regularly reaching the tens or hundreds of billions, it is increasingly necessary to use parallel optimization algorithms to handle the large scale and to benefit from data stored on different machines.

There are many ways of employing parallelism to solve (1), but the most popular approaches in practice are first-order methods based on stochastic gradient descent (SGD). At each iteration, SGD employs stochastic estimates of  $\nabla F$  to update the parameters as  $\mathbf{x}_k = \mathbf{x}_{k-1} - \gamma_k \nabla/(x_{k-1}; \xi_{k-1})$  for an i.i.d. sample  $\xi_{k-1} \sim \mathcal{D}$ . Given  $M$  machines capable of computing these stochastic gradient estimates  $\nabla f(\mathbf{x}; \xi)$  in parallel, one approach to parallelizing SGD is what we call “Minibatch SGD.” This refers to a synchronous, parallel algorithm that dispatches the current parameters  $\mathbf{x}_{k-1}$  to each of the  $M$  machines, waits while they compute and communicate back their gradient estimates  $\mathbf{g}_{k-1}^1, \dots, \mathbf{g}_{k-1}^M$ , and then takes a minibatch SGD step  $\mathbf{x}_k = \mathbf{x}_{k-1} - \gamma_k \cdot \frac{1}{M} \sum_{m=1}^M \mathbf{g}_{k-1}^m$ . This is a natural idea with long history [16, 18, 55] and it is commonly used in practice [e.g., 22]. However, since Minibatch SGD waits for all  $M$  of the machines to finish computing their gradient estimates before updating, it proceeds only at the speed of the *slowest* machine.

There are several possible sources of delays: nodes may have heterogeneous hardware with different computational throughputs [23, 25], network latency can slow the communication of gradients, and

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

arXiv: June 15, 2022

**Sharper Convergence Guarantees for Asynchronous SGD for Distributed and Federated Learning**

---

Anastasia Koloskova   Sebastian U. Stich   Martin Jaggi

EPFL   CISPA\*   EPFL  
anastasia.koloskova@epfl.ch   stich@cispa.de   martin.jaggi@epfl.ch

**Abstract**

We study the asynchronous stochastic gradient descent algorithm for distributed training over  $n$  workers which have varying computation and communication frequency over time. In this algorithm, workers compute stochastic gradients in parallel at their own pace and return those to the server without any synchronization. Existing convergence rates for this algorithm for non-convex smooth objectives depend on the maximum gradient delay  $\tau_{\max}$  and show that an  $\varepsilon$ -stationary point is reached after  $\mathcal{O}(\sigma^2 c^{-2} + \tau_{\max} c^{-1})$  iterations, where  $\sigma$  denotes the variance of stochastic gradients.

In this work we obtain (i) a tighter convergence rate of  $\mathcal{O}(\sigma^2 \varepsilon^{-2} + \sqrt{\tau_{\max} \tau_{avg} \varepsilon^{-1}})$  without any change in the algorithm, where  $\tau_{avg}$  is the average delay, which can be significantly smaller than  $\tau_{\max}$ . We also provide (ii) a simple delay-adaptive learning rate scheme, under which asynchronous SGD achieves a convergence rate of  $\mathcal{O}(\sigma^2 \varepsilon^{-2} + \tau_{avg} \varepsilon^{-1})$ , and does not require any extra hyperparameter tuning nor extra communications. Our result allows to show for the first time that asynchronous SGD is always faster than mini-batch SGD. In addition, (iii) we consider the case of heterogeneous functions motivated by federated learning applications and improve the convergence rate by proving a weaker dependence on the maximum delay compared to prior works. In particular, we show that the heterogeneity term in convergence rate is only affected by the average delay within each worker.

**1 Introduction**

The stochastic gradient descent (SGD) algorithm [43, 13] and its variants (momentum SGD, Adam, etc.) form the foundation of modern machine learning and frequently achieve state of the art results. With recent growth in the size of models and available training data, parallel and distributed versions of SGD are becoming increasingly important [57, 17, 16]. Without those, modern state-of-the-art language models [44], generative models [40, 41], and many others [50] would not be possible. In the distributed setting, also known as data-parallel training, optimization is distributed over many compute devices working in parallel (e.g. cores, or GPUs on a cluster) in order to speed up training. Every worker computes gradients on a subset of the training data, and the resulting gradients are aggregated (averaged) on a server.

The same type of SGD variants also form the core algorithms for federated learning applications [34, 24] where the training process is naturally distributed over many user devices, or clients, that keep their local data private, and only transfer (e.g. encrypted or differentially private) gradients to the server.

A rich literature exists on the convergence theory of above mentioned parallel SGD methods, see e.g. [17, 13] and references therein. Plain parallel SGD still faces many challenges in practice, motivat-

\*CISPA Helmholtz Center for Information Security

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

arXiv: June 16, 2022

---

**Optimal Time Complexities of  
Parallel Stochastic Optimization Methods  
Under a Fixed Computation Model**

---

Alexander Tyurin  
KAUST  
Saudi Arabia  
alexander.tyurin@gmail.com

Peter Richtárik  
KAUST  
Saudi Arabia  
richtarik@gmail.com

**Abstract**

Parallelization is a popular strategy for improving the performance of iterative algorithms. Optimal time complexities are an exception of efficient parallel optimization methods and tight analysis of their theoretical properties are an important research endeavor. While the minimax complexities are well known for sequential optimization methods, the theory of parallel optimization methods is less explored. In this paper, we propose a framework for analyzing the performance of parallel optimization methods under a fixed computation model. We introduce a new framework approach. Using this protocol, we establish *minimax complexities for parallel optimization methods* that have access to an unbiased stochastic gradient oracle with a bounded variance. Our results apply to a wide class of parallel models characterized by each worker requiring a fixed but worker-dependent time to calculate stochastic gradients. We prove lower bounds and develop optimal algorithms that attain them. Our results have surprising consequences for the literature of asynchronous optimization methods.

**1 Introduction**

We consider the nonconvex optimization problem

$$\min_{x \in Q} f(x) := \mathbb{E}_{\xi \sim \mathcal{D}} [f(x|\xi)], \quad (1)$$

where  $f: \mathbb{R}^n \times \mathcal{S}_\xi \rightarrow \mathbb{R}$ ,  $Q \subseteq \mathbb{R}^n$ , and  $\xi$  is a random variable with some distribution  $\mathcal{P}$  on  $\mathcal{S}_\xi$ . In machine learning,  $\mathcal{S}_\xi$  could be a set of all possible data,  $\mathcal{D}$  is the distribution of the training dataset, and  $f(\cdot, \cdot)$  is the loss of a data sample  $\xi$ . In this paper we address the following natural setup:

- i)  $n$  workers are available to work in parallel;
- ii) the  $i$  worker requires  $\tau_i$  seconds<sup>1</sup> to calculate a stochastic gradient of  $f$ .

The function  $f$  is  $L$ -smooth and lower-bounded (see Assumptions 7.1–7.2), and stochastic gradients are unbiased and  $\sigma^2$ -variance-bounded (see Assumption 7.3).

**1.1 Classical theory**

In the nonconvex setting, gradient descent (GD) is an optimal method with respect to the number of gradient ( $\nabla f$ ) calls (Lan, 2020; Nesterov, 2018; Cenner et al., 2020) for finding an approximately stationary point of  $f$ . Obviously, a key issue with GD is that it requires access to the exact gradients

<sup>1</sup>Or any other unit of time.

37th Conference on Neural Information Processing Systems (NeurIPS 2023).

# Part 4

# Rennala SGD



Alexander Tyurin and P.R.  
**Optimal time complexities of parallel stochastic optimization  
methods under a fixed computation model**  
NeurIPS 2023

# Setup

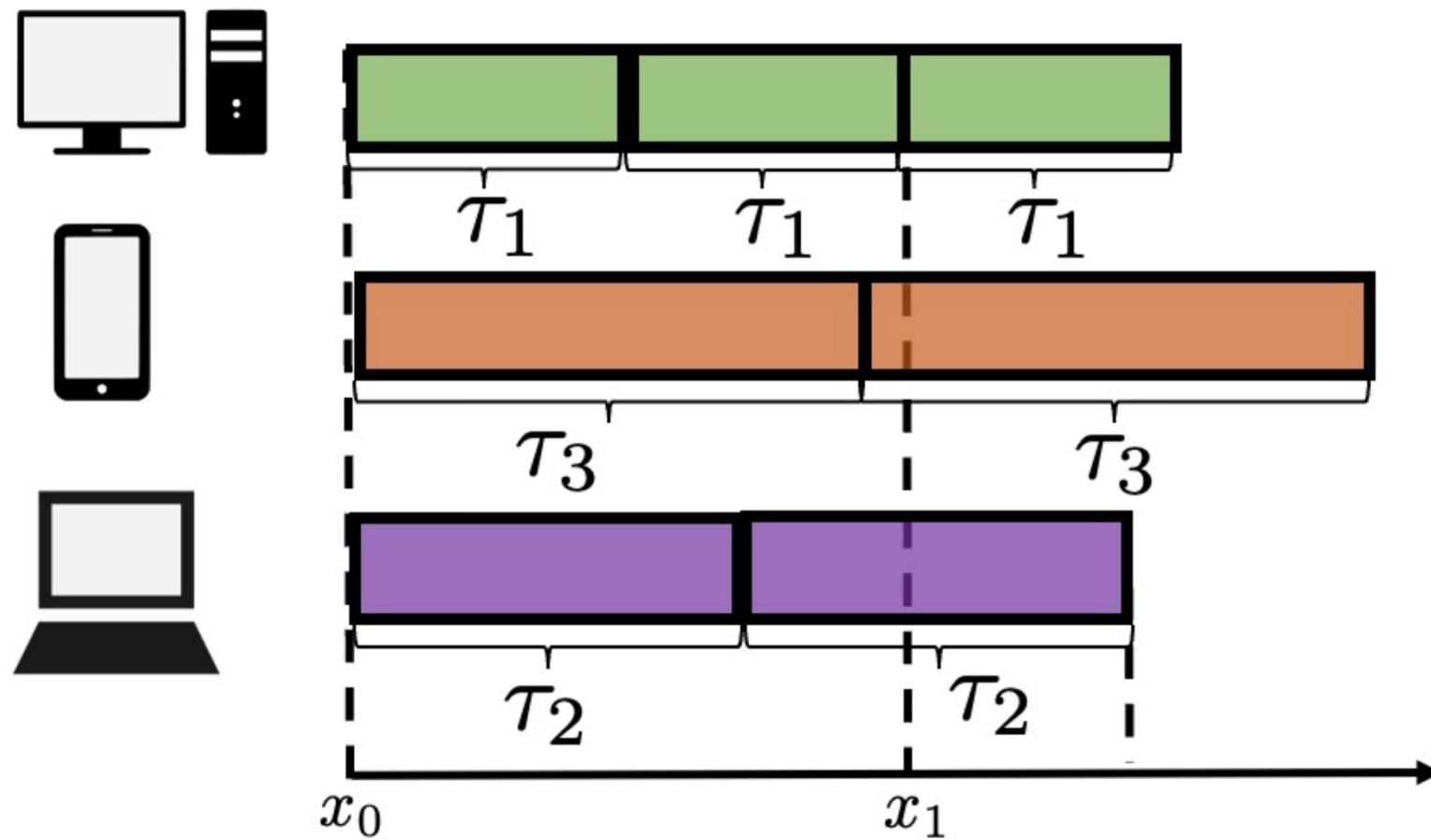
## Optimal Parallel Stochastic Gradient Methods



	Data Heterogeneity ( $\mathcal{D}_i$ different)	Compute Heterogeneity ( $\tau_i$ different)	Communication Heterogeneity ( $\theta_i$ different)	Smooth Nonconvex	Smooth Convex	Infinite / Finite Sum?	Supports Decentralized Setup?	Optimal Time Complexity?
<b>Rennala SGD</b> Tyurin & R (NeurIPS '23)	✗	✓	0	✓		Inf	✗	✓
<b>Malenia SGD</b> Tyurin & R (NeurIPS '23)	✓	✓	0	✓		Inf	✗	✓
<b>Accelerated Rennala SGD</b> Tyurin & R (NeurIPS '23)	✗	✓	0		✓	Inf	✗	✓
<b>Shadowheart SGD</b> Tyurin, Pozzi, Ilin & R '24	✗	✓	✓	✓		Inf	✗	✓
<b>Freya PAGE</b> Tyurin, Grunkowska & R '24	✗	✓	0	✓		Finite	✗	✓ big data regime
<b>Freya SGD</b> Tyurin, Grunkowska & R '24	✗	✓	0	✓		Finite	✗	
<b>Fragile SGD</b> Tyurin & R '24	✗	✓	✓	✓		Inf	✓	nearly
<b>Amelie SGD</b> Tyurin & R '24	✓	✓	✓	✓		Inf	✓	✓

# Rennala SGD

Algorithmic idea: Minibatch SGD with asynchronous minibatch collection



# Upper Bound

## Theorem (informal)

Assume data homogeneity and zero communication times.

Then Rennala SGD solves the problem in

Number of parallel machines

$$96 \times \min_{m \in \{1, \dots, n\}} \left( \frac{1}{m} \sum_{i=1}^m \frac{1}{\tau_i} \right)^{-1} \left( \frac{L\Delta}{\varepsilon} + \frac{L\Delta\sigma^2}{\varepsilon^2 m} \right)$$

seconds.

Compute times

$$0 < \tau_1 \leq \tau_2 \leq \dots \leq \tau_n$$

Algorithm outputs  $\hat{x}$  such that  $\mathbb{E} [\|\nabla f(\hat{x})\|^2] \leq \varepsilon$

Gradient of  $f$  is  $L$ -Lipschitz

$$\Delta := f(x^0) - \inf f$$

$$\sup_{x \in \mathbb{R}^d} \mathbb{E}_{\xi \sim \mathcal{D}} [\|\nabla f(x, \xi) - \nabla f(x)\|^2] \leq \sigma^2$$

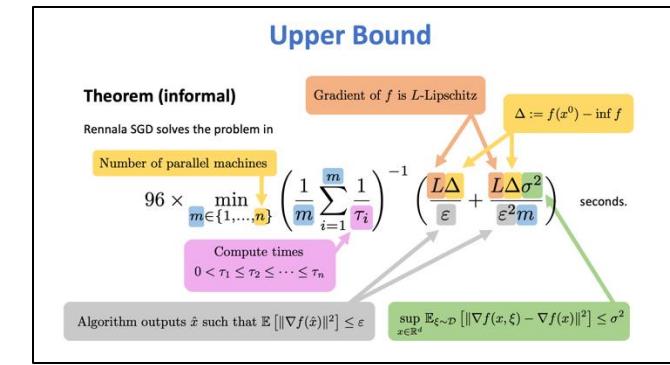
# Matching Lower Bound

## Theorem (informal)

It is not possible to design a method that will find a solution faster than in

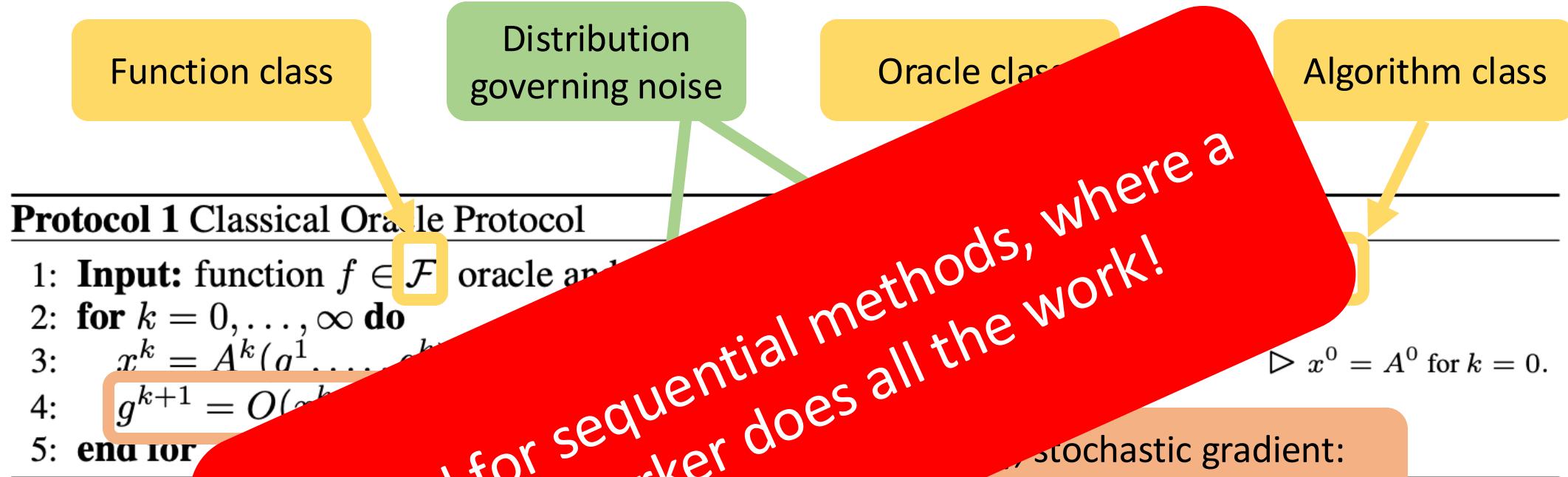
$$\Omega \left( \min_{m \in \{1, \dots, n\}} \left( \frac{1}{m} \sum_{i=1}^m \frac{1}{\tau_i} \right)^{-1} \left( \frac{L\Delta}{\varepsilon} + \frac{L\Delta\sigma^2}{\varepsilon^2 m} \right) \right)$$

seconds.



Rennala SGD = first optimal parallel SGD

# Classical Oracle: Keeps Track of # Iterations



Iteration complexity (in terms of oracle calls):

$$m_{\text{oracle}}(\mathcal{A}, \mathcal{F}) = \sup_{A \in \mathcal{A}} \sup_{f \in \mathcal{F}} \inf_{(O, \mathcal{D}) \in \mathcal{O}(f)} \left\{ k \in \mathbb{N} \mid \mathbb{E} [\|\nabla f(x^k)\|^2] \leq \varepsilon \right\}$$

[Nemirovsky and Yudin, 1983] [Nesterov, 2018]  
[Carmon et al, 2020] [Arjevani et al, 2022]

# New Oracle: Keeps Track of Time

## Protocol 2 Time Oracle Protocol

```
1: Input: functions  $f \in \mathcal{F}$ , oracle and distribution  $(O, \mathcal{D}) \in \mathcal{O}(f)$ 
2:  $s^0 = 0$ 
3: for  $k = 0, \dots, \infty$  do
4:    $(t^{k+1}, x^k) = A^k(g^1, \dots, g^k),$   $\triangleright t^k \in \mathbb{R}$ 
5:    $(s^{k+1}, g^{k+1}) = O(t^{k+1}, x^k, s^k, g^k)$ 
6: end for
```

$$\triangleright t^{k+1} \geq t^k$$

Iteration complexity:

$$m_{\text{oracle}}(\mathcal{A}, \mathcal{F}) := \inf_{A \in \mathcal{A}} \sup_{f \in \mathcal{F}} \inf_{(O, \mathcal{D}) \in \mathcal{O}(f)} \inf \left\{ k \in \mathbb{N} \mid \mathbb{E} [\|\nabla f(x^k)\|^2] \leq \varepsilon \right\}$$

Natural for parallel methods!

Time complexity (with respect to time measure):

$$m_{\text{time}}(\mathcal{A}, \mathcal{F}) := \inf_{A \in \mathcal{A}} \sup_{f \in \mathcal{F}} \sup_{(O, \mathcal{D}) \in \mathcal{O}(f)} \inf \left\{ t \geq 0 \mid \mathbb{E} \left[ \inf_{k \in S_t} \|\nabla f(x^k)\|^2 \right] \leq \varepsilon \right\}$$

$$S_t := \{k \in \mathbb{N} \cup \{0\} \mid t^k \leq t\}$$

# Data Homogeneous Regime

Method	Time Complexity
Minibatch SGD	$\tau_n \left( \frac{L\Delta}{\varepsilon} + \frac{\sigma^2 L\Delta}{n\varepsilon^2} \right)$
Asynchronous SGD (Cohen et al., 2021) (Koloskova et al., 2022) (Mishchenko et al., 2022)	$\left( \frac{1}{n} \sum_{i=1}^n \frac{1}{\tau_i} \right)^{-1} \left( \frac{L\Delta}{\varepsilon} + \frac{\sigma^2 L\Delta}{n\varepsilon^2} \right)$
Rennala SGD (Theorem 7.5)	$\min_{m \in [n]} \left[ \left( \frac{1}{m} \sum_{i=1}^m \frac{1}{\tau_i} \right)^{-1} \left( \frac{L\Delta}{\varepsilon} + \frac{\sigma^2 L\Delta}{m\varepsilon^2} \right) \right]$
Lower Bound (Theorem 6.4)	$\min_{m \in [n]} \left[ \left( \frac{1}{m} \sum_{i=1}^m \frac{1}{\tau_i} \right)^{-1} \left( \frac{L\Delta}{\varepsilon} + \frac{\sigma^2 L\Delta}{m\varepsilon^2} \right) \right]$

# Experimental Results (Sample)

$$\tau_i = \sqrt{i} \text{ seconds}$$

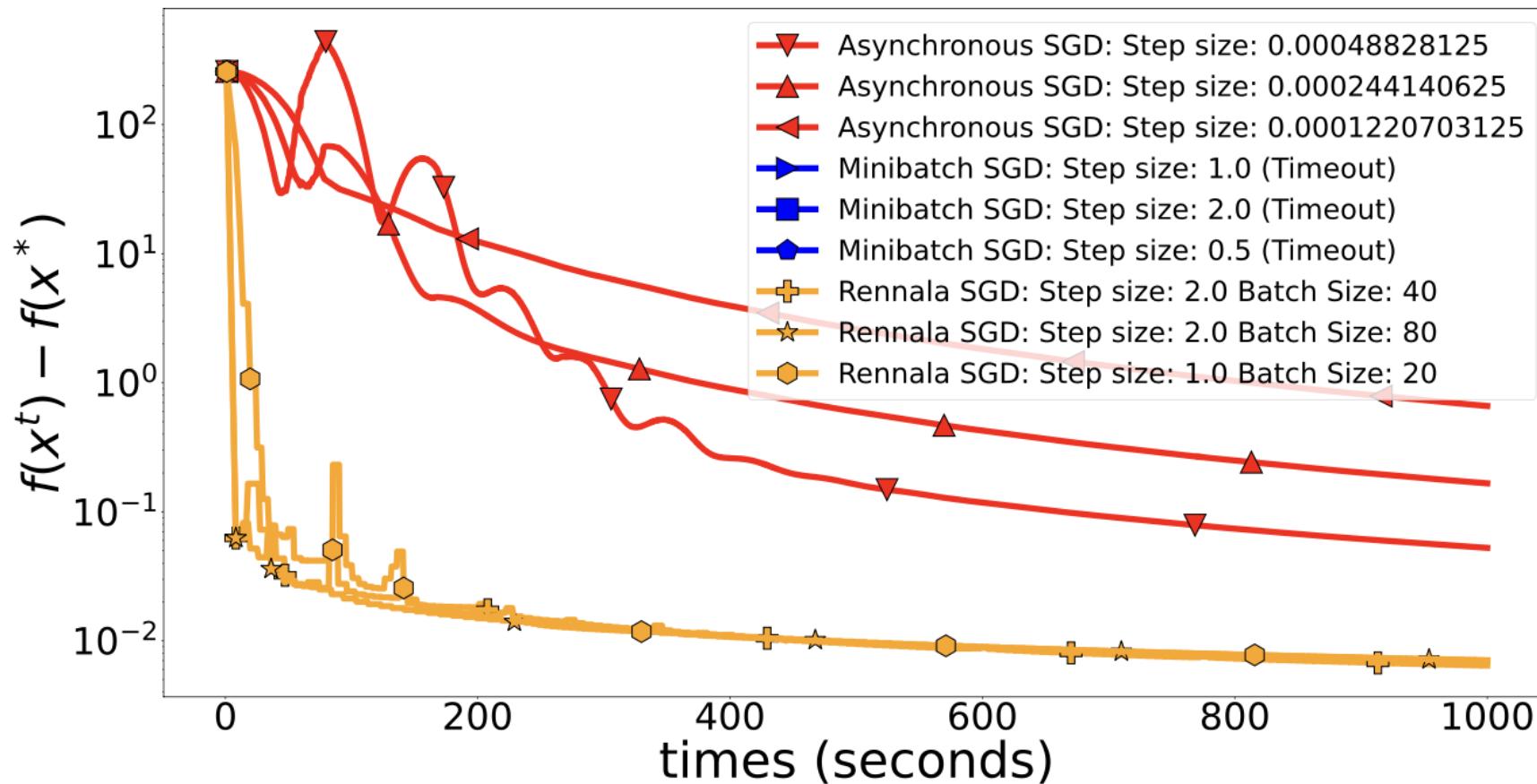


Figure 3: # of workers  $n = 10000$ .



**The End  
(kind of)**

---

**Optimal Time Complexities of  
Parallel Stochastic Optimization Methods  
Under a Fixed Computation Model**

---

Alexander Tyurin  
KAUST  
Saudi Arabia  
alexander.tyurin@gmail.com

Peter Richtárik  
KAUST  
Saudi Arabia  
richtarik@gmail.com

**Abstract**

Parallelization is a popular strategy for improving the performance of iterative algorithms. Optimal time complexities are an exception of efficient parallel optimization methods and tight analysis of their theoretical properties are an important research endeavor. While the minimax complexities are well known for sequential optimization methods, the theory of parallel optimization methods is less explored. In this paper, we propose a framework for analyzing the performance of parallel optimization methods under a fixed computation model. We introduce a framework approach. Using this protocol, we establish *minimax complexities for parallel optimization methods* that have access to an unbiased stochastic gradient oracle with a bounded variance. Our results apply to a wide class of parallel models characterized by each worker requiring a fixed but worker-dependent time to calculate stochastic gradients. We prove lower bounds and develop optimal algorithms that attain them. Our results have surprising consequences for the literature of asynchronous optimization methods.

**1 Introduction**

We consider the nonconvex optimization problem

$$\min_{x \in Q} \{ f(x) := \mathbb{E}_{\xi \sim \mathcal{D}} [f(x|\xi)] \}, \quad (1)$$

where  $f: \mathbb{R}^n \times \mathcal{S}_\xi \rightarrow \mathbb{R}$ ,  $Q \subseteq \mathbb{R}^n$ , and  $\xi$  is a random variable with some distribution  $\mathcal{P}$  on  $\mathcal{S}_\xi$ . In machine learning,  $\mathcal{S}_\xi$  could be a set of all possible data,  $\mathcal{D}$  is the distribution of the training dataset, and  $f(\cdot; \cdot)$  is the loss of a data sample  $\xi$ . In this paper we address the following natural setup:

- i)  $n$  workers are available to work in parallel;
- ii) the  $i$  worker requires  $\tau_i$  seconds<sup>1</sup> to calculate a stochastic gradient of  $f$ .

The function  $f$  is  $L$ -smooth and lower-bounded (see Assumptions 7.1–7.2), and stochastic gradients are unbiased and  $\sigma^2$ -variance-bounded (see Assumption 7.3).

**1.1 Classical theory**

In the nonconvex setting, gradient descent (GD) is an optimal method with respect to the number of gradient ( $\nabla f$ ) calls (Lan, 2020; Nesterov, 2018; Cenner et al., 2020) for finding an approximately stationary point of  $f$ . Obviously, a key issue with GD is that it requires access to the exact gradients

<sup>1</sup>Or any other unit of time.

37th Conference on Neural Information Processing Systems (NeurIPS 2023).

# Part 5

# Two Extensions



Alexander Tyurin and P.R.  
**Optimal time complexities of parallel stochastic optimization  
methods under a fixed computation model**  
NeurIPS 2023

# **Extension 1**

# **Handling Data Heterogeneity**

## **(Malenia SGD)**

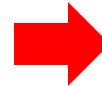
# Malenia SGD: Setup

$$\min_{x \in \mathbb{R}^d} \left\{ f(x) = \frac{1}{n} \sum_{i=1}^n f_i(x) \right\}$$

$$f_i(x) := \mathbb{E}_{\xi \sim \mathcal{D}_i} [f_i(x, \xi)]$$

## Optimal Parallel Stochastic Gradient Methods

	Data Heterogeneity ( $\mathcal{D}_i$ different)	Compute Heterogeneity ( $\tau_i$ different)	Communication Heterogeneity ( $\theta_i$ different)	Smooth Nonconvex	Smooth Convex	Infinite / Finite Sum?	Supports Decentralized Setup?	Optimal Time Complexity?
Rennala SGD Tyurin & R (NeurIPS '23)	✗	✓	0	✓		Inf	✗	✓
Malenia SGD Tyurin & R (NeurIPS '23)	✓	✓	0	✓		Inf	✗	✓
Accelerated Rennala SGD Tyurin & R (NeurIPS '23)	✗	✓	0		✓	Inf	✗	✓
Shadowheart SGD Tyurin, Pozzi, Ilin & R '24	✗	✓	✓	✓		Inf	✗	✓
Freya PAGE Tyurin, Gruntkowska & R '24	✗	✓	0	✓		Finite	✗	✓ big data regime
Freya SGD Tyurin, Gruntkowska & R '24	✗	✓	0	✓		Finite	✗	✗
Fragile SGD Tyurin & R '24	✗	✓	✓	✓		Inf	✓	nearly
Amelie SGD Tyurin & R '24	✓	✓	✓	✓		Inf	✓	✓



The distributions  $\mathcal{D}_1, \dots, \mathcal{D}_n$  are allowed to be different

# Malenia SGD

## Method 6 Malenia SGD

```

1: Input: starting point  $x^0$ , stepsize  $\gamma$ , parameter  $S$ 
2: Run Method 7 in all workers
3: for  $k = 0, 1, \dots, K - 1$  do
4:   Init  $g_i^k = 0$  and  $B_i = 0$ 
5:   while  $\left( \frac{1}{n} \sum_{i=1}^n \frac{1}{B_i} \right)^{-1} < \frac{S}{n}$  do
6:     Wait for the next worker
7:     Receive gradient, iteration index, worker's index  $(g, k', i)$ 
8:     if  $k' = k$  then
9:        $g_i^k = g_i^k + g$ 
10:       $B_i = B_i + 1$ 
11:    end if
12:    Send  $(x^k, k)$  to the worker
13:  end while
14:   $g^k = \frac{1}{n} \sum_{i=1}^n \frac{1}{B_i} g_i^k$ 
15:   $x^{k+1} = x^k - \gamma g^k$ 
16: end for

```

Minibatch size

$$S = \max \left\{ \left\lceil \frac{\sigma^2}{\varepsilon} \right\rceil, n \right\}$$

## Method 7 Worker's Infinite Loop

```

1: Init  $g = 0$ ,  $k' = -1$ , and worker's index  $i$ 
2: while True do
3:   Send  $(g, k', i)$  to the server
4:   Receive  $(x^k, k)$  from the server
5:    $k' = k$ 
6:    $g = \hat{\nabla} f_i(x^k; \xi)$ ,  $\xi \sim \mathcal{D}$ 
7: end while

```

# (Nonconvex) Data Heterogeneous Regime

Method	Time Complexity
Minibatch SGD	$\tau_n \left( \frac{L\Delta}{\varepsilon} + \frac{\sigma^2 L\Delta}{n\varepsilon^2} \right)$
Malenia SGD (Theorem A.4)	$\tau_n \frac{L\Delta}{\varepsilon} + \left( \frac{1}{n} \sum_{i=1}^n \tau_i \right) \frac{\sigma^2 L\Delta}{n\varepsilon^2}$
Lower Bound (Theorem A.2)	$\tau_n \frac{L\Delta}{\varepsilon} + \left( \frac{1}{n} \sum_{i=1}^n \tau_i \right) \frac{\sigma^2 L\Delta}{n\varepsilon^2}$

# **Extension 2**

## **Handling the Convex Regime**

### **(Accelerated Rennala SGD)**

# Accelerated Rennala SGD: Setup

## Optimal Parallel Stochastic Gradient Methods

	Data Heterogeneity ( $\mathcal{D}_i$ different)	Compute Heterogeneity ( $\tau_i$ different)	Communication Heterogeneity ( $\theta_i$ different)	Smooth Nonconvex	Smooth Convex	Infinite / Finite Sum?	Supports Decentralized Setup?	Optimal Time Complexity?
<b>Rennala SGD</b> Tyurin & R (NeurIPS '23)	✗	✓	0	✓		Inf	✗	✓
<b>Malenia SGD</b> Tyurin & R (NeurIPS '23)	✓	✓	0	✓		Inf	✗	✓
<b>Accelerated Rennala SGD</b> Tyurin & R (NeurIPS '23)	✗	✓	0		✓	Inf	✗	✓
<b>Shadowheart SGD</b> Tyurin, Pozzi, Ilin & R '24	✗	✓	✓	✓		Inf	✗	✓
<b>Freya PAGE</b> Tyurin, Grunkowska & R '24	✗	✓	0	✓		Finite	✗	✓ big data regime
<b>Freya SGD</b> Tyurin, Grunkowska & R '24	✗	✓	0	✓		Finite	✗	✗
<b>Fragile SGD</b> Tyurin & R '24	✗	✓	✓	✓		Inf	✓	nearly
<b>Amelie SGD</b> Tyurin & R '24	✓	✓	✓	✓		Inf	✓	✓

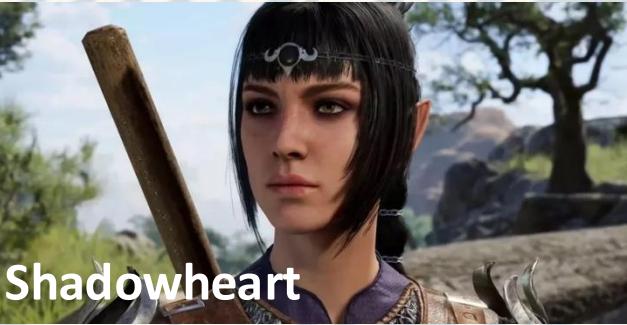


# Convex (Data Homogeneous) Regime

Method	Time Complexity
Minibatch SGD	$\tau_n \left( \min \left\{ \frac{\sqrt{L}R}{\sqrt{\varepsilon}}, \frac{M^2 R^2}{\varepsilon^2} \right\} + \frac{\sigma^2 R^2}{n \varepsilon^2} \right)$
Asynchronous SGD (Mishchenko et al., 2022)	$\left( \frac{1}{n} \sum_{i=1}^n \frac{1}{\tau_i} \right)^{-1} \left( \frac{LR^2}{\varepsilon} + \frac{\sigma^2 R^2}{n \varepsilon^2} \right)$
(Accelerated) Rennala SGD (Theorems B.9 and B.11)	$\min_{m \in [n]} \left[ \left( \frac{1}{m} \sum_{i=1}^m \frac{1}{\tau_i} \right)^{-1} \left( \min \left\{ \frac{\sqrt{L}R}{\sqrt{\varepsilon}}, \frac{M^2 R^2}{\varepsilon^2} \right\} + \frac{\sigma^2 R^2}{m \varepsilon^2} \right) \right]$
Lower Bound (Theorem B.4)	$\min_{m \in [n]} \left[ \left( \frac{1}{m} \sum_{i=1}^m \frac{1}{\tau_i} \right)^{-1} \left( \min \left\{ \frac{\sqrt{L}R}{\sqrt{\varepsilon}}, \frac{M^2 R^2}{\varepsilon^2} \right\} + \frac{\sigma^2 R^2}{m \varepsilon^2} \right) \right]$
Lower Bound (Section M) (Woodworth et al., 2018)	$\tau_1 \min \left\{ \frac{\sqrt{L}R}{\sqrt{\varepsilon}}, \frac{M^2 R^2}{\varepsilon^2} \right\} + \left( \frac{1}{n} \sum_{i=1}^n \frac{1}{\tau_i} \right)^{-1} \frac{\sigma^2 R^2}{n \varepsilon^2}$

$\nabla f$  is  $L$ -Lipschitz,  $f$  is  $M$ -Lipschitz, and  $\|x^0 - x^\star\| \leq R$

# **Further Extensions**



# Shadowheart SGD

## Optimal Parallel SGD under Compute Heterogeneity & Communication Heterogeneity

Alexander Tyurin, Marta Pozzi, Ivan Ilin and P.R.

**Shadowheart SGD: Distributed asynchronous SGD with optimal time complexity under arbitrary computation and communication heterogeneity**

arXiv:2402.04785, 2024



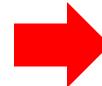
# Shadowheart SGD: Setup

$$\min_{x \in \mathbb{R}^d} \left\{ f(x) = \frac{1}{n} \sum_{i=1}^n f_i(x) \right\}$$

$$f_i(x) := \mathbb{E}_{\xi \sim \mathcal{D}_i} [f_i(x, \xi)]$$

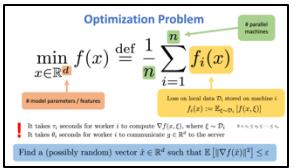
## Optimal Parallel Stochastic Gradient Methods

	Data Heterogeneity ( $\mathcal{D}_i$ different)	Compute Heterogeneity ( $\tau_i$ different)	Communication Heterogeneity ( $\theta_i$ different)	Smooth Nonconvex	Smooth Convex	Infinite / Finite Sum?	Supports Decentralized Setup?	Optimal Time Complexity?
Rennala SGD Tyurin & R (NeurIPS '23)	✗	✓	0	✓		Inf	✗	✓
Malenia SGD Tyurin & R (NeurIPS '23)	✓	✓	0	✓		Inf	✗	✓
Accelerated Rennala SGD Tyurin & R (NeurIPS '23)	✗	✓	0		✓	Inf	✗	✓
<b>Shadowheart SGD</b> Tyurin, Pozzi, Ilin & R '24	✗	✓	✓	✓		Inf	✗	✓
Freya PAGE Tyurin, Gruntkowska & R '24	✗	✓	0	✓		Finite	✗	✓ big data regime
Freya SGD Tyurin, Gruntkowska & R '24	✗	✓	0	✓		Finite	✗	✗
Fragile SGD Tyurin & R '24	✗	✓	✓	✓		Inf	✓	nearly
Amelie SGD Tyurin & R '24	✓	✓	✓	✓		Inf	✓	✓



$\mathcal{D}_1 = \dots = \mathcal{D}_n$

Communication costs  $\theta_1, \dots, \theta_n$  are nonzero (and possibly different)



# Shadowheart SGD

Unbiased compressor:

$$\mathbb{E} [\mathcal{C}_{ij}(g)] = g \quad \& \quad \mathbb{E} [\|\mathcal{C}_{ij}(g) - g\|^2] \leq \omega \|g\|^2 \quad \forall g \in \mathbb{R}^d$$

Aggregation weight associated with worker  $i$

$$w_i = \left( \omega b_i + \omega \frac{\sigma^2}{\varepsilon} + m_i \frac{\sigma^2}{\varepsilon} \right)^{-1}$$

$$x^{k+1} = x^k - \gamma \cdot \frac{\sum_{i=1}^n w_i}{\sum_{i=1}^n w_i m_i b_i} \mathcal{C}_{ij} \left( \sum_{l=1}^n \nabla f(x^k, \xi_{il}^k) \right)$$

$$\gamma = \frac{1}{2L}$$

Equilibrium time:  $t^* : \left( \omega, \frac{\sigma^2}{\varepsilon}, (\tau_i)_{i=1}^n, (\theta_i)_{i=1}^n \right) \mapsto \mathbb{R}_+$

# of compressed batches sent by worker  $i$  to the server

$$m_i = \left\lfloor \frac{t^*}{\theta_i} \right\rfloor$$

Batch size to compress by worker  $i$

$$b_i = \left\lfloor \frac{t^*}{\tau_i} \right\rfloor$$

**Table 1: Time Complexities of Centralized Distributed Algorithms.** Assume that it takes at most  $h_i$  seconds to worker  $i$  to calculate a stochastic gradient and  $\dot{\tau}_i$  seconds to send *one coordinate/float* to server. Abbreviations:  $L$  = smoothness constant,  $\varepsilon$  = error tolerance,  $\Delta = f(x^0) - f^*$ ,  $n$  = # of workers,  $d$  = dimension of the problem. We take the RandK compressor with  $K = 1$  (Def. C.1) (as an example) in QSGD and Shadowheart SGD. Due to Property 5.2, the choice  $K = 1$  is optimal for Shadowheart SGD up to a constant factor.

Method	Time Complexity	$\max\{h_n, \dot{\tau}_n\} \rightarrow \infty$ , $\max\{h_i, \dot{\tau}_i\} < \infty \forall i < n$ (the last worker is slow)	Time Complexities in Some Regimes			Numerical Comparison <sup>(b)</sup> $\sigma^2/\varepsilon =$
			$h_i = h, \dot{\tau}_i = \dot{\tau} \forall i \in [n]$ (equal performance)	$\max\{h, d\dot{\tau}, \frac{d\dot{\tau}\sigma^2}{n\varepsilon}, \frac{h\sigma^2}{n\varepsilon}\} \frac{L\Delta}{\varepsilon}$ (worse, e.g., when $\dot{\tau}, d$ or $n$ large)	$\times 10^3 \quad \times 10^3 \quad \times 10^4$	
Minibatch SGD (see (3))	$\max_{i \in [n]} \max\{h_i, d\dot{\tau}_i\} \left( \frac{L\Delta}{\varepsilon} + \frac{\sigma^2 L\Delta}{n\varepsilon^2} \right)$	$\infty$ (non-robust)	$\max\{h, d\dot{\tau}, \frac{d\dot{\tau}\sigma^2}{n\varepsilon}, \frac{h\sigma^2}{n\varepsilon}\} \frac{L\Delta}{\varepsilon}$ (worse, e.g., when $\dot{\tau}, d$ or $n$ large)	$\times 10^3 \quad \times 10^3 \quad \times 10^4$		
QSGD (see (7)) (Alistarh et al., 2017) (Khaled & Richtárik, 2020)	$\max_{i \in [n]} \max\{h_i, \dot{\tau}_i\} \left( \left( \frac{d}{n} + 1 \right) \frac{L\Delta}{\varepsilon} + \frac{d\sigma^2 L\Delta}{n\varepsilon^2} \right)$	$\infty$ (non-robust)	$\geq \frac{dh\sigma^2}{n\varepsilon} \frac{L\Delta}{\varepsilon}$ (worse, e.g., when $\varepsilon$ small)	$\times 3 \quad \times 10^2 \quad \times 10^4$		
Rennala SGD (Tyurin & Richtárik, 2023c), Asynchronous SGD (e.g., (Mishchenko et al., 2022))	$\geq \min_{j \in [n]} \max \left\{ h_{\bar{\pi}_j}, d\dot{\tau}_{\bar{\pi}_j}, \frac{\sigma^2}{\varepsilon} \left( \sum_{i=1}^j \frac{1}{h_{\bar{\pi}_i}} \right)^{-1} \right\} \frac{L\Delta}{\varepsilon}$ <sup>(a)</sup>	$< \infty$ (robust)	$\geq \max \left\{ h, d\dot{\tau}, \frac{h\sigma^2}{n\varepsilon} \right\} \frac{L\Delta}{\varepsilon}$ (worse, e.g., when $\dot{\tau}, d$ or $n$ large)	$\times 10^2 \quad \times 10 \quad \times 1.5$		
Shadowheart SGD (see (9) and Alg. 1) (Corollary 4.4)	$t^*(d - 1, \sigma^2/\varepsilon, [h_i, \dot{\tau}_i]_1^n) \frac{L\Delta}{\varepsilon}$ <sup>(c)</sup>	$< \infty$ (robust)	$\max \left\{ h, \dot{\tau}, \frac{d\dot{\tau}}{n}, \sqrt{\frac{d\dot{\tau}h\sigma^2}{n\varepsilon}}, \frac{h\sigma^2}{n\varepsilon} \right\} \frac{L\Delta}{\varepsilon}$	$\times 1 \quad \times 1 \quad \times 1$		

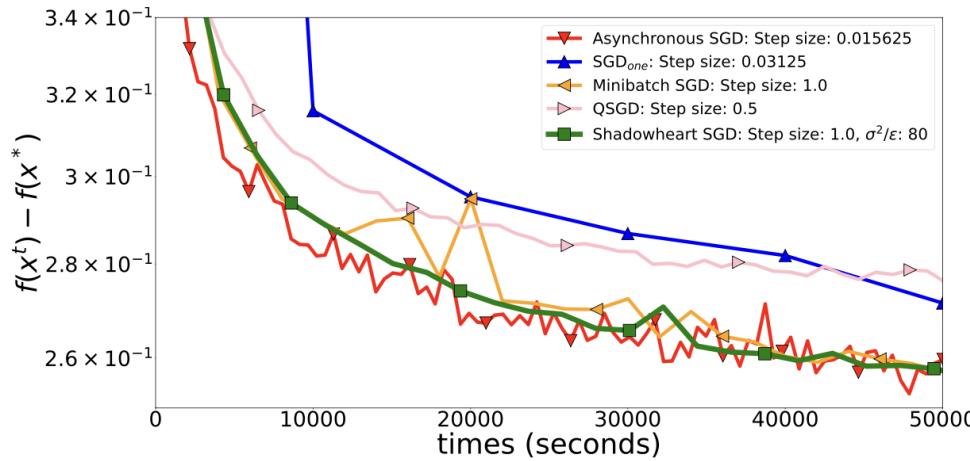
The time complexity of Shadowheart SGD is not worse than the time complexity of the competing centralized methods (see Sec. 6), and is *strictly* better in many regimes.

We show that (12) is the *optimal time complexity* in the family of centralized methods with compression (see Sec. 7).

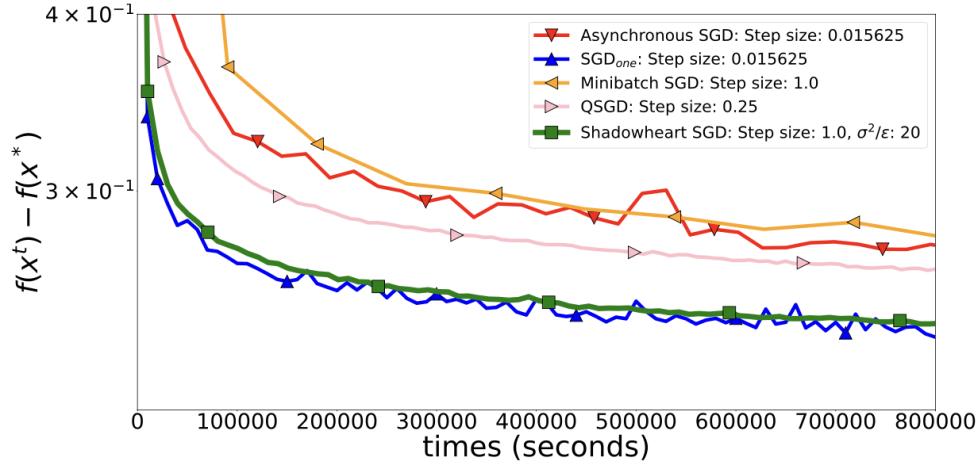
<sup>(a)</sup> Upper bound time complexities are not derived for Rennala SGD and Asynchronous SGD. However, we can derive the lower bound using Theorem N.5 with  $\omega = 0$ . One should take  $d\dot{\tau}_i$  instead of  $\tau_i$  when apply Theorem N.5 because these methods send  $d$  coordinates.  $\bar{\pi}$  is a permutation that sorts  $\max\{h_i, d\dot{\tau}_i\} : \max\{h_{\bar{\pi}_1}, d\dot{\tau}_{\bar{\pi}_1}\} \leq \dots \leq \max\{h_{\bar{\pi}_n}, d\dot{\tau}_{\bar{\pi}_n}\}$

<sup>(b)</sup> We numerically compute time complexities for  $d = 10^6$ ,  $n = 10^3$ ,  $h_i \sim U(0.1, 1)$ ,  $\dot{\tau}_i \sim U(0.1, 1)$  (uniform i.i.d.), and three noise regimes  $\sigma^2/\varepsilon \in \{1, 10^3, 10^6\}$ . We report the factors by which the time complexities of the competing methods are worse compared to the time complexity of our method Shadowheart SGD. So, for example, Minibatch SGD, QSGD and Asynchronous SGD can be worse by the factors  $\times 10^4$ ,  $\times 10^4$ , and  $\times 10^2$ , respectively.

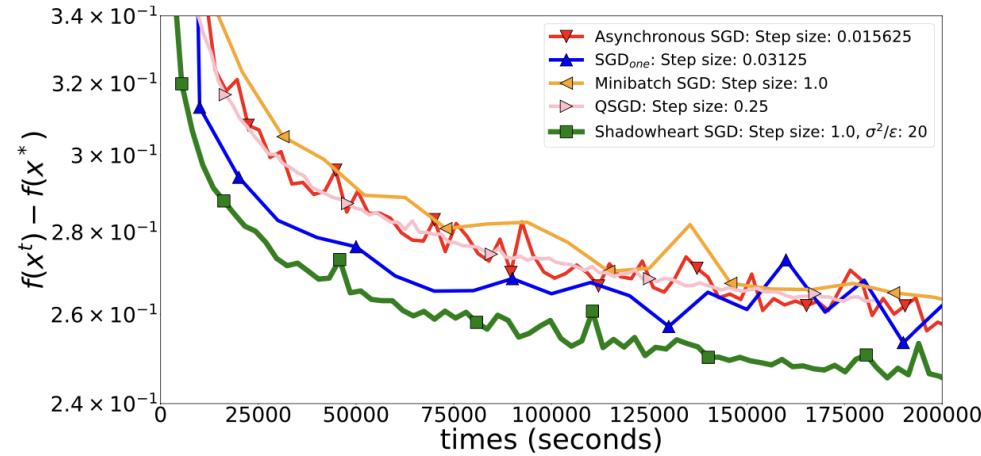
<sup>(c)</sup> The mapping  $t^*$  is defined in Def. 4.2.



**Fast communication:**  $\dot{\theta}_i = \frac{\sqrt{i}}{d}$



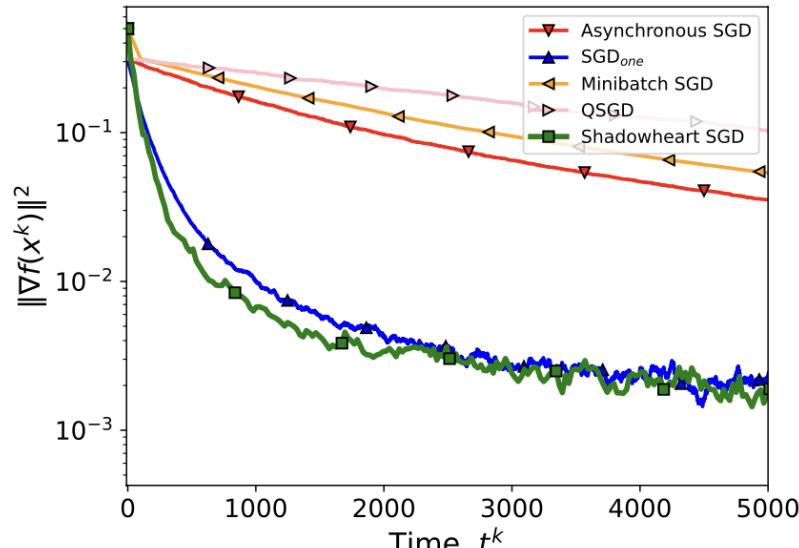
**Slow communication:**  $\dot{\theta}_i = \frac{\sqrt{i}}{d^{1/2}}$



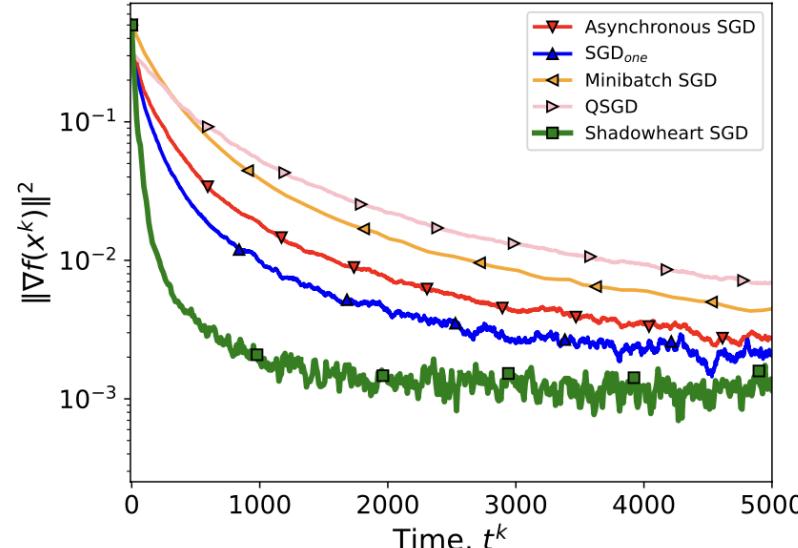
**Medium-speed communication:**  $\dot{\theta}_i = \frac{\sqrt{i}}{d^{3/4}}$

Computation times:  $\tau_i = \sqrt{i}$  for all machines  $i = 1, \dots, n$

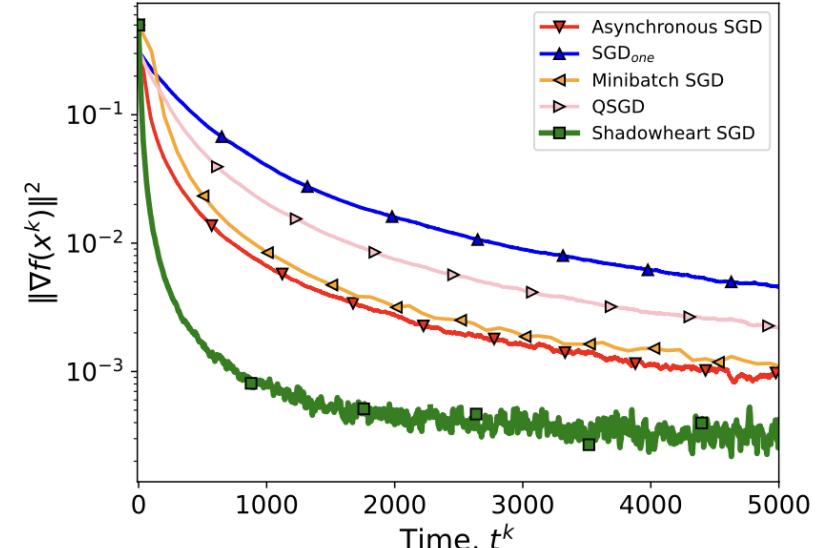
# Shadowheart SGD: Adding More Workers...



(a)  $n = 10$



(b)  $n = 10^2$



(c)  $n = 10^3$

$\tau_i^k, \dot{\theta}_i^k \sim \text{Uniform}(0.1, 1)$  for all  $i \in \{1, \dots, n\}$  and  $k \geq 0$

**Freya PAGE: First Optimal Time Complexity for Large-Scale Nonconvex Finite-Sum Optimization with Heterogeneous Asynchronous Computations**

Alexander Tyurin  
KAUST  
Kaja Gruntkowska  
KAUST  
Peter Richter  
KAUST

Abstract

In practical distributed systems, workers are typically not homogeneous, and due to differences in hardware configurations and network conditions, can have highly varying computation times. This motivates the need for a new parallel optimization framework that can handle asynchronous updates and introduce a new parallel regime. Freya PAGE, designed to handle asynchronous updates, provides the first optimal time complexity guarantee for large-scale nonconvex finite-sum optimization problems. Freya PAGE offers significantly improved time complexity guarantees compared to existing parallel SGD variants, such as P-SGD and P-SAGA, while maintaining the same convergence rate. The algorithm relies on novel generic assumptions that are more natural than those used by existing methods, such as P-SGD and PAGE, while requiring weaker assumptions. The algorithm relies on novel generic assumptions that are more natural than those used by existing methods, such as P-SGD and PAGE, while requiring weaker assumptions.

The algorithm is designed to work in two regimes: the synchronous regime, where all workers have the same computation time, and the asynchronous regime, i.e., when  $\gamma \neq 1$ , where  $\gamma$  is a worker, and  $n$  is the number of data samples.

1 Introduction

In real world distributed systems and for large-scale machine learning tasks, it is common to encounter device heterogeneity and variations in processing times among different computational nodes. This heterogeneity can arise from differences in hardware configurations, network conditions, and other factors, resulting in different computational capabilities and speeds across devices. Such heterogeneity poses significant challenges for distributed optimization algorithms.

Due to the above reasons, we aim to address the challenge posed by device heterogeneity in the context of solving finite-sum nonconvex optimization problems of the form

$$\min_{\theta} \left\{ f(\theta) + \frac{\lambda}{2} \|\theta\|^2 \right\},$$

where  $f: \mathbb{R}^n \rightarrow \mathbb{R}$  can be viewed as the loss of a machine learning model (e.g. the  $\ell_2$  example in a linear regression problem),  $\lambda > 0$  is a regularization parameter, and  $\theta \in \mathbb{R}^n$  is a parameter point such that  $\mathbb{E}[f(\theta)] \leq c$ . We focus on the heterogeneous distributed setup

- \* each worker has access to stochastic gradients  $\nabla f_i(x)$  ( $i \in [m]$ );
- \* worker  $i$  calculates  $\nabla f_i(x)$  in less or equal to  $\tau_i \in [0, \infty)$  seconds for all  $x \in [n], i \in [m]$ .

Preprint. Under review.

Shadowheart



Freya

# Freya PAGE

## Optimal Parallel SGD for Large-Scale Finite-Sum Problems



Alexander Tyurin, Kaja Gruntkowska, and P.R.

**Freya PAGE: First optimal time complexity for large-scale nonconvex finite-sum optimization with heterogeneous asynchronous computations**

arXiv:2405.1554, 2024

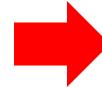
# Freya PAGE: Setup

$$\min_{x \in \mathbb{R}^d} \left\{ f(x) = \frac{1}{n} \sum_{i=1}^n f_i(x) \right\}$$

$$f_i(x) := \mathbb{E}_{\xi \sim \mathcal{D}_i} [f_i(x, \xi)]$$

## Optimal Parallel Stochastic Gradient Methods

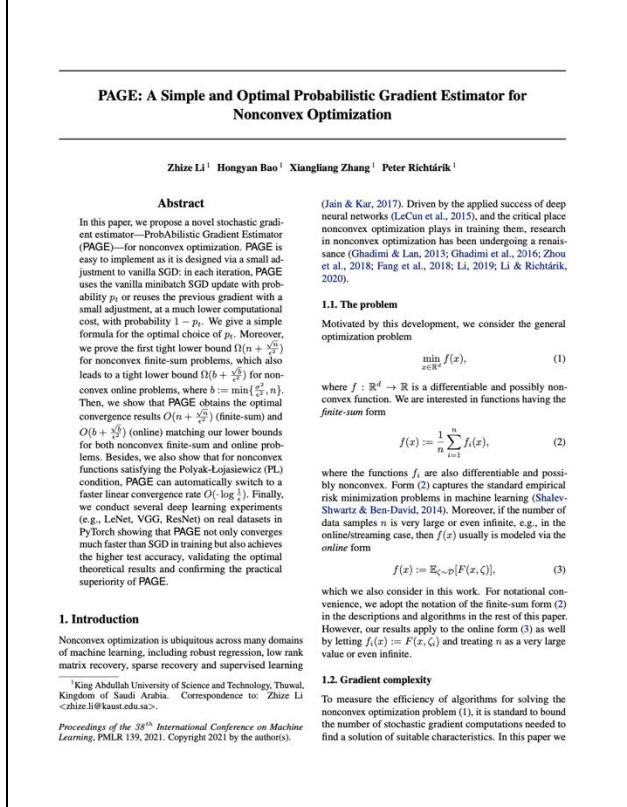
	Data Heterogeneity ( $\mathcal{D}_i$ different)	Compute Heterogeneity ( $\tau_i$ different)	Communication Heterogeneity ( $\theta_i$ different)	Smooth Nonconvex	Smooth Convex	Infinite / Finite Sum?	Supports Decentralized Setup?	Optimal Time Complexity?
Rennala SGD Tyurin & R (NeurIPS '23)	✗	✓	0	✓		Inf	✗	✓
Malenia SGD Tyurin & R (NeurIPS '23)	✓	✓	0	✓		Inf	✗	✓
Accelerated Rennala SGD Tyurin & R (NeurIPS '23)	✗	✓	0		✓	Inf	✗	✓
Shadowheart SGD Tyurin, Pozzi, Ilin & R '24	✗	✓	✓	✓		Inf	✗	✓
Freya PAGE Tyurin, Gruntkowska & R '24	✗	✓	0	✓		Finite	✗	✓ big data regime
Freya SGD Tyurin, Gruntkowska & R '24	✗	✓	0	✓		Finite	✗	✗
Fragile SGD Tyurin & R '24	✗	✓	✓	✓		Inf	✓	nearly
Amelie SGD Tyurin & R '24	✓	✓	✓	✓		Inf	✓	✓



$\mathcal{D}_1 = \dots = \mathcal{D}_n$

$\mathcal{D}_i =$  uniform distribution over  $m$  outcomes

# PAGE: Optimal Serial SGD for Finite-Sum Nonconvex Optimization



$$\min_{x \in \mathbb{R}^d} \left\{ f(x) = \frac{1}{n} \sum_{i=1}^n f_i(x) \right\}$$

$$f_i(x) := \mathbb{E}_{\xi \sim \mathcal{D}_i} [f_i(x, \xi)]$$

$$\mathcal{D}_1 = \dots = \mathcal{D}_n$$

$\mathcal{D}_i = \text{uniform distribution over } m \text{ outcomes}$

$$\min_{x \in \mathbb{R}^d} \left\{ f(x) = \frac{1}{m} \sum_{i=1}^m f_i(x) \right\}$$

(after butchering/redefining notation)



**Table 1:** Comparison of the *worst-case time complexity* guarantees of methods that work with asynchronous computations in the setup from Section 1 (up to smoothness constants). We assume that  $\tau_i \in [0, \infty]$  is the bound on the times required to calculate one stochastic gradient  $\nabla f_j$  by worker  $i$ ,  $\tau_1 \leq \dots \leq \tau_n$ , and  $m \geq n \log n$ .  
 Abbr:  $\delta^0 := f(x^0) - f^*$ ,  $m = \#$  of data samples,  $n = \#$  of workers,  $\varepsilon = \text{error tolerance}$ .

Method	Worst-Case Time Complexity	Comment
Hero GD (Soviet GD)	$\tau_1 m \frac{\delta^0}{\varepsilon} \quad (\tau_n \frac{m}{n} \frac{\delta^0}{\varepsilon})$	Suboptimal
Hero PAGE (Soviet PAGE) [Li et al., 2021]	$\tau_1 m + \tau_1 \frac{\delta^0}{\varepsilon} \sqrt{m} \quad (\tau_n \frac{m}{n} + \tau_n \frac{\delta^0}{\varepsilon} \frac{\sqrt{m}}{n})$	Suboptimal
SYNTHESIS [Liu et al., 2022]	—	Limitations: bounded gradient assumption, calculates the full gradients <sup>(a)</sup> , suboptimal. <sup>(b)</sup>
Asynchronous SGD [Koloskova et al., 2022] [Mishchenko et al., 2022]	$\frac{\delta^0}{\varepsilon} \left( \left( \sum_{i=1}^n \frac{1}{\tau_i} \right)^{-1} \left( \frac{\sigma^2}{\varepsilon} + n \right) \right)$	Limitations: $\sigma^2$ -bounded variance assumption, suboptimal when $\varepsilon$ is small.
Rennala SGD [Tyurin and Richtárik, 2023]	$\frac{\delta^0}{\varepsilon} \min_{j \in [n]} \left( \left( \sum_{i=1}^j \frac{1}{\tau_i} \right)^{-1} \left( \frac{\sigma^2}{\varepsilon} + j \right) \right)$	Limitations: $\sigma^2$ -bounded variance assumption, suboptimal when $\varepsilon$ is small.
Freya PAGE (Theorems 7 and 8)	$\begin{aligned} & \min_{j \in [n]} \left( \left( \sum_{i=1}^j \frac{1}{\tau_i} \right)^{-1} (m + j) \right) \\ & + \frac{\delta^0}{\varepsilon} \min_{j \in [n]} \left( \left( \sum_{i=1}^j \frac{1}{\tau_i} \right)^{-1} (\sqrt{m} + j) \right) \end{aligned}$	Optimal in the large-scale regime, i.e., $\sqrt{m} \geq n$ (see Section 5)
Lower bound (Theorem 10)	$\begin{aligned} & \min_{j \in [n]} \left( \left( \sum_{i=1}^j \frac{1}{\tau_i} \right)^{-1} (m + j) \right) \\ & + \frac{\delta^0}{\sqrt{m}\varepsilon} \min_{j \in [n]} \left( \left( \sum_{i=1}^j \frac{1}{\tau_i} \right)^{-1} (m + j) \right) \end{aligned}$	—

Freya PAGE has *universally* better guarantees than all previous methods: the dependence on  $\varepsilon$  is  $\mathcal{O}(1/\varepsilon)$  (unlike Rennala SGD and Asynchronous SGD), the dependence on  $\{\tau_i\}$  is harmonic-like and robust to slow workers (robust to  $\tau_n \rightarrow \infty$ ) (unlike Soviet PAGE and SYNTHESIS), the assumptions are weak, and the time complexity of Freya PAGE is optimal when  $\sqrt{m} \geq n$ .

<sup>(a)</sup> In Line 3 of their Algorithm 3, they calculate the full gradient, assuming that it can be done for free and not explaining how.

<sup>(b)</sup> Their convergence rates in Theorems 1 and 3 depend on a bound on the delays  $\Delta$ , which in turn depends on the performance of the slowest worker. Our method does not depend on the slowest worker if it is too slow (see Section 4.3), which is required for optimality.

<sup>(c)</sup> We prove better time complexity in Theorem 6, but this result requires the knowledge of  $\{\tau_i\}$  in advance, unlike Theorems 7 and 8.

---

**Algorithm 1** Freya PAGE

---

- 1: **Parameters:** starting point  $x^0 \in \mathbb{R}^d$ , learning rate  $\gamma > 0$ , minibatch size  $S \in \mathbb{N}$ , probability  $p \in (0, 1]$ , initialization  $g^0 = \nabla f(x^0)$  using `ComputeGradient( $x^0$ )` (Alg. 2)
- 2: **for**  $k = 0, 1, \dots, K - 1$  **do**
- 3:    $x^{k+1} = x^k - \gamma g^k$
- 4:   Sample  $c^k \sim \text{Bernoulli}(p)$
- 5:   **if**  $c^k = 1$  **then** (with probability  $p$ )
- 6:      $\nabla f(x^{k+1}) = \text{ComputeGradient}(x^{k+1})$  (Alg. 2)
- 7:      $g^{k+1} = \nabla f(x^{k+1})$
- 8:   **else** (with probability  $1 - p$ )
- 9:      $\frac{1}{S} \sum_{i \in \mathcal{S}^k} (\nabla f_i(x^{k+1}) - \nabla f_i(x^k)) = \text{ComputeBatchDifference}(S, x^{k+1}, x^k)$  (Alg. 3)
- 10:      $g^{k+1} = g^k + \frac{1}{S} \sum_{i \in \mathcal{S}^k} (\nabla f_i(x^{k+1}) - \nabla f_i(x^k))$
- 11:   **end if**
- 12: **end for**

(note):  $\mathcal{S}^k$  is a set of i.i.d. indices that are sampled from  $[m]$ , *uniformly with replacement*,  $|\mathcal{S}^k| = S$

---

---

**Algorithm 2** ComputeGradient( $x$ )

---

```
1: Input: point  $x \in \mathbb{R}^d$ 
2: Init  $g = 0 \in \mathbb{R}^d$ , set  $\mathcal{M} = \emptyset$ 
3: Broadcast  $x$  to all workers
4: For each worker  $i \in [n]$ , sample  $j$  from  $[m]$  uniformly and ask it to calculate  $\nabla f_j(x)$ 
5: while  $\mathcal{M} \neq [m]$  do
6:   Wait for  $\nabla f_p(x)$  from a worker
7:   if  $p \in [m] \setminus \mathcal{M}$  then
8:      $g \leftarrow g + \frac{1}{m} \nabla f_p(x)$ 
9:     Update  $\mathcal{M} \leftarrow \mathcal{M} \cup \{p\}$ 
10:  end if
11:  Sample  $j$  from  $[m] \setminus \mathcal{M}$  uniformly and ask this worker to calculate  $\nabla f_j(x)$ 
12: end while
13: Return  $g = \frac{1}{m} \sum_{i=1}^m \nabla f_i(x)$ 
```

---

---

**Algorithm 3** ComputeBatchDifference( $S, x, y$ )

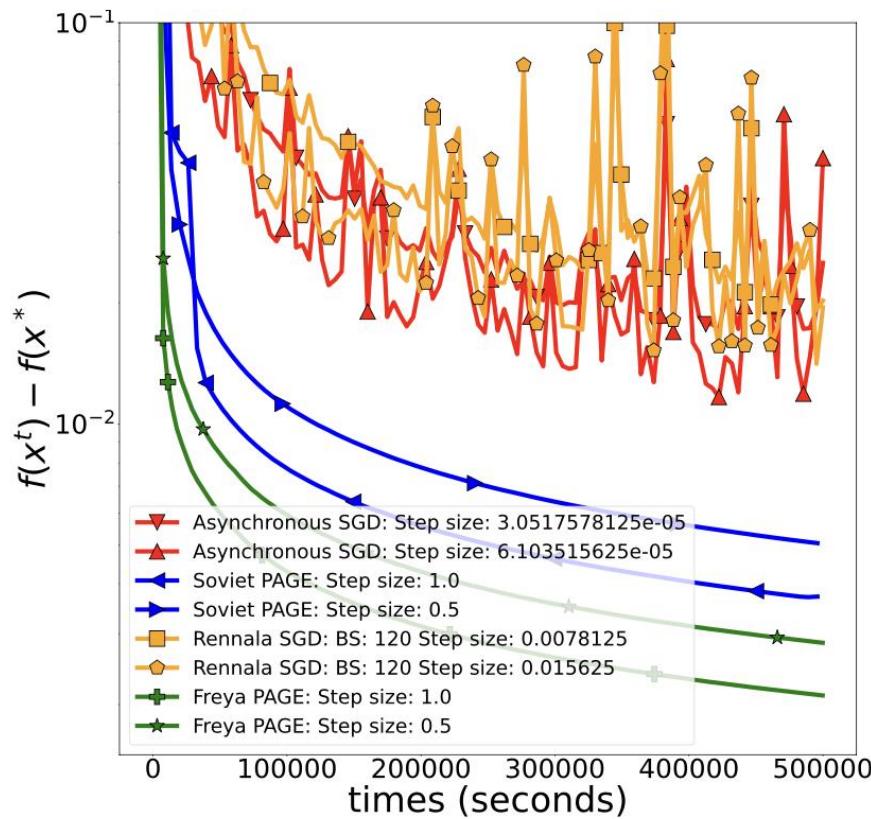
---

```
1: Input: batch size  $S \in \mathbb{N}$ , points  $x, y \in \mathbb{R}^d$ 
2: Init  $g = 0 \in \mathbb{R}^d$ 
3: Broadcast  $x, y$  to all workers
4: For each worker, sample  $j$  from  $[m]$  uniformly and ask it to calculate  $\nabla f_j(x) - \nabla f_j(y)$ 
5: for  $i = 1, 2, \dots, S$  do
6:   Wait for  $\nabla f_p(x) - \nabla f_p(y)$  from a worker
7:    $g \leftarrow g + \frac{1}{S} (\nabla f_p(x) - \nabla f_p(y))$ 
8:   Sample  $j$  from  $[m]$  uniformly and ask this worker to calculate  $\nabla f_j(x) - \nabla f_j(y)$ 
9: end for
10: Return  $g$ 
```

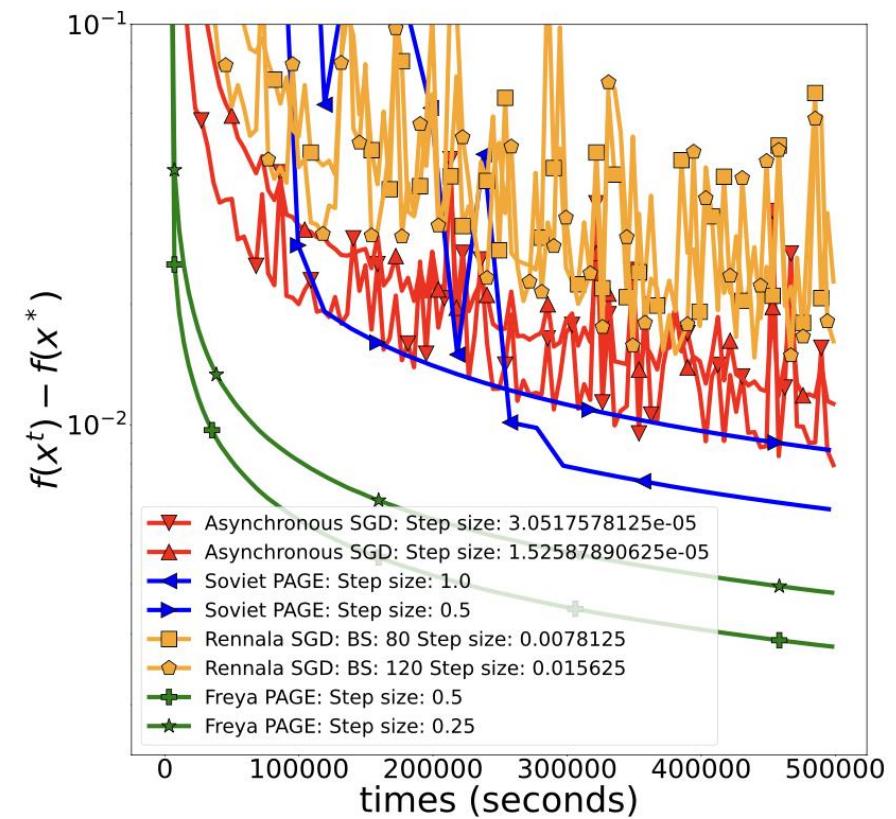
---

Notes: i) the workers can aggregate  $\nabla f_p$  locally, and the algorithm can call AllReduce once to collect all calculated gradients. ii) By splitting  $[m]$  into blocks, instead of one  $\nabla f_p$ , we can ask the workers to calculate the sum of one block in Alg. 2 (and use a similar idea in Alg. 3).

# Freya PAGE: Experiment 1



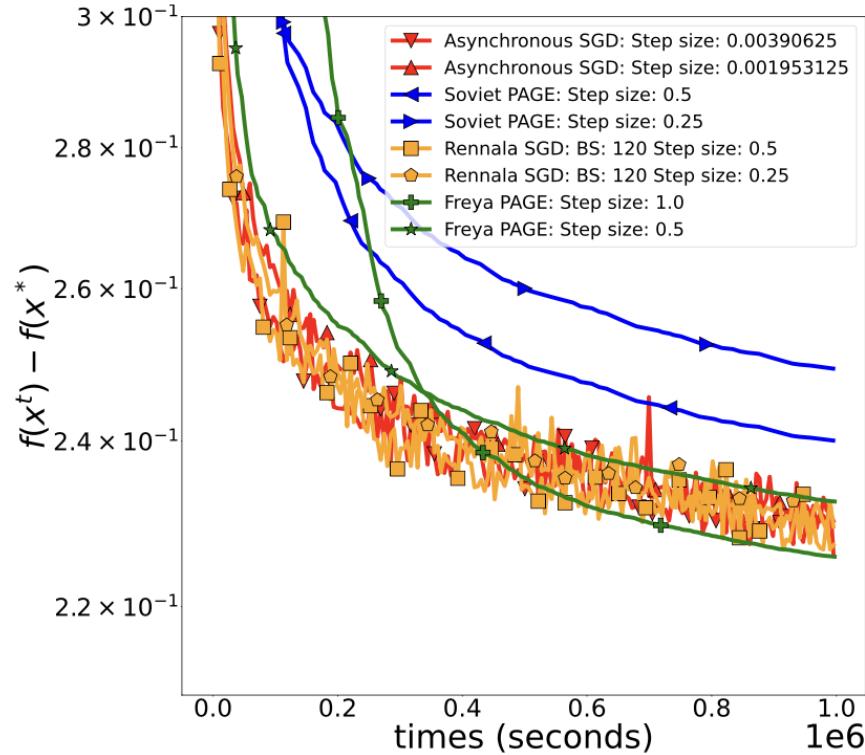
(a)  $n = 1000$



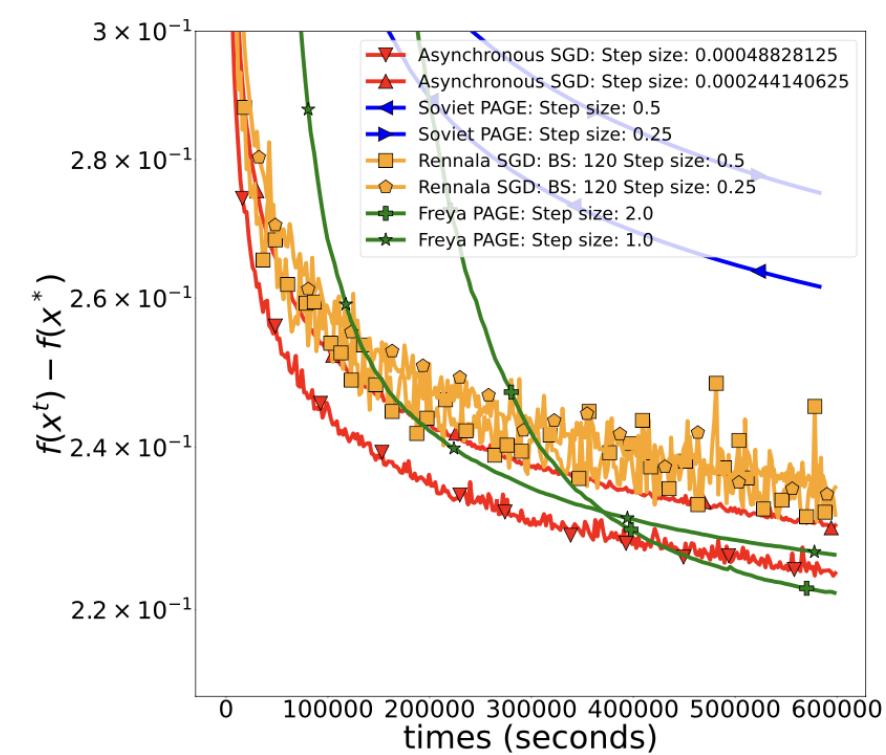
(b)  $n = 10000$

Figure 1: Experiments with nonconvex quadratic optimization tasks. We plot function suboptimality against elapsed time.

# Freya PAGE: Experiment 2



(a)  $n = 100$



(b)  $n = 10000$

Figure 2: Experiments with the logistic regression problem on the MNIST dataset.

# Freya PAGE: Experiment 2

Table 2: Mean and variance of algorithm accuracies on the MNIST test set during the final 100K seconds of the experiments from Figure 2b.

Method	Accuracy	Variance of Accuracy
Asynchronous SGD [Koloskova et al., 2022] [Mishchenko et al., 2022]	92.60	5.85e-07
Soviet PAGE [Li et al., 2021]	92.31	1.62e-07
Rennala SGD [Tyurin and Richtárik, 2023]	92.37	3.12e-06
<b>Freya PAGE</b>	<b>92.66</b>	<b>1.01e-07</b>



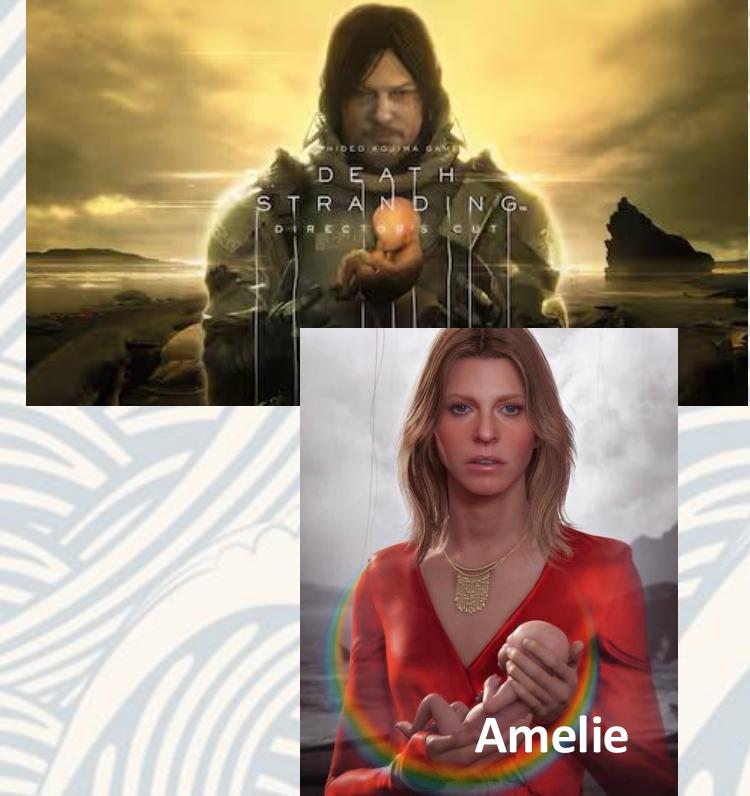
# Amelie SGD

## Optimal **Decentralized** SGD under Computation & Communication Heterogeneity



Alexander Tyurin and P.R.

**On the optimal time complexities in decentralized stochastic asynchronous optimization**  
arXiv:2405.16218, 2024



# Decentralized Setup: Amelie SGD

Method	The Worst-Case Time Complexity Guarantees	Comment
Minibatch SGD	$\frac{L\Delta}{\varepsilon} \max \left\{ \left(1 + \frac{\sigma^2}{n\varepsilon}\right) \max \left\{ \max_{i,j \in [n]} \tau_{i \rightarrow j}, \max_{i \in [n]} h_i \right\} \right\}$	suboptimal if $\sigma^2/\varepsilon$ is large
RelaySGD, Gradient Tracking <small>(Vogels et al., 2021) (Liu et al., 2024)</small>	$\geq \frac{\max_{i \in [n]} L_i \Delta}{\varepsilon} \frac{\sigma^2}{n\varepsilon} \max_{i \in [n]} h_i$	requires local $L_i$ -smooth. of $f_i$ , suboptimal if $\sigma^2/\varepsilon$ is large (even if $\max_{i \in [n]} L_i = L$ )
Asynchronous SGD <small>(Even et al., 2024)</small>	—	requires similarity of the functions $\{f_i\}$ , requires local $L_i$ -smooth. of $f_i$
Amelie SGD and Lower Bound <small>(Thm. 7 and Cor. 2)</small>	$\frac{L\Delta}{\varepsilon} \max \left\{ \max_{i,j \in [n]} \tau_{i \rightarrow j}, \max_{i \in [n]} h_i, \frac{\sigma^2}{n\varepsilon} \left( \frac{1}{n} \sum_{i=1}^n h_i \right) \right\}$	Optimal up to a constant factor



**The End  
(for real)**