

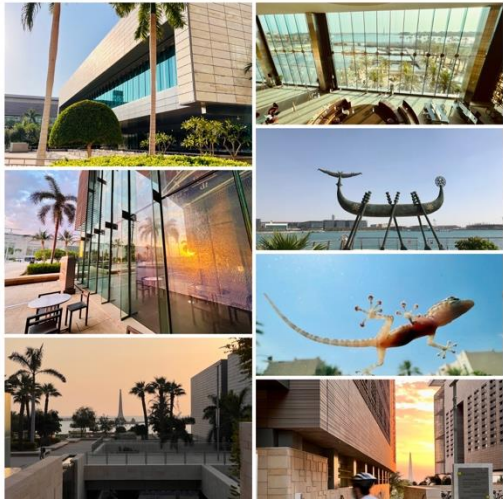
Federated Optimization: A Biased Tutorial

Peter Richtárik

King Abdullah University of Science and Technology
Kingdom of Saudi Arabia



Simons Institute, Berkeley
January 27, 2026



Plan for the Tutorial

Lecture 1
Overview

Lecture 2
Communication Compression

Lecture 3
Local Training

Lecture 4
Device Heterogeneity

The background is a traditional Chinese ink wash painting. It features a misty, layered mountain landscape with soft, hazy peaks. In the foreground, dark, silhouetted mountain ridges are visible. Delicate, dark brown branches with clusters of pink cherry blossoms and scattered petals frame the top and sides of the image. The overall style is elegant and serene, typical of classical Chinese art.

Lecture 1

Overview



Introduction

The Field is Too Broad: Can't Cover it All!

- **History & future of the field**
- **Mathematical formulations of federated optimization/learning**
 - **Empirical risk minimization**
 - Fine-tuning
 - Personalization/generalization
 - Agnostic FL
- **Computing architecture**
 - **Parameter-server architecture** (e.g., cross-silo / cross-device / ...)
 - Fully decentralized (connected network)
 - Cross-datacenter architecture
- Types of parallelism (**data**, pipeline, model, tensor, memory)
- **Algorithmic considerations & theory**
- **Practical systems & applications**
 - training classifiers, DNNs, LLMs, ...
 - industrial systems & achievements
- **Privacy** (e.g., encryption, diff. privacy, TEEs, SMPC)
- **Security** (e.g., guarding against Byzantine workers)
- **Federated X** (e.g., Fed analytics)

Distributed Optimization “In The Wild”

- Not necessarily focusing on prevalent industrial FL architectures (cross-silo / cross-device) only
- **Bias towards classical/new fundamental issues and techniques**
 - Understanding them in isolation
 - Does a combination of the techniques have an additive effect?

Mathematical Formulations of FL

- Minimize the average of finitely many functions
- Minimize the weighted average of finitely many functions
- Minimize the expectation of (possibly infinite #) functions
- Find the fixed point of the average of finitely many operators
- “Personalized formulations”
- Agnostic FL, ...

a bit



Mehryar Mohri, Gary Sivek, Ananda Theertha Suresh
Agnostic Federated Learning
arXiv 1902.00146

Learn the worst case mixture of local distributions rather than the uniform mixture (standard FL)

Data-Parallel Federated Training

(Minimize the average of finitely many functions)

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

model parameters / features

$\stackrel{\text{def}}{=}$

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

devices /
machines

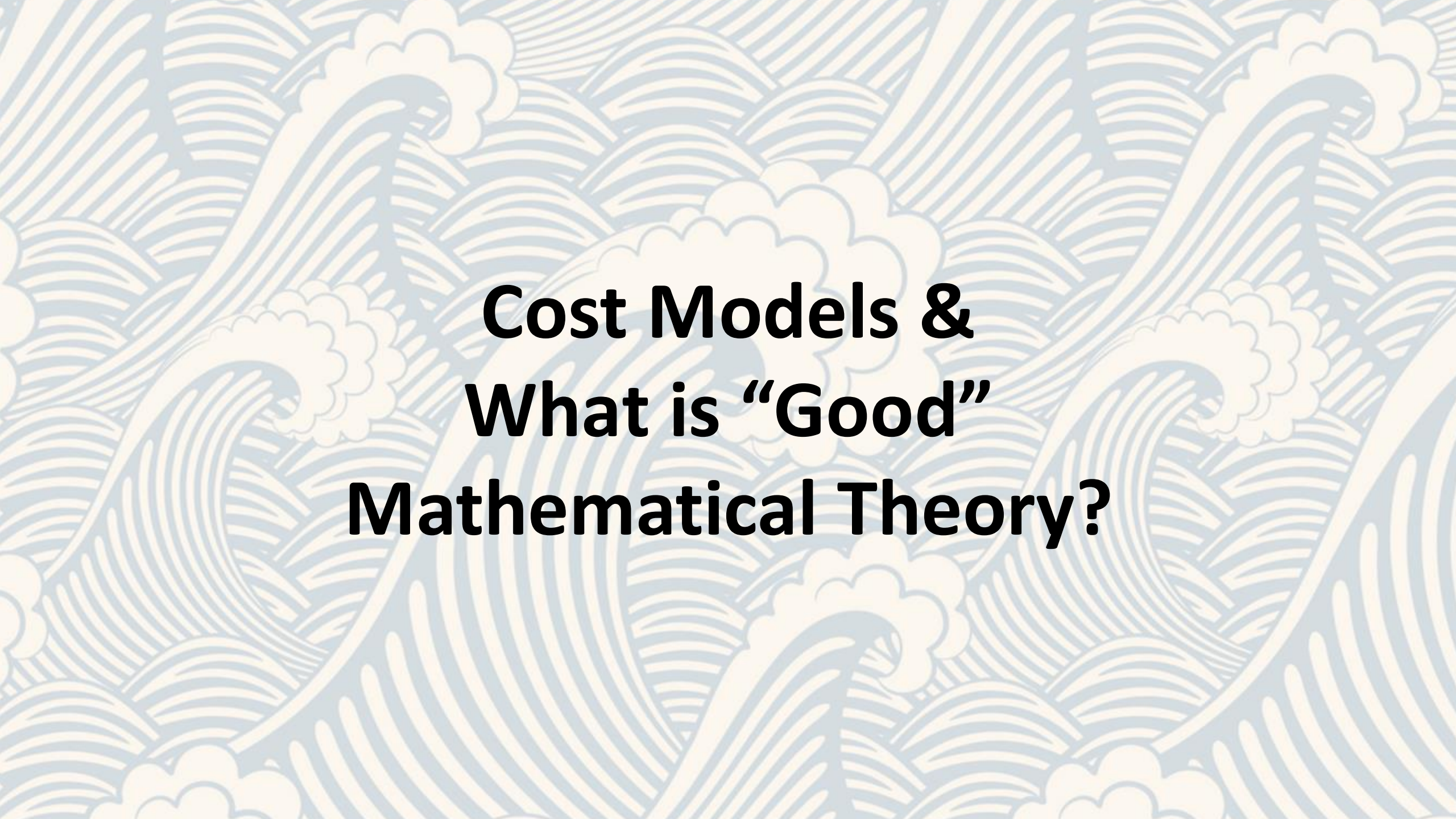
Loss on local data $\mathcal{D}_i$ stored on device i

$$f_i(x) = \mathbb{E}_{\xi \sim \mathcal{D}_i} f_{i,\xi}(x)$$

The datasets $\mathcal{D}_1, \dots, \mathcal{D}_n$ can be arbitrarily heterogeneous

Algorithmic Considerations & Theory

- Partial participation
 - Random
 - Greedy / Optimal
 - Structured (e.g., cyclic)
 - “Destiny”
- Stochastic approximation
- **Local training (e.g., Local GD, FedProx, ProxSkip, drift reduction, acceleration)**
- **Communication compression (e.g., quantization, sparsification, low-rank compression, error-feedback)**
- **Device heterogeneity (e.g., compute & communication cost variability, role of asynchronicity)**
- Data/statistical heterogeneity
- Server extrapolation
 - e.g., Nastya, FedExP, FedExProx
- Server aggregation
 - e.g., FedNova



Cost Models & What is “Good” Mathematical Theory?

Approaches to FL Research

- **Most papers (“99%”)**

- Are empirical and contain no (useful) theory
-  focus on **stuff that is useful in practice**
-  **may collapse / stop working / be unstable in new situations**
-  **difficult to operate a system composed of many heuristic components**
- Research process:
 - Start with a method that is theoretically understood (e.g., SGD, Adam, ...)
 - Tinker away = trial & error
 - trial = random hyperparameter search or following some intuitions (from exps or theory)

- **Almost all theoretical papers (“99% of the remaining 1%”)**

- Either bound
 - `#iters` or
 - `#rounds`
- This is a poor proxy for runtime - we need better cost models

Cost Model = A Model of the Total Training Time

- Classical example: `#iters`
 - OK as a proof of correctness of an algorithm
 - BAD as a proof of efficiency
 - Most papers focus on this model
- Complexity of modern compute architectures makes it hard to form accurate models
 - Hub-spokes
 - Hierarchical
 - Decentralized network
 - data/pipeline/tensor/memory parallelism
- So, we abstract the issues that are relatively minor, and focus on bottlenecks
- Alternative/better models:
 - `#rounds`
 - `#rounds` x [`A` x `#up_bits/round`]
 - `#rounds` x [`A` x `#up_bits/round` + `B` x `#down_bits/round`]
 - `#rounds` x [`A` x `#up_bits/round` + `B` x `#down_bits/round` + `C` x `compute_time/round`]
 - etc

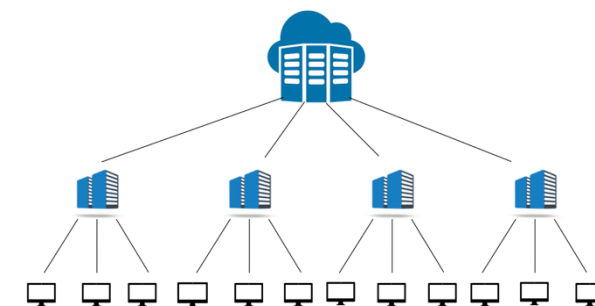


Figure 1: Server-hubs-clients FL architecture with 4 hubs and 12 clients.



Grigory Malinovsky, Kai Yi and P.R.

Variance reduced ProxSkip: algorithm, theory and application to federated learning

NeurIPS 2022

Construction of Useful Cost Models is Further Complicated by Heterogeneity

- **Data heterogeneity**

- Same data - each device is in principle able to train
- Samples from the same data distribution & #samples can be different
- Samples from different distributions & #samples can be different
- Arbitrarily different data

- **Compute heterogeneity** (devices have different compute power)

- Sensors, smart watches, smart phones, tablets, laptops, workstations, clusters,

data centers)

- **Communication heterogeneity**

- Infiniband, NVLink, 3.5 Gbps internet, 1Gbps internet, 100 Mbps, ..

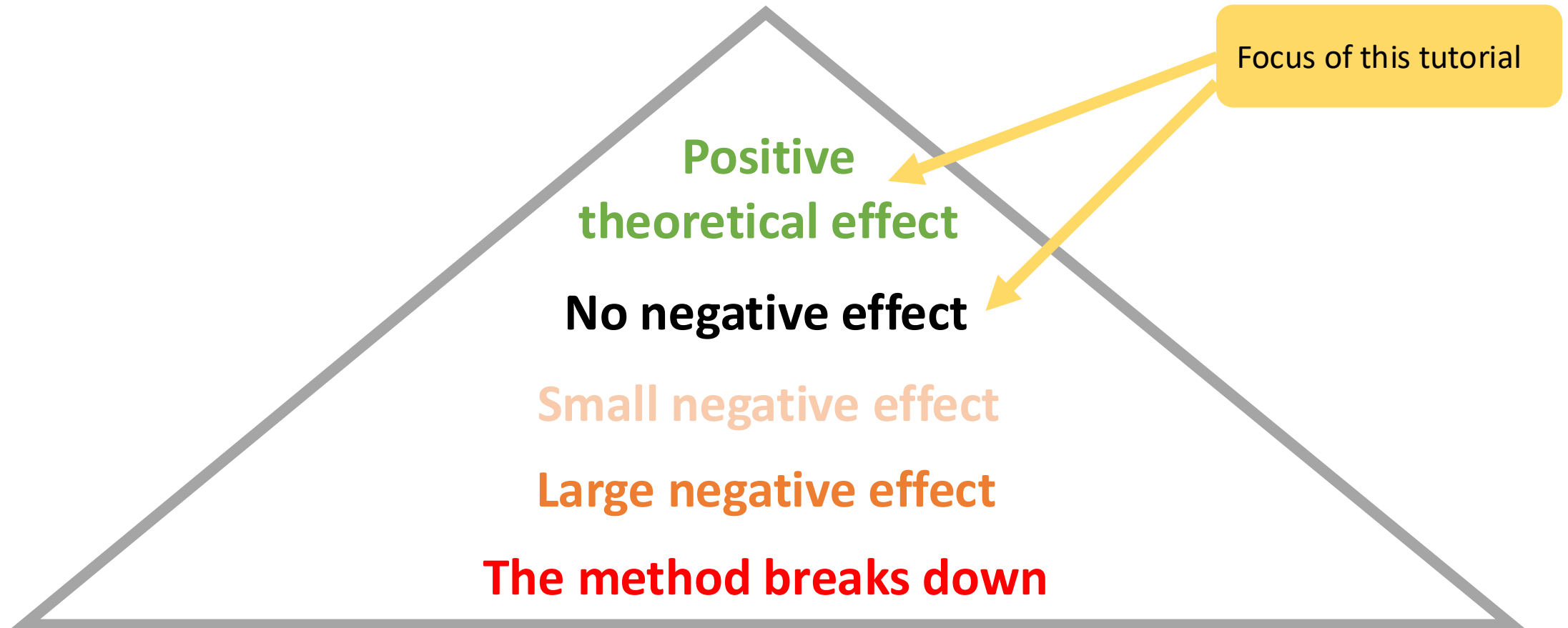
- **Time heterogeneity**

- Some characteristics can vary in time

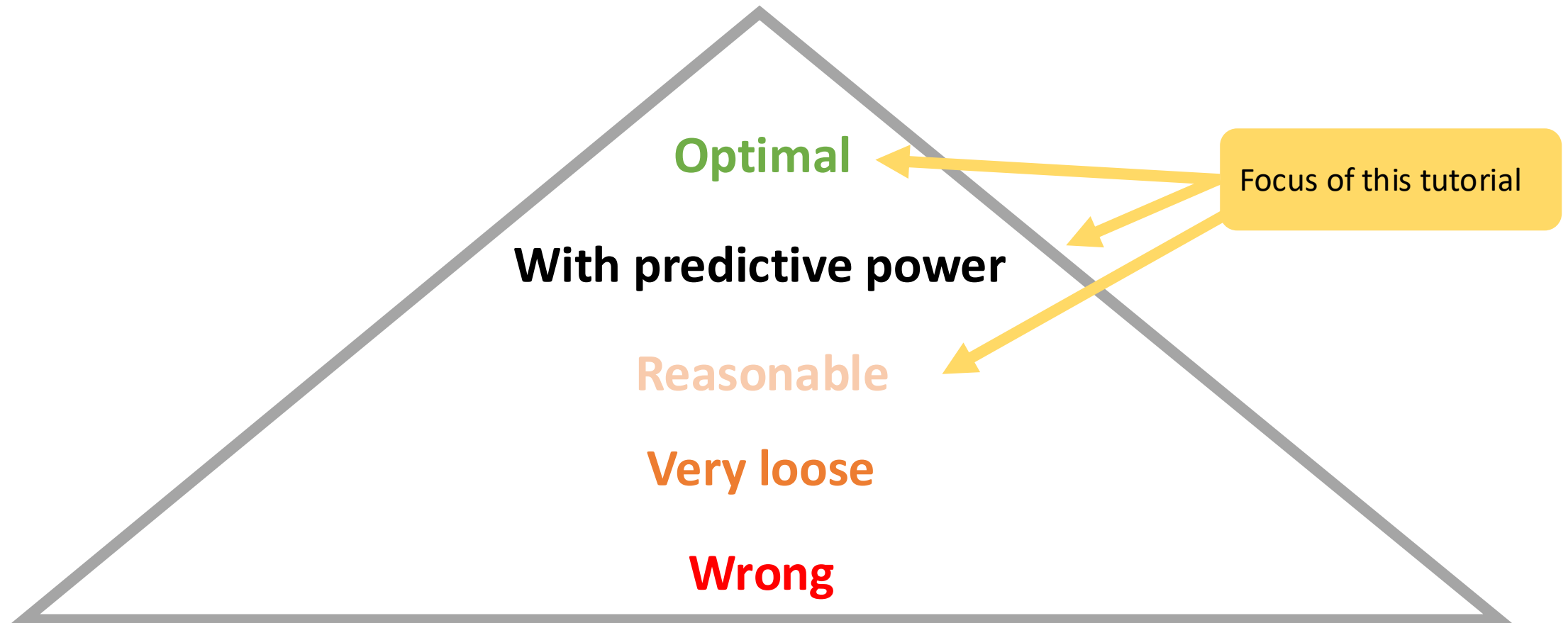
- **Memory heterogeneity**

- **Model heterogeneity**

Hierarchy of Theoretical Understanding of Algorithmic Tricks



Hierarchy of Theoretical Bounds





Selected Excluded Topics

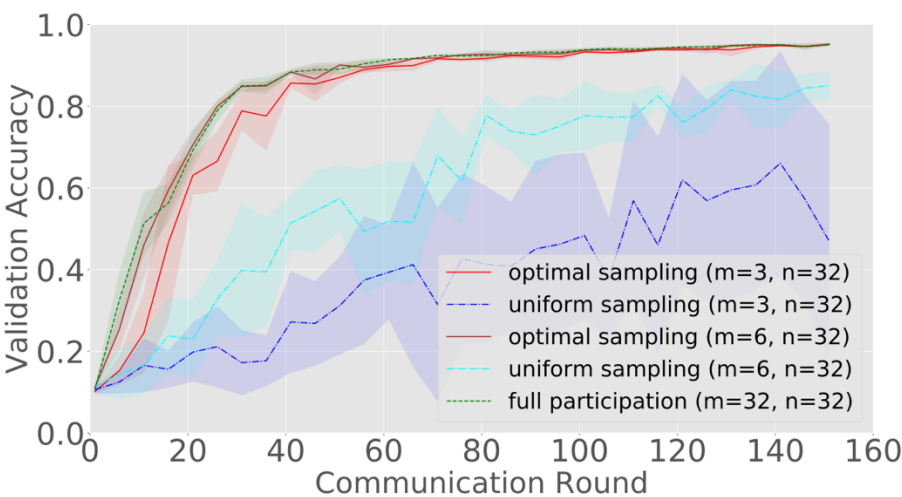
Client Sampling

10/2020

DSGD, FedAvg



Wenlin Chen, Samuel Horváth, P.R.
Optimal Client Sampling for Federated Learning
TMLR 2022



5/2022



Shiqiang Wang, Mingyue Ji
A Unified Analysis of Federated Learning with Arbitrary Client Participation
NeurIPS 2022

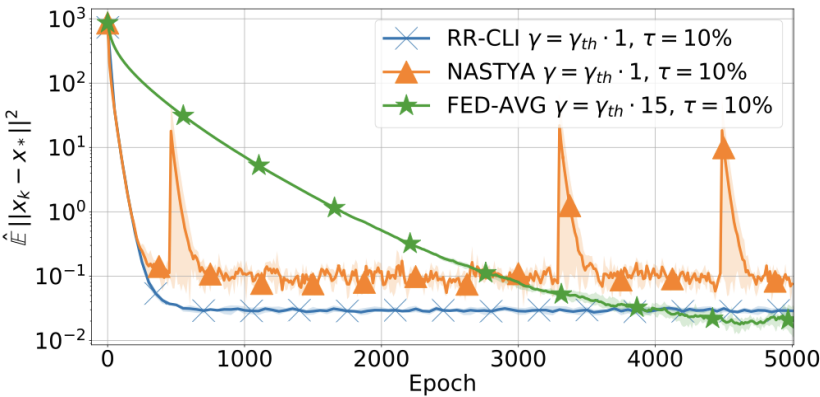
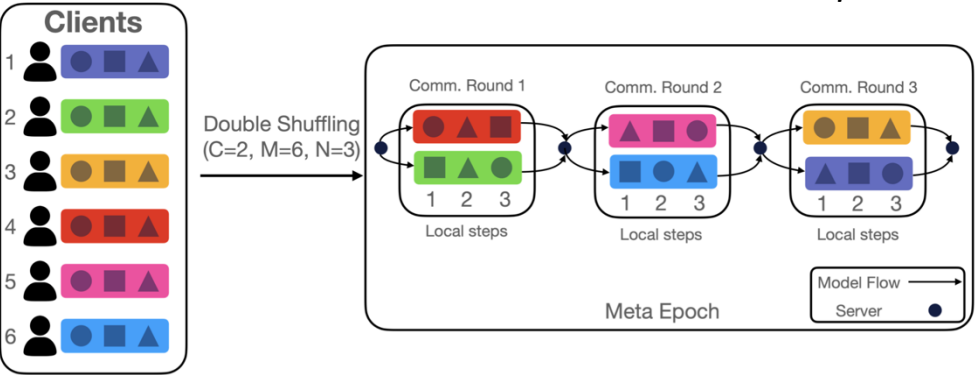
2/2023

RR-CLI



Grigory Malinovsky, Samuel Horváth, Konstantin Burlachenko and P.R.
Federated learning with regularized client participation
ICML Workshop 2023

Method	Participation	Convergence error upper bound	Remark
Yan et al. [35]	Unavailable $\leq E$ rounds	$\mathcal{O}\left(\frac{(1+\sigma^2)N^{1/4}\sqrt{E}}{\sqrt{ST}}\right)$	Add'l assumption: bounded gradient norm
Gu et al. [12]	Unavailable $\leq E$ rounds	$\mathcal{O}\left(\frac{(1+\sigma^2)\sqrt{E}}{\sqrt{NIT}}\right)$	Add'l assumptions: Hessian Lipschitz, a.s. bounded noise
Ours (this paper)	Regularized	$\mathcal{O}\left(\frac{\sigma}{\sqrt{SIT}}\right)$	Cor. 4.3, matching centralized SGD lower bound
	Ergodic	Approaches zero as $T \rightarrow \infty$	Prop. 4.4
	Mixing	$\mathcal{O}\left(\frac{1+\sigma^2}{\sqrt{SIT}}\right)$ i.E.; $\mathcal{O}\left(\frac{1+\sigma^2}{c\sqrt{SIT}}\right)$ w.p. $1-c$	Prop. 4.6 & Cor. 4.8, matching idealized bound
	Independent	$\mathcal{O}\left(\frac{1+\sigma^2}{\sqrt{SIT}}\right)$ i.E.; $\mathcal{O}\left(\frac{(1+\sigma^2)\ln(2/c)}{\sqrt{SIT}}\right)$ w.p. $1-c$	Prop. 4.7 & Cor. 4.8, matching idealized bound



Federated Fine-Tuning

10/2024

RAC-LoRA
Fed-RAC-LoRA



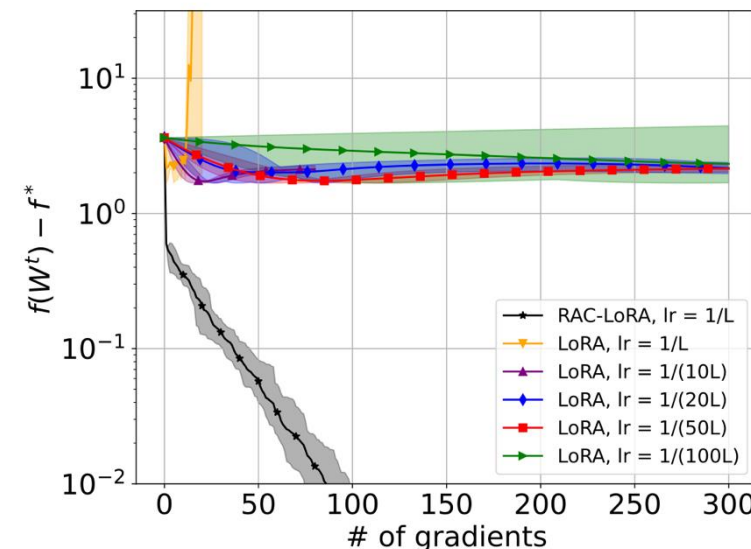
Grigory Malinovsky, Umberto Michieli, Hasan Abed Al Kader Hammoud, Taha Ceritli, Hayder Elesedy, Mete Ozay, and P.R.
Randomized asymmetric chain of LoRA: The first meaningful theoretical framework for low-rank adaptation
arXiv 2410.08305

- LORA and COLA can diverge! (simple 9-dim counterexample)
- ***Provides a fix***
- ***1st meaningful theory for LoRA-type methods***
- ***federated variant***

Numerical counterexample. We present a clear and illustrative example demonstrating that the LoRA and COLA methods may not converge to the solution of the optimization problem. To illustrate this, let us consider a quadratic function of the following form:

$$f(x) = x^T M x + b^T x, \quad (2)$$

where $x \in \mathbb{R}^d$ is a vector of parameters, $M \in \mathbb{R}^{d \times d}$ is a positive definite matrix, and $b \in \mathbb{R}^d$ is a vector corresponding to the linear term. In our numerical example, we consider $d = 9$, $M = \text{Diag}(10, 1, 1, 1, 1, 1, 1, 1, 1)$, and $b = (1, 1, 1, 1, 1, 1, 1, 1, 1)^T$. This function has a Lipschitz gradient (Assumption 3.1) with $L = 10$. We represent the vector $x \in \mathbb{R}^9$ as a matrix $W \in \mathbb{R}^{3 \times 3}$. In the LoRA adaptation, we use a rank $r = 1$ and set $\alpha = r$.



6/2025

RAVAN



Arian Raje, Baris Askin, Divyansh Jhunjunwala, Gauri Joshi
Ravan: Multi-Head Low-Rank Adaptation for Federated Fine-Tuning

*Adaptive multi-head
LoRA*

8/2025

Bernoulli-LoRA



Igor Sokolov, Abdurakhmon Sadiev, Yury Demidovich, Fawaz S Al-Qahtani, and P.R.
Bernoulli-LoRA: A theoretical framework for randomized low-rank adaptation
arXiv 2508.03820

*Making RAC-LoRA
symmetric + further
improvements*

Server/Outer Extrapolation/Solvers

1/2022

Nastya



Grigory Malinovsky, Konstantin Mishchenko, and P.R.
Server-side Stepsizes and Sampling Without Replacement Provably Help in Federated Optimization
DistributedML 2023

First proof of the benefit of server extrapolation (limited to random reshuffling as local solver)

1/2023

FedExP



Divyansh Jhunjunwala, Shiqiang Wang, and Gauri Joshi
FedExP: Speeding Up Federated Averaging via Extrapolation
ICLR 2023

An adaptive extrapolation technique inspired by convex feasibility + analysis

5/2024

FedExProx

FedExProx-GraDS/StoPS



Hanmin Li, Kirill Acharya, and P.R.
The Power of Extrapolation in Federated Learning
NeurIPS 2024

In-depth analysis of extrapolation benefits for FedProx (explained via gradient diversity and stochastic Polyak stepsizes)

10/2024

FedExProx



Hanmin Li and P.R.
On the Convergence of FedProx with Extrapolation and Inexact Prox
OPT 2024

9/2025



Ahmed Khaled, Satyen Kale, Arthur Douillard, Chi Jin, Rob Fergus, Manzil Zaheer
Understanding Outer Optimizers in Local SGD: Learning Rates, Momentum, and Acceleration
arXiv 2509.10439v2

First understanding of an accelerated outer optimizer (as in DiLoCo), the effect of **momentum**, first high prob analysis of Local SGD

Personalization

$$\min_{x_1, \dots, x_n} \left\{ \frac{1}{n} \sum_{i=1}^n f_i(x_i) + \frac{\lambda}{2n} \sum_{i=1}^n \left\| x_i - \frac{1}{n} \sum_j x_j \right\|^2 \right\}$$

2/2020

Loopless Local GD
(L2GD)



Filip Hanzely and P.R.

Federated learning of a mixture of global and local models

SpicyFL 2020: NeurIPS Workshop on Scalability, Privacy, and Security in Federated Learning

Expected # comm rounds = $\frac{2\lambda}{L+\lambda} \frac{L}{\mu} \log \frac{1}{\epsilon}$

10/2020



Filip Hanzely, Slavomír Hanzely, Samuel Horváth, P.R.

Lower Bounds and Optimal Algorithms for Personalized Federated Learning

NeurIPS 2020

Optimal methods for
the formulation above

2/2021



Filip Hanzely, Boxin Zhao, Mladen Kolar

Personalized Federated Learning: A Unified Framework and Universal Optimization Techniques

TMLR 2023

11/2021



Elnur Gasanov, Ahmed Khaled, Samuel Horváth and P.R.

FLIX: A simple and communication-efficient alternative to local methods in federated learning

AISTATS 2022

$$\min_{x, x_1, \dots, x_n} \frac{1}{n} \sum_{i=1}^n f_i(x, x_i)$$

4/2022



Krishna Pillutla, Kshitiz Malik, Abdelrahman Mohamed, Michael

Rabbat, Maziar Sanjabi, Lin Xiao

Federated Learning with Partial Model Personalization

ICML 2022

Partial personalization

5/2023



Konstantin Mishchenko, Rustem Islamov, Eduard Gorbunov, and Samuel Horváth

Partially Personalized Federated Learning: Breaking the Curse of Data Heterogeneity

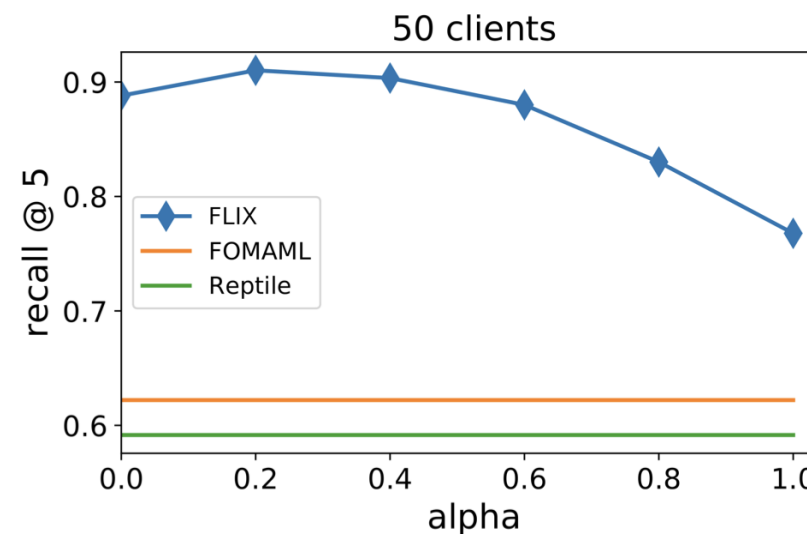
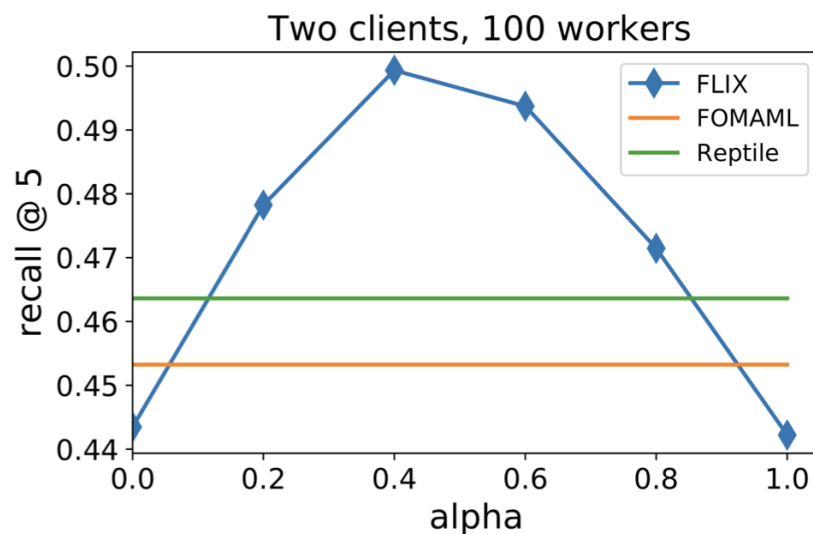
TMLR 2025

Personalization

Personalization level

$\alpha = 1$ recovers standard FL formulation

$$\min_x \frac{1}{n} \sum_{i=1}^n f_i(\alpha x + (1 - \alpha)x_i^*)$$



(a) 100 workers created out of two clients' data (b) 50 workers with distinct data distributions

11/2021



Elnur Gasanov, Ahmed Khaled, Samuel Horváth and P.R.

FLIX: A simple and communication-efficient alternative to local methods in federated learning

AISTATS 2022

Dealing with Byzantine Workers

12/2020



Sai Praneeth Karimireddy, Lie He, Martin Jaggi
Learning from History for Byzantine Robust Optimization
ICML 2021

Prior robust aggregation rules can fail – fix via iterative clipping & momentum

6/2022

Byz-MARINA-PAGE



Eduard Gorbunov, Samuel Horvath, PR, and Gauthier Gidel
Variance Reduction is an Antidote to Byzantine Workers: Better Rates, Weaker Assumptions and Communication Compression as a Cherry on the Top
ICLR 2023

Shows that variance reduction is useful to work with Byzantine workers

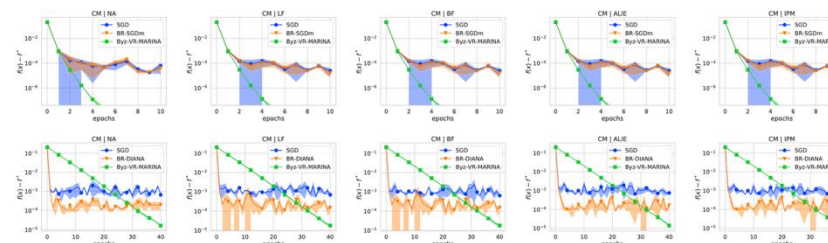
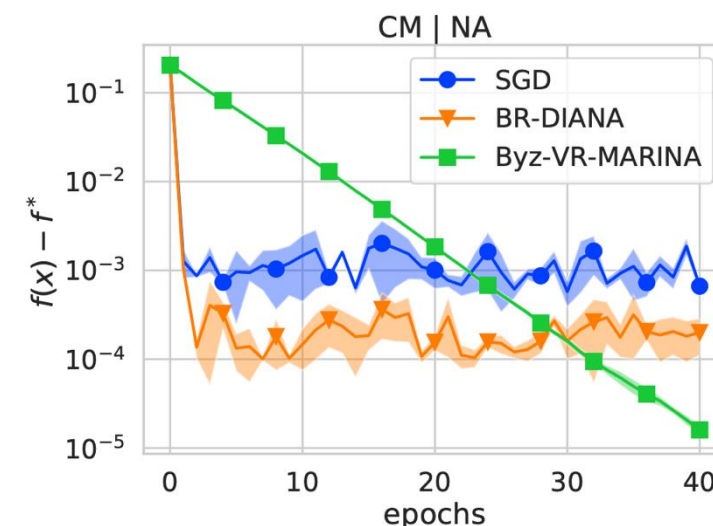


Figure 1: The optimality gap $f(x^k) - f(x^*)$ of 3 aggregation rules (AVG, CM, RFA) under 5 attacks (NA, LF, BF, ALIE, IPM) on a9a dataset, where each worker has access full dataset with 4 good and 1 Byzantine worker. In the first row, we do not use any compression, in the second row each method uses RandK sparsification with $K = 0.1d$.



10/2023

Byz-DASHA-PAGE



Ahmad Rammal, Kaja Gruntkowska, Nikita Fedin, Eduard Gorbunov, and P.R.
Communication compression for Byzantine robust learning: New efficient algorithms and improved rates
AISTATS 2024

DASHA improves on MARINA when applied to Byzantine setup

Model Heterogeneity

2/2021

Ordered
Dropout

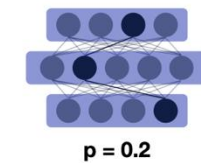


Samuel Horváth, Stefanos Laskaridis, Mario Almeida, Ilias Leontiadis, Stylianos I. Venieris, Nicholas D. Lane

FjORD: Fair and Accurate Federated Learning under Heterogeneous Targets with Ordered Dropout

NeurIPS 2021

Random Dropout (RD)



Ordered Dropout (OD)

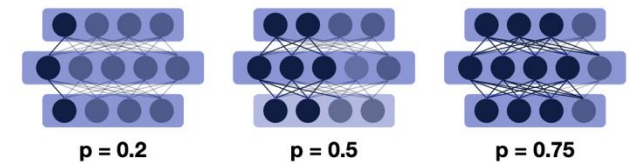


Figure 2: Ordered vs. Random Dropout. In this example, the left-most features are used by more devices during training, creating a natural ordering to the importance of these features.

4/2024

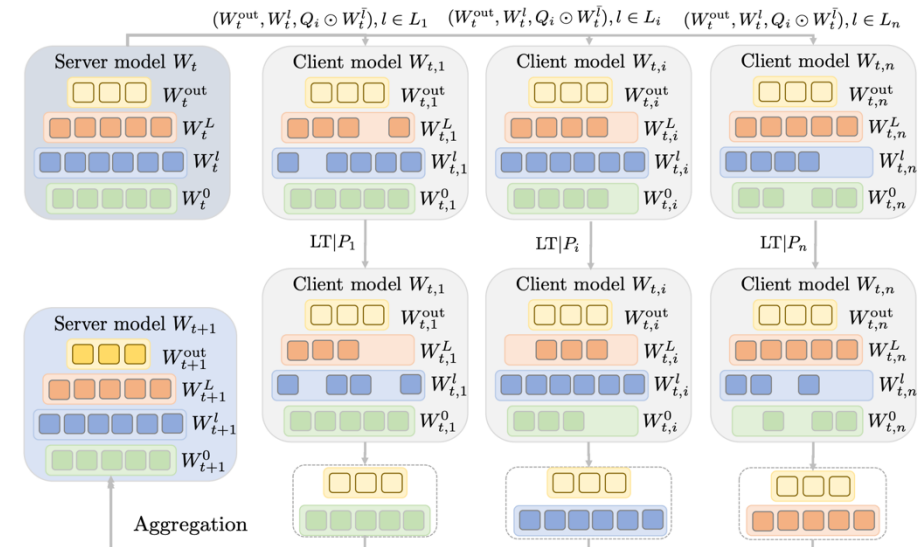
FedP3



Kai Yi, Nidham Gazagnadou, P.R., and Lingjuan Lyu

FedP3: Personalized and privacy-friendly federated network pruning under model heterogeneity

ICLR 2024



Objective Inconsistency

7/2020
FedNova



Jianyu Wang, Qinghua Liu, Hao Liang, Gauri Joshi, H. Vincent Poor
Tackling the Objective Inconsistency Problem in Heterogeneous Federated Optimization
NeurIPS 2020

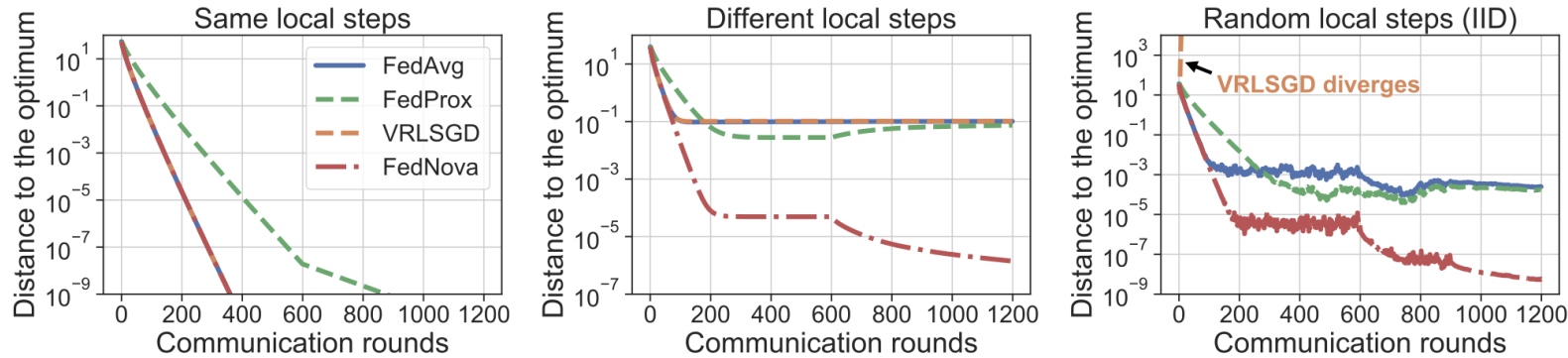


Figure 2: Simulations comparing the FedAvg, FedProx ($\mu = 1$), VRLSGD and our proposed FedNova algorithms for 30 clients with the quadratic objectives defined in (3), where $e_i \sim \mathcal{N}(0, 0.01I)$, $i \in [1, 30]$. Clients perform GD with $\eta = 0.05$, which is decayed by a factor of 5 at rounds 600 and 900. **Left:** Clients perform the same number of local steps $\tau_i = 30$ – FedNova is equivalent to FedAvg in this case; **Middle:** Clients take different local steps $\tau_i \in [1, 96]$ with mean 30 but fixed across rounds; **Right:** local steps are IID, and time-varying Gaussians with mean 30, *i.e.*, $\tau_i(t) \in [1, 96]$. FedNova significantly outperforms others in the heterogeneous τ_i setting.

Assorted Topics: Privacy, Federated Analytics/Computations, Applications, Systems, ...

4/2024



Hubert Eichner, Daniel Ramage, Kallista Bonawitz, Dzmitry Huba, Tiziano Santoro, Brett McLarnon, Timon Van Overveldt, Nova Fallen, Peter Kairouz, Albert Cheu, Katharine Daly, Adria Gascon, Marco Gruteser, and Brendan McMahan

Confidential Federated Computations

arXiv 2404.10764

8/2024

BLT-DP-FTRL



H. Brendan McMahan, Zheng Xu, Yanxiang Zhang

A Hassle-free Algorithm for Private Learning in Practice: Don't Use Tree Aggregation, Use BLTs

arXiv 2408.08868

10/2024



Katharine Daly, Hubert Eichner, Peter Kairouz, H. Brendan McMahan, Daniel Ramage, Zheng Xu

Federated Learning in Practice: Reflections and Projections

arXiv 2410.08892

2/2025

Clip21-SGD2M



Rustem Islamov, Samuel Horváth, Aurelien Lucchi, P.R., and Eduard Gorbunov

Double Momentum and Error Feedback for Clipping with Fast Rates and Differential Privacy

arXiv 2502.1168

1st time achieving strong DP & optimization guarantees without requiring bounded gradients/heterogeneity

How? Via a new way of combining clipping, heavy ball momentum and error feedback

More Notable Papers

Secure Distributed Training at Scale

Eduard Gorbunov^{*123} Alexander Borzunov^{*43} Michael Diskin⁴³ Max Ryabinin⁴³

Abstract

Many areas of deep learning benefit from using increasingly larger neural networks trained on public data, as is the case for pre-trained models for NLP and computer vision. Training such models requires a lot of computational resources (e.g., HPC clusters) that are not available to small research groups and independent researchers. One way to address it is for several smaller groups to pool their computational resources together and train a model that benefits all participants. Unfortunately, in this case, any participant can jeopardize the entire training run by sending incorrect updates, deliberately or by mistake. Training in presence of such peers requires specialized distributed training algorithms with Byzantine tolerance. These algorithms often sacrifice efficiency by introducing redundant communication or passing all updates through a trusted server, making it infeasible to apply them to large-scale deep learning, where models can have billions of parameters. In this work, we propose a novel protocol for secure (Byzantine-tolerant) decentralized training that emphasizes communication efficiency.

Recently, similar techniques were proposed for deep learning. They aim to solve the challenges caused by the sheer computational complexity of many machine learning tasks, such as pretraining transformers for NLP (Devlin et al., 2019; Brown et al., 2020; Liu et al., 2019) or learning on huge datasets in vision (Sun et al., 2017; Kolesnikov et al., 2020; Goyal et al., 2021). Recent works (Kjisipongse et al., 2018; Ryabinin & Gusev, 2020; Atre et al., 2021; Diskin et al., 2021) propose several systems sharing the computation across many volunteers that donate the idle time of their computers to train large models on public datasets.

Despite their strengths, volunteer computing systems have so far seen limited practical applications (Kjisipongse et al., 2018). A major roadblock towards the global adoption of these techniques is trust in reliability of each participant. For distributed training, all progress made by the collaboration can be undermined if a single peer sends incorrect outputs due to an error in computation (Smith, 2019) or malicious intent (Tolpegin et al., 2020).

Prior art in decentralized optimization proposed several optimization algorithms that are resistant to such “Byzantine” faults. However, most Byzantine-tolerant training protocols require either passing all updates through a trusted central server or exchanging additional messages that in-

FEDERATED LEARNING CAN FIND FRIENDS THAT ARE ADVANTAGEOUS

A PREPRINT

Nazarii Tupitsa
Department of Machine Learning
MBZUAI

Samuel Horváth
Department of Machine Learning
MBZUAI

Martin Takáč
Department of Machine Learning
MBZUAI

Eduard Gorbunov
Department of Machine Learning
MBZUAI

ABSTRACT

In Federated Learning (FL), the distributed nature and heterogeneity of client data present both opportunities and challenges. While collaboration among clients can significantly enhance the learning process, not all collaborations are beneficial; some may even be detrimental. In this study, we introduce a novel algorithm that assigns adaptive aggregation weights to clients participating in FL training, identifying those with data distributions most conducive to a specific learning objective. We demonstrate that our aggregation method converges no worse than the method that aggregates only the updates received from clients with the same data distribution. Furthermore, empirical evaluations consistently reveal that collaborations guided by our algorithm outperform traditional FL approaches. This underscores the critical role of judicious client selection and lays the foundation for more streamlined and effective FL implementations in the coming years.

Moshpit SGD: Communication-Efficient Decentralized Training on Heterogeneous Unreliable Devices

Max Ryabinin*
Yandex, Russia
HSE University, Russia

Eduard Gorbunov*
MIPT, Russia
HSE University, Russia
Yandex, Russia

Vsevolod Plokhotnyuk
Yandex, Russia
HSE University, Russia

Gennady Pekhimenko
University of Toronto, Canada
Vector Institute, Canada

Abstract

Training deep neural networks on large datasets can often be accelerated by using multiple compute nodes. This approach, known as distributed training, can utilize hundreds of computers via specialized message-passing protocols such as Ring All-Reduce. However, running these protocols at scale requires reliable high-speed networking that is only available in dedicated clusters. In contrast, many real-world applications, such as federated learning and cloud-based distributed training, operate on unreliable devices with unstable network bandwidth. As a result, these applications are restricted to using parameter servers or gossip-based averaging protocols. In this work, we lift that restriction by proposing Moshpit All-Reduce — an iterative averaging protocol that exponentially converges to the global average. We demonstrate the efficiency of our protocol for distributed optimization with strong theoretical guarantees. The experiments show 1.3x speedup for ResNet-50 training on ImageNet compared to competitive gossip-based strategies and 1.5x speedup when training ALBERT-large on preemptible compute nodes.

The background is a traditional Chinese ink wash painting. It features dark, jagged mountain peaks in the foreground and middle ground, rendered in shades of blue and grey. The sky is a pale, hazy pinkish-white. Delicate, dark brown branches with clusters of small, pink cherry blossoms and some falling petals are scattered across the upper and side portions of the image. The overall style is soft and artistic, typical of traditional Chinese landscape art.

Lecture 2

Communication Compression

Federated Training of a Supervised Machine Learning Model

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

devices /
machines

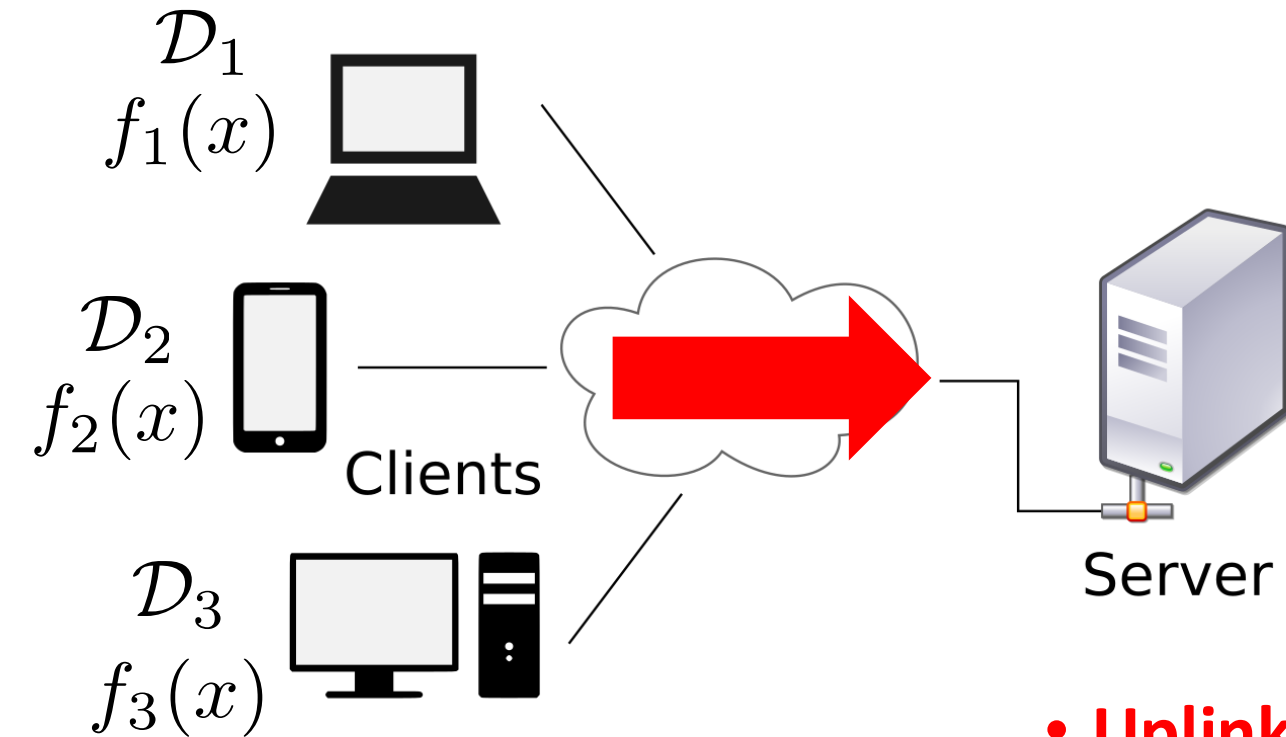
model parameters / features

Loss on local data $\mathcal{D}_i$ stored on device i

$$f_i(x) = \mathbb{E}_{\xi \sim \mathcal{D}_i} f_{i,\xi}(x)$$

The datasets $\mathcal{D}_1, \dots, \mathcal{D}_n$ can be arbitrarily heterogeneous

Communication is the Bottleneck



$$f(x) = \frac{1}{3} \sum_{i=1}^3 f_i(x)$$

Cost model:

- Uplink (w2s) communication cost = 1 (per bit)
- Worker computation cost = 0
- Server aggregation cost = 0
- Downlink (=s2w) communication cost = 0

Distributed Compressed Gradient Descent (DCGD)



Dan Alistarh, Demjan Grubic, Jerry Li, Ryota Tomioka, and Milan Vojnovic
QSGD: Communication-efficient SGD via gradient quantization and encoding
NeurIPS 2017



Sarit Khirirat, Hamid Reza Feyzmahdavian, and Mikael Johansson
Distributed learning with compressed gradients
arXiv 1806.06573

Distributed Compressed Gradient Descent (DCGD)

$$x^{k+1} = x^k - \gamma^k g^k$$

$$g^k = \frac{1}{n} \sum_{i=1}^n \mathcal{C}_i^k(\nabla f_i(x^k))$$

Compression operator

$$\mathcal{C}_i^k : \mathbb{R}^d \rightarrow \mathbb{R}^d$$



Dan Alistarh, Demjan Grubic, Jerry Li, Ryota Tomioka, and Milan Vojnovic

QSGD: Communication-efficient SGD via gradient quantization and encoding

NeurIPS 2017



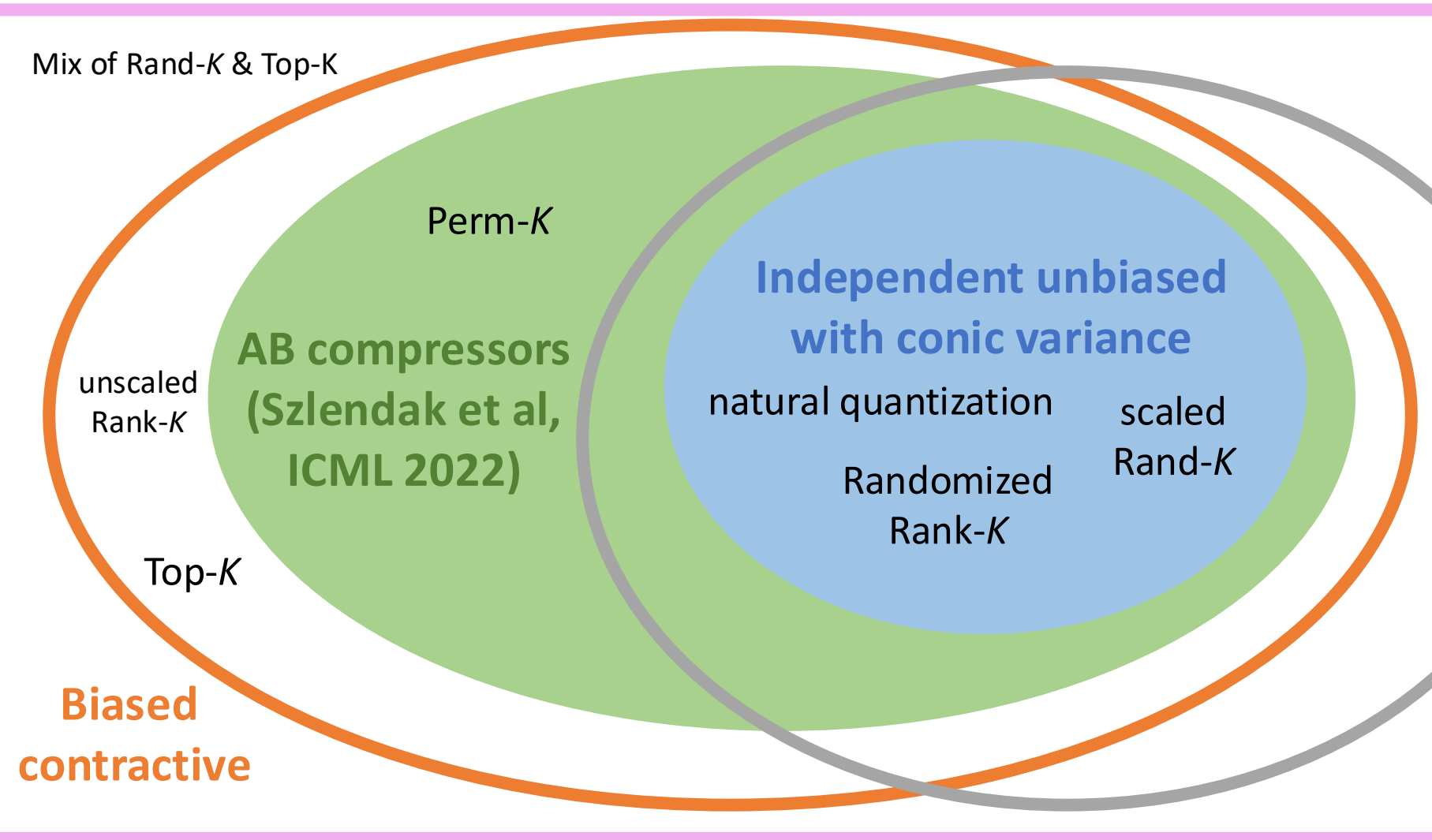
Sarit Khirirat, Hamid Reza Feyzmahdavian, and Mikael Johansson

Distributed learning with compressed gradients

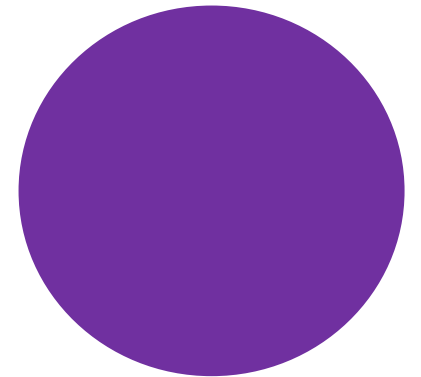
arXiv 1806.06573

Compression Operators: Selected Classes

Bias-Variance (BV) compressors (Condat et al, NeurIPS 2022)



Randomized integer rounding (Mishchenko et al, ICLR 2022)



Data-aware independent unbiased with conic variance (Safaryan et al, NeurIPS 2021)

Papers with Definitions and Examples of Compression Operators 1

10/2017
sparsification



Jianqiao Wangni, Jialei Wang, Ji Liu, Tong Zhang
Gradient Sparsification for Communication-Efficient Distributed Optimization
NIPS 2018

09/2018
quantization



Dan Alistarh, Demjan Grubic, Jerry Li, Ryota Tomioka, and Milan Vojnovic
QSGD: Communication-efficient SGD via gradient quantization and encoding
NeurIPS 2017

6/2018
quantization



Sarit Khirirat, Hamid Reza Feyzmahdavian, and Mikael Johansson
Distributed learning with compressed gradients
arXiv 1806.06573

5/2019
natural
quantization



Samuel Horváth, Chen-Yu Ho, Ľudovít Horváth, Atal Narayan Sahu, Marco Canini, P.R.
Natural compression for distributed deep learning
MSML 2022

Approach	$\mathcal{C}_{W_i}$	No. iterations $T^r(\omega_W) = \mathcal{O}((\omega_W + 1)^\theta)$	Bits per 1 Iter. $W_i \mapsto M$	Speedup Factor
Baseline	identity	1	$32d$	1
New	$\mathcal{C}_{\text{nat}}$	$(9/s)^\theta$	$9d$	$3.2\times\text{--}3.6\times$
Sparsification	$\mathcal{S}^q$	$(d/q)^\theta$	$(33 + \log_2 d)q$	$0.6\times\text{--}6.0\times$
New	$\mathcal{C}_{\text{nat}} \circ \mathcal{S}^q$	$(9d/sq)^\theta$	$(10 + \log_2 d)q$	$1.0\times\text{--}10.7\times$
Dithering	$\mathcal{D}_{\text{sta}}^{p, 2^{s-1}}$	$(1 + \kappa d^{1/p} 2^{1-s})^\theta$	$31 + d(2 + s)$	$1.8\times\text{--}15.9\times$
New	$\mathcal{D}_{\text{nat}}^{p, s}$	$(9/s + \kappa d^{1/p} 2^{1-s})^\theta$	$31 + d(2 + \log_2 s)$	$4.1\times\text{--}16.0\times$

Papers with Definitions and Examples of Compression Operators 2

2/2020
Kashin
compression



Mher Safaryan, Egor Shulgin and P.R.

Uncertainty principle for communication compression in distributed and federated learning and the search for an optimal compressor
Information and Inference: A Journal of the IMA, 2021

COMPRESSION METHOD	UNBIASED?	VARIANCE ω	VARIANCE α	BITS b (in binary32)
RANDOM SPARSIFICATION	YES	$\frac{d}{k} - 1 \approx \mathcal{O}(\frac{d}{k})$	$1 - \frac{k}{d}$	$32k + \log_2 \binom{d}{k}$
TOP- k SPARSIFICATION	NO	$\min(\frac{\sqrt{d}}{s}, \frac{d}{s^2}) \approx \mathcal{O}(\frac{\sqrt{d}}{s})$		$32k + \log_2 \binom{d}{k}$
STANDARD DITHERING	YES	$\min(\frac{\sqrt{d}}{2^{s-1}}, \frac{d}{2^{2s-2}}) \approx \mathcal{O}(\frac{\sqrt{d}}{2^{s-1}})$		$\mathcal{O}(s(s + \sqrt{d}))$
NATURAL DITHERING	YES	$\min(\frac{\sqrt{d}}{2^{s-1}}, \frac{d}{2^{2s-2}}) \approx \mathcal{O}(\frac{\sqrt{d}}{2^{s-1}})$		$31 + d \log_2(2s + 1)$
TERNARY QUANTIZATION	YES	$\sqrt{d} - 1 \approx \mathcal{O}(\sqrt{d})$	$1 - \frac{1}{d}$	$31 + d \log_2 3$
SCALED SIGN QUANTIZATION	NO			$31 + d$
Kashin Compression (new)	YES	$(\frac{10\sqrt{\lambda}}{\sqrt{\lambda}-1})^4 \approx \mathcal{O}(1)$		$31 + \log_2 3 \cdot \lambda d$

2/2020
many...



Aleksandr Beznosikov, Samuel Horváth, P.R. and Mher Safaryan
On biased compression for distributed learning
JMLR 2022

Compression Operator C	Unbiased?	α	β	γ	δ	ζ
Unbiased random sparsification	✓					d/k
Biased random sparsification [NEW]	✗	q	1	q	$1/q$	
Adaptive random sparsification [NEW]	✗	$1/d$	1	$1/d$	d	
Top- k sparsification (Alistarh et al., 2018)	✗	k/d	1	k/d	d/k	
General unbiased rounding [NEW]	✓					$\frac{1}{4} \sup \left(\frac{a_{k-1}}{a_{k+1}} + \frac{a_{k+1}}{a_k} + 2 \right)$
Unbiased exponential rounding [NEW]	✓					$\frac{1}{4} \left(b + \frac{1}{b} + 2 \right)$
Biased exponential rounding [NEW]	✗	$\left(\frac{2}{b+1} \right)^2$	$\frac{2b}{b+1}$	$\frac{2}{b+1}$	$\frac{(b+1)^2}{4b}$	
Natural compression (Horváth et al., 2019a)	✓					$9/k$
General exponential dithering [NEW]	✓					ζ_b
Natural dithering (Horváth et al., 2019a)	✓					ζ_2
Top- k + exponential dithering [NEW]	✗	k/d	ζ_b	k/d	$\zeta_b d/k$	

10/2020



Alyazeed Albasyoni, Mher Safaryan, Laurent Condat, and P.R.

Optimal gradient compression for distributed and federated learning

SpicyFL 2020: NeurIPS Workshop on Scalability, Privacy, and Security in Federated Learning

2/2021



Konstantin Mishchenko, Bokun Wang, Dmitry Kovalev and P.R.

IntSGD: Floatless compression of stochastic gradients

ICLR 2022

Papers with Definitions and Examples of Compression Operators 3

2/2021

data-aware
sparsification



Mher Safaryan, Filip Hanzely and P.R.

Smoothness matrices beat smoothness constants: better communication compression techniques for distributed optimization

NeurIPS 2021



6/2021

data-aware
quantization



Bokun Wang, Mher Safaryan and P.R.

Theoretically better and numerically faster distributed optimization with smoothness-aware quantization techniques

NeurIPS 2022



10/2021

AB compressors
Perm-K



Rafał Szlendak, Alexander Tyurin and P.R.

Permutation compressors for provably faster distributed nonconvex optimization

ICLR 2022



5/2022



Laurent Condat, Kai Yi and P.R.

EF-BV: A unified theory of error feedback and variance reduction mechanisms for biased and unbiased compression in distributed optimization

NeurIPS 2022



Sparsification vs Quantization

Sparsification

Reduce the number of entries
in the vector
from d to some small number K

Compression ratio = $d : K$



Sarit Khirirat, Hamid Reza Feyzmahdavian, and Mikael Johansson
Distributed learning with compressed gradients
arXiv 1806.06573



Aleksandr Beznosikov, Samuel Horváth, P.R. and Mher Safaryan
On biased compression for distributed learning
JMLR 2022

Quantization

Reduce the number of bits
representing every entry of the vector
from 32 to some small number b

Compression ratio = $32 : b$



Dan Alistarh, Demjan Grubic, Jerry Li, Ryota Tomioka, and Milan Vojnovic
QSGD: Communication-efficient SGD via gradient quantization and encoding
NeurIPS 2017



Samuel Horváth, Chen-Yu Ho, Ľudovít Horváth, Atal Narayan Sahu, Marco Canini, P.R.
Natural compression for distributed deep learning
MSML 2022

Rand-K Sparsification

Unbiased & independent compressors

$$\mathcal{C}_i^k : \mathbb{R}^d \rightarrow \mathbb{R}^d \quad \mathbb{E} [\mathcal{C}_i^k(v)] = v$$

$$\mathbb{E} [\|\mathcal{C}_i^k(v) - v\|^2] \leq \omega_i \|v\|^2$$

Example: $d = 5; K = 1$

$$\begin{pmatrix} 4 \\ -7 \\ 2 \\ 1 \\ -3 \end{pmatrix} \xrightarrow{\mathcal{C}_i^k} 5 \times \begin{pmatrix} 0 \\ 0 \\ 2 \\ 0 \\ 0 \end{pmatrix}$$

Random entry

$$\omega_i = \frac{d}{K} - 1 = \frac{5}{1} - 1 = 4$$

Rand-K Sparsification

Unbiased & independent compressors

$$\mathcal{C}_i^k : \mathbb{R}^d \rightarrow \mathbb{R}^d \quad \mathbb{E} [\mathcal{C}_i^k(v)] = v$$

$$\mathbb{E} [\|\mathcal{C}_i^k(v) - v\|^2] \leq \omega_i \|v\|^2$$

Example: $d = 5; K = 1$

$$\begin{pmatrix} 4 \\ -7 \\ 2 \\ 1 \\ -3 \end{pmatrix} \xrightarrow{\mathcal{C}_i^k} 5 \times \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ -3 \end{pmatrix}$$

Random entry

$$\omega_i = \frac{d}{K} - 1 = \frac{5}{1} - 1 = 4$$

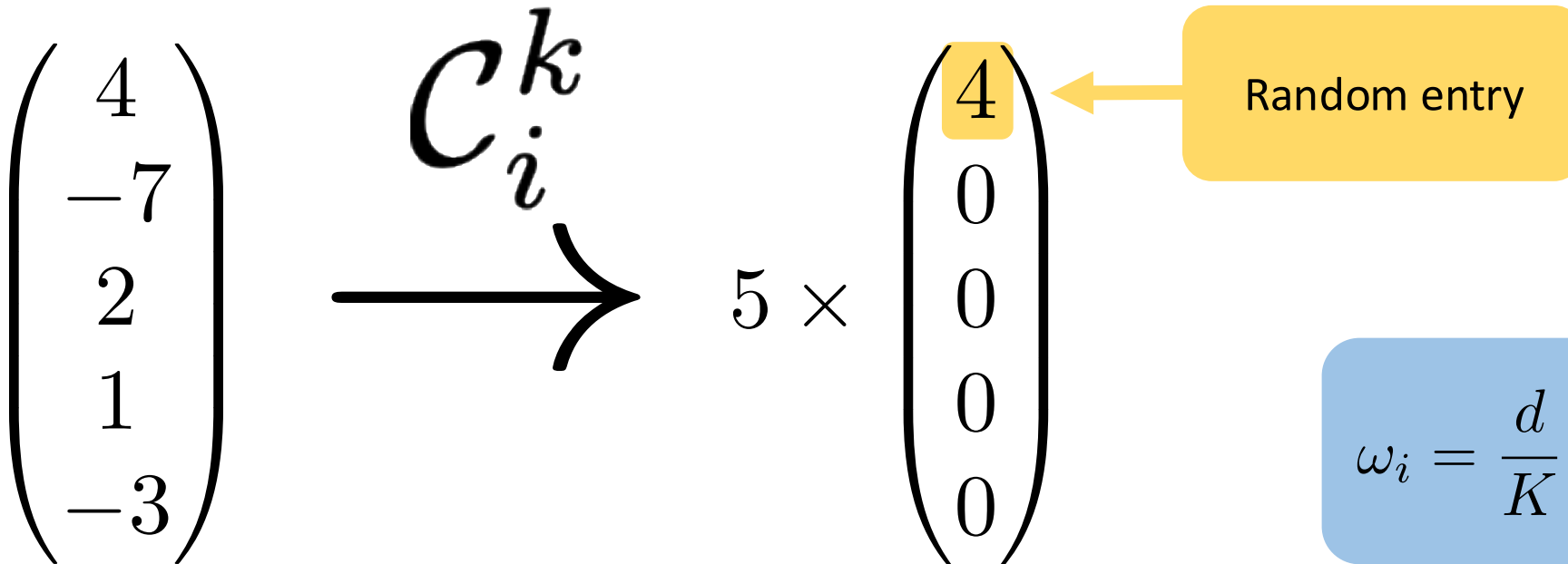
Rand- K Sparsification

Unbiased & independent compressors

$$\mathcal{C}_i^k : \mathbb{R}^d \rightarrow \mathbb{R}^d \quad \mathbb{E} [\mathcal{C}_i^k(v)] = v$$

$$\mathbb{E} [\|\mathcal{C}_i^k(v) - v\|^2] \leq \omega_i \|v\|^2$$

Example: $d = 5; K = 1$



$$\omega_i = \frac{d}{K} - 1 = \frac{5}{1} - 1 = 4$$

Top- K Sparsification

Contractive compression operator

$$\mathcal{C}_i^k : \mathbb{R}^d \rightarrow \mathbb{R}^d$$

$$\mathbb{E} \left[\|\mathcal{C}_i^k(v) - v\|^2 \right] \leq (1 - \alpha_i) \|v\|^2$$

Example: $d = 5; K = 1$

$$\begin{pmatrix} 4 \\ -7 \\ 2 \\ 1 \\ -3 \end{pmatrix} \xrightarrow{\mathcal{C}_i^k} \begin{pmatrix} 0 \\ -7 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

Largest entry in
absolute value

$$\alpha_i = \frac{K}{d} = \frac{1}{5}$$

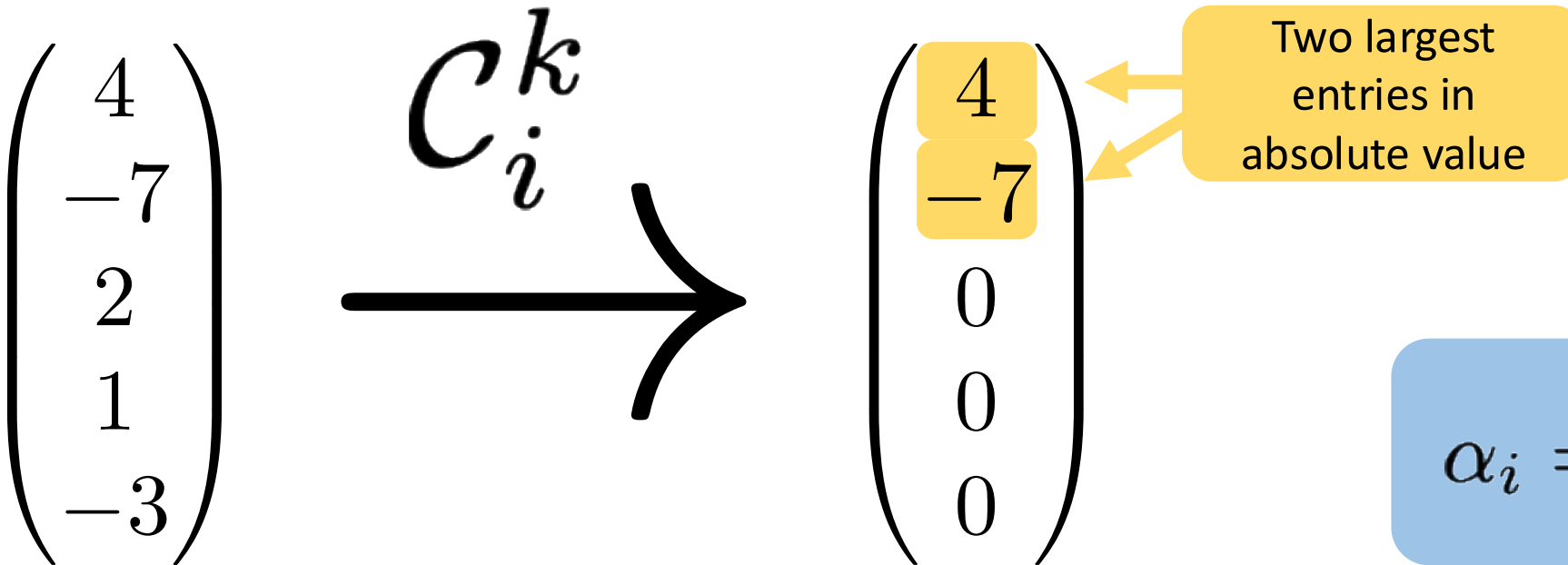
Top- K Sparsification

Contractive compression operator

$$\mathcal{C}_i^k : \mathbb{R}^d \rightarrow \mathbb{R}^d$$

$$\mathbb{E} \left[\|\mathcal{C}_i^k(v) - v\|^2 \right] \leq (1 - \alpha_i) \|v\|^2$$

Example: $d = 5; K = 2$



$$\alpha_i = \frac{K}{d} = \frac{2}{5}$$

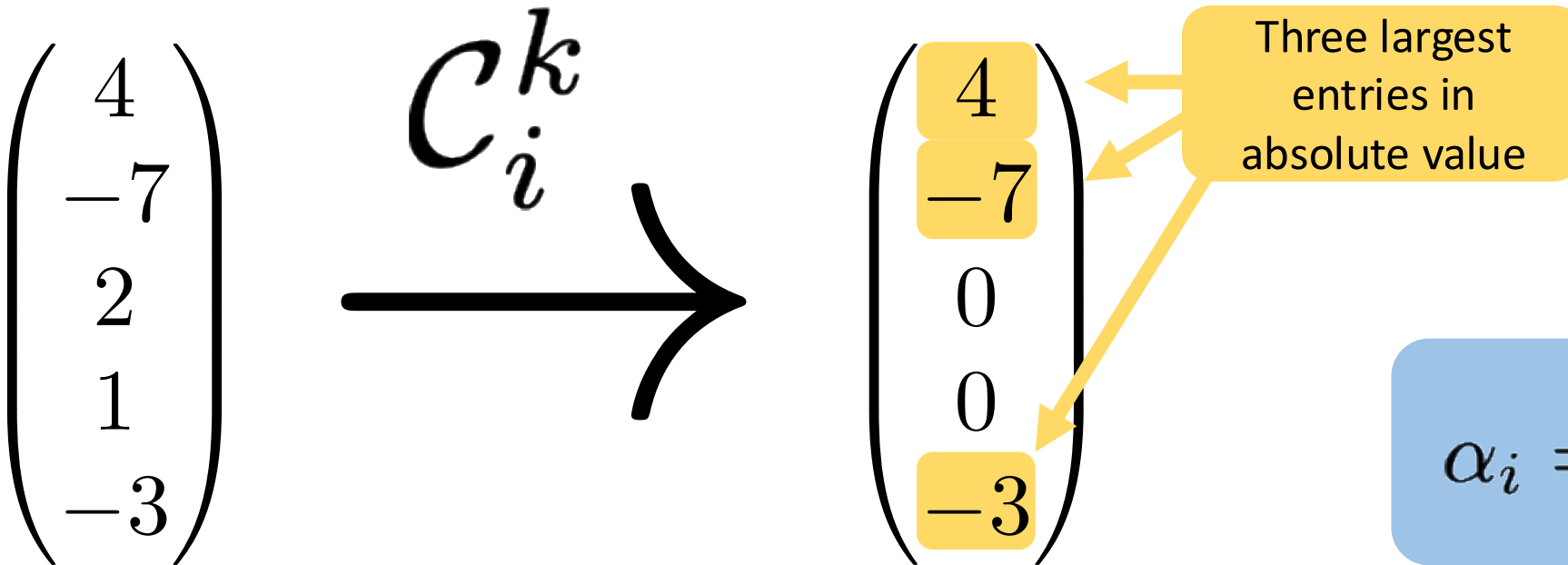
Top- K Sparsification

Contractive compression operator

$$\mathcal{C}_i^k : \mathbb{R}^d \rightarrow \mathbb{R}^d$$

$$\mathbb{E} \left[\|\mathcal{C}_i^k(v) - v\|^2 \right] \leq (1 - \alpha_i) \|v\|^2$$

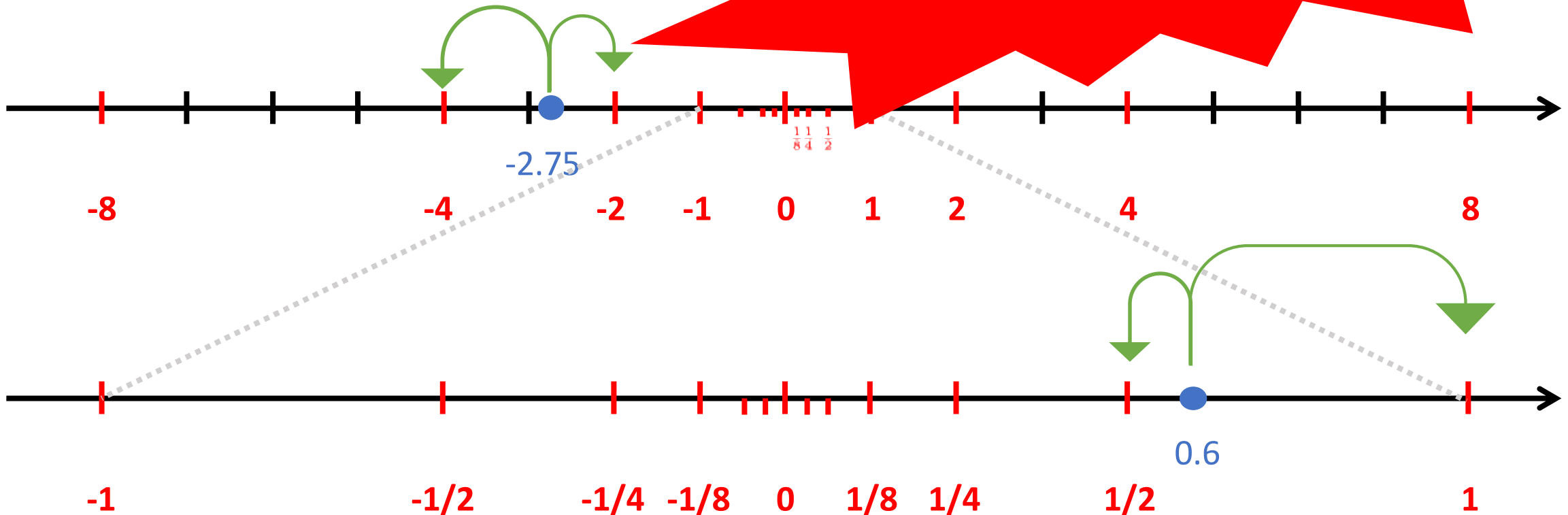
Example: $d = 5; K = 3$



$$\alpha_i = \frac{K}{d} = \frac{3}{5}$$

Natural Quantization (Rand. Rounding to the

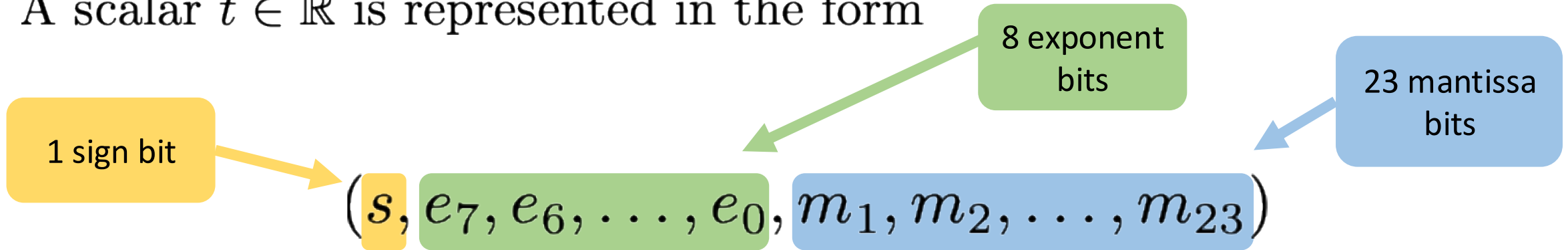
Exponential improvement (in variance or bits) over previous quantization techniques based on uniform subdivision of the real line



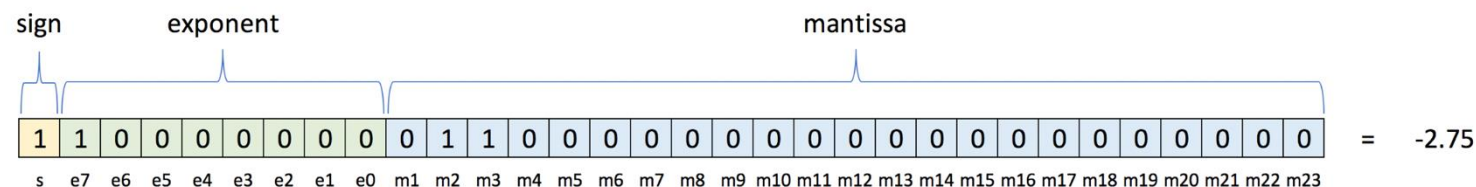
Samuel Horváth, Chen-Yu Ho, Ľudovít Horváth, Atal Narayan Sahu, Marco Canini and P.R.
Natural compression for distributed deep learning
MSML 2022 (arXiv 5/2019)

binary32 (IEEE 754 Single-Precision Binary Floating-Point Format)

A scalar $t \in \mathbb{R}$ is represented in the form



$$t = (-1)^s \times 2^{e-127} \times (1 + m), \quad e = \sum_{i=0}^7 e_i 2^i, \quad m = \sum_{j=1}^{23} m_j 2^{-j}$$



Natural Quantization of a Vector (Randomized Rounding Applied Coordinate-Wise)

Example: $d = 5$

$$\begin{pmatrix} 3.1 \\ -1.9 \\ 2.5 \\ 1.2 \\ -6.7 \end{pmatrix} \xrightarrow{\mathcal{C}_i} \begin{pmatrix} 4 \\ -2 \\ 4 \\ 1 \\ 8 \end{pmatrix}$$

Unbiased & independent compressors

$$\mathcal{C}_i^k : \mathbb{R}^d \rightarrow \mathbb{R}^d \quad \mathbb{E} [\mathcal{C}_i^k(v)] = v$$

$$\mathbb{E} [\|\mathcal{C}_i^k(v) - v\|^2] \leq \omega_i \|v\|^2$$

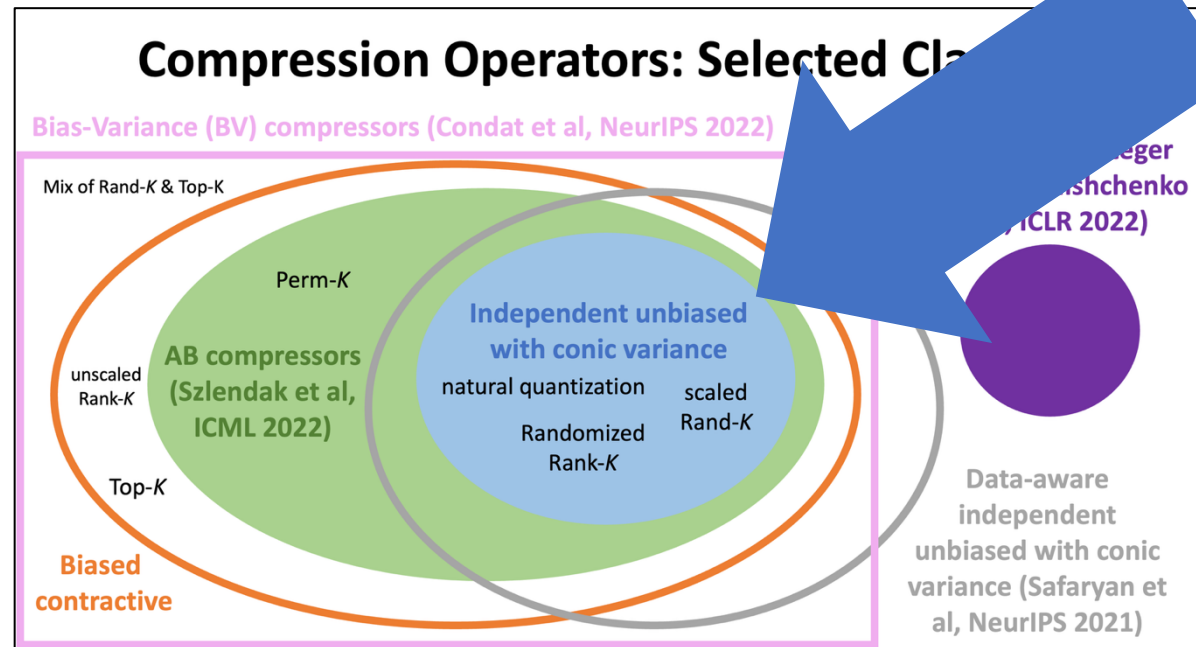
$$\omega_i = \frac{1}{8}$$



Samuel Horváth, Chen-Yu Ho, Lúdvít Horváth, Atal Narayan Sahu, Marco Canini and P.R.

Natural compression for distributed deep learning
MSML 2022 (arXiv 5/2019)

DCGD Theory for Independent & Unbiased Compressors with Conic Variance



DCGD Theory: Iteration Complexity

f_i is convex and L_i -smooth

$$f_i(y) + \langle \nabla f_i(y), x - y \rangle \leq f_i(x)$$

$$f_i(x) \leq f_i(y) + \langle \nabla f_i(y), x - y \rangle + \frac{L_i}{2} \|x - y\|^2$$

Unbiased & independent compressors

$$\mathcal{C}_i^k : \mathbb{R}^d \rightarrow \mathbb{R}^d \quad \mathbb{E} [\mathcal{C}_i^k(v)] = v$$

$$\mathbb{E} [\|\mathcal{C}_i^k(v) - v\|^2] \leq \omega_i \|v\|^2$$

$$k \geq \frac{L + \max_i \left\{ \frac{L_i \omega_i}{n} \right\}}{\mu} \log \frac{1}{\epsilon} \Rightarrow \mathbb{E} [\|x^k - x^*\|^2] \leq \epsilon \|x^0 - x^*\|^2$$

f is L -smooth

f is μ -convex

Assuming interpolation!



Robert Mansel Gower, Nicolas Loizou, Xun Qian, Alibek Sailanbayev, Egor Shulgin and P.R.
SGD: General analysis and improved rates
ICML 2019



P.R.
Introductory Lectures on Stochastic Gradient Descent
Manuscript 2026

Communication Complexity

GD vs DCGD

$\mathcal{C}_i^k$ = Rand-1 sparsifier
 $\omega_i = d - 1$

GD: $d \frac{L}{\mu} \log \frac{1}{\epsilon}$

DCGD: $\frac{L + \frac{(d-1)L_{\max}}{n}}{\mu} \log \frac{1}{\epsilon}$

Improvement Factor of DCGD over GD:

$$\text{IF} = \frac{d \frac{L}{\mu} \log \frac{1}{\epsilon}}{\frac{L + \frac{(d-1)L_{\max}}{n}}{\mu} \log \frac{1}{\epsilon}} = \frac{nd}{n + (d-1) \frac{L_{\max}}{L}} \in \left[1, \frac{nd}{n + d - 1} \right]$$

Can be achieved!

$1 \leq \frac{L_{\max}}{L} \leq n$

$\leq \min\{n, d\}$

DCGD+: Data-dependent Compressors

Matrix smoothness:

$$f_i(x) \leq f_i(y) + \langle \nabla f_i(y), x - y \rangle + \frac{1}{2} \|x - y\|_{\mathbf{L}_i}^2$$
$$\mathbf{L}_i \in \mathbb{S}_+^d$$

$$g^k = \frac{1}{n} \sum_{i=1}^n \mathbf{L}_i^{\frac{1}{2}} \mathcal{C}_i^k (\mathbf{L}_i^{\dagger \frac{1}{2}} \nabla f_i(x^k))$$

$$x^{k+1} = x^k - \gamma^k g^k$$

Unbiased & independent compression operators

$$\mathcal{C}_i^k : \mathbb{R}^d \rightarrow \mathbb{R}^d$$

$$\mathbb{E} [\mathcal{C}_i^k(v)] = v \quad \forall v \in \mathbb{R}^d$$

$$\mathbb{E} [\|\mathcal{C}_i(v) - v\|_{\mathbf{L}_i}^2] \leq \mathcal{L}_i \|v\|^2 \quad \forall v \in \mathbb{R}^d$$



Mher Safaryan, Filip Hanzely and P.R.
Smoothness matrices beat smoothness constants: better communication compression techniques for distributed optimization
NeurIPS 2021



Bokun Wang, Mher Safaryan and P.R.
Theoretically better and numerically faster distributed optimization with smoothness-aware quantization techniques
NeurIPS 2022

Iteration Complexity of DCGD vs DCGD+

$$f_i(x) \leq f_i(y) + \langle \nabla f_i(y), x - y \rangle + \frac{1}{2} \|x - y\|_{\mathbf{L}_i}^2$$
$$\mathbb{E} [\|C_i(v) - v\|_{\mathbf{L}_i}^2] \leq \mathcal{L}_i \|v\|^2 \quad \forall v \in \mathbb{R}^d$$

DCGD+

$$\frac{L + \max_i \left\{ \frac{\mathcal{L}_i}{n} \right\}}{\mu} \log \frac{1}{\epsilon}$$

DCGD

$$\frac{L + \max_i \left\{ \frac{L_i \omega_i}{n} \right\}}{\mu} \log \frac{1}{\epsilon}$$

Always true:

$$\mathcal{L}_i \leq L_i \omega_i$$

Can have:

$$\mathcal{L}_i \ll L_i \omega_i$$

IntGD = DCGD Based on Scaling & Integer Rounding

$$C_i^k(v) = \frac{1}{\alpha_k} \text{Int}(\alpha_k v)$$

$$\mathbb{E}[C_i^k(v)] = v$$

$$\mathbb{E}[\|C_i^k(v) - v\|^2] \leq \frac{d}{4\alpha_k^2}$$

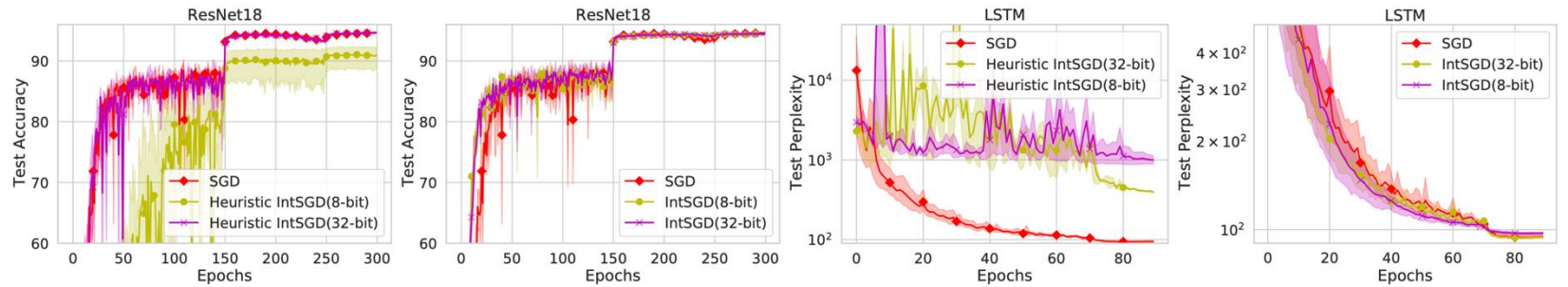


Figure 1: Comparison among IntSGD (8-bit or 32-bit), Heuristic IntSGD (8-bit or 32-bit), and full-precision SGD on the tasks of training ResNet18 and LSTM.

$$g^k = \frac{1}{n} \sum_{i=1}^n C_i^k(\nabla f_i(x^k))$$

$$x^{k+1} = x^k - \gamma^k g^k$$



Amedeo Sapio, Marco Canini, Chen-Yu Ho, Jacob Nelson, Panos Kalnis, Changhoon Kim, Arvind Krishnamurthy, Masoud Moshref, Dan R. K. Ports, and P.R.
Scaling distributed machine learning with in-network aggregation
NSDI 21



Konstantin Mishchenko, Bokun Wang, Dmitry Kovalev and P.R.
IntSGD: Adaptive Floatless Compression of Stochastic Gradients
ICLR 2022

IntGD: Benefits, Theory & Extensions

-  **Adaptive choice of the scaling factors!**
-  Convergence rates provided in these regimes:
 - Convex nonsmooth
 - Convex smooth
 - Nonconvex smooth
-  Supports all-reduce
-  Supports in-network/switch aggregation
-  Error-feedback not needed
-  Extends to stochastic gradients
-  Admits DIANA-style variance reduction

Table 1: Conceptual comparison of our method to the related literature. If all-reduce is supported, the method does not need any decompression. If all-reduce is not supported, the expensive all-gather operation is required and decompression is slow. See also Section 5 for numerical comparisons.

Algorithm	Supports all-reduce	Supports switch	Provably works	Fast compression	Works without error-feedback	Adaptive	Reference
IntSGD	✓	✓	✓	✓	✓	✓	Ours
Heuristic IntSGD	✓	✓	✗	✓	✓	✗	Sapio et al. (2021)
PowerSGD (theoretical)	✓	✗	✓	✗ ⁽¹⁾	✗	✗ ⁽²⁾	Vogels et al. (2019)
PowerSGD (practical)	✓	✗	✗	✓ ⁽¹⁾	✗	✗ ⁽²⁾	Vogels et al. (2019)
NatSGD	✗	✓	✓	✗	✓	N/A	Horváth et al. (2019)
QSGD	✗	✗	✓	✓	✓	N/A	Alistarh et al. (2017)
SignSGD	✗	✗	✓	✓	✗	N/A	Karimireddy et al. (2019)

⁽¹⁾ In theory, PowerSGD requires computing low-rank decompositions. In practice, an approximation is found by power iteration, which requires just a few matrix-vector multiplications but it is not analyzed theoretically and might be less stable.

⁽²⁾ PowerSGD requires tuning the rank of the low-rank decomposition. Vogels et al. (2019) reported that rank-1 PowerSGD consistently underperformed in their experiments, and, moreover, rank-2 was optimal for image classification while language modeling required rank-4. Ramesh et al. (2021) reported that a much larger rank was needed to avoid a gap in the training loss.



Konstantin Mishchenko, Bokun Wang, Dmitry Kovalev and P.R.
IntSGD: Adaptive Floatless Compression of Stochastic Gradients
 ICLR 2022

Variance Reduction for DCGD (DIANA)



Konstantin Mishchenko, Eduard Gorbunov, Martin Takáč and P.R.
Distributed learning with compressed gradient differences
arXiv:1901.09269, 2019

DIANA: Variance Reduction for Compression

Learned compression “shift”

$$h_i^k \rightarrow \nabla f_i(x^*)$$

Unbiased & independent compressors

$$\mathcal{C}_i^k : \mathbb{R}^d \rightarrow \mathbb{R}^d \quad \mathbb{E} [\mathcal{C}_i^k(v)] = v$$
$$\mathbb{E} [\|\mathcal{C}_i^k(v) - v\|^2] \leq \omega_i \|v\|^2$$

$$g^k = \frac{1}{n} \sum_{i=1}^n h_i^k + \mathcal{C}_i^k(\nabla f_i(x^k) - h_i^k)$$

$$h_i^{k+1} = h_i^k + \alpha \mathcal{C}_i^k(\nabla f_i(x^k) - h_i^k)$$

$$x^{k+1} = x^k - \gamma^k g^k$$



DIANA Theory: Iteration Complexity

f is L -smooth

$$\mathbb{E} [\|C_i^k(v) - v\|^2] \leq \omega_i \|v\|^2$$

$$\omega_{\max} := \max_i \omega_i$$

f_i is convex and L_i -smooth

No need to assume interpolation!

$$k \geq \max \left\{ \frac{L}{\mu} + 6 \max_i \left\{ \frac{\omega_i}{n}, \frac{L_i}{\mu} \right\}, 2(\omega_{\max} + 1) \right\} \log \frac{1}{\epsilon} \Rightarrow \mathbb{E} [V^k] \leq \epsilon V^0$$

f is μ -convex

Lyapunov function that implies

$$x^k \rightarrow x^* \quad h_i^k \rightarrow \nabla f_i(x^*)$$



Eduard Gorbunov, Filip Hanzely and P.R.
A unified theory of SGD: variance reduction, sampling, quantization and coordinate descent
AISTATS 2020

General analysis of variance-reduced SGD
 Includes DIANA as a special case



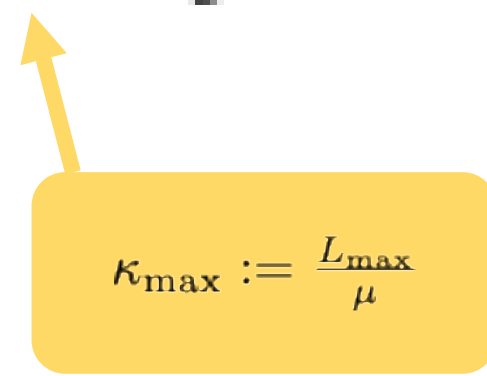
P.R.
Introductory Lectures on Stochastic Gradient Descent
Manuscript 2026

Includes the **tight / clean analysis of DIANA** described here

DIANA vs GD: Theoretical Speedup Factor in Communication Complexity

Improvement factor of DIANA over GD (for Rand-1 sparsification):

$$1 \leq \text{IF} \leq \min\{d, n, \kappa_{\max}\}$$


$$\kappa_{\max} := \frac{L_{\max}}{\mu}$$



P.R.

Introductory Lectures on Stochastic Gradient Descent
Manuscript 2026

Problem	Method	Alg #	Citation	VR?	AS?	Quant?	RCD?	Section	Result
(1)+(2)	SGD	Alg 1	[26]	✗	✗	✗	✗	A.1	Cor A.1
(1)+(3)	SGD-SR	Alg 2	[6]	✗	✓	✗	✗	A.2	Cor A.2
(1)+(3)	SGD-MB	Alg 3	NEW	✗	✗	✗	✗	A.3	Cor A.3
(1)+(3)	SGD-star	Alg 4	NEW	✓	✓	✗	✗	A.4	Cor A.4
(1)+(3)	SAGA	Alg 5	[5]	✓	✗	✗	✗	A.5	Cor A.5
(1)+(3)	N-SAGA	Alg 6	NEW	✗	✗	✗	✗	A.6	Cor A.6
(1)	SEGA	Alg 7	[11]	✓	✗	✗	✓	A.7	Cor A.7
(1)	N-SEGA	Alg 8	NEW	✗	✗	✗	✓	A.8	Cor A.8
(1)+(3)	SVRG ^a	Alg 9	[15]	✓	✗	✗	✗	A.9	Cor A.9
(1)+(3)	L-SVRG	Alg 10	[13, 18]	✓	✗	✗	✗	A.10	Cor A.10
(1)+(3)	DIANA	Alg 11	[20, 14]	✗	✗	✓	✗	A.11	Cor A.11
(1)+(3)	DIANA ^b	Alg 12	[20, 14]	✓	✗	✓	✗	A.11	Cor A.12
(1)+(3)	Q-SGD-SR	Alg 13	NEW	✗	✓	✓	✗	A.12	Cor A.13
(1)+(3)+(4)	VR-DIANA	Alg 14	[14]	✓	✗	✓	✗	A.13	Cor A.15
(1)+(3)	JacSketch	Alg 15	[9]	✓	✓✗	✗	✗	A.14	Cor A.16

Table 1: List of specific existing (in some cases generalized) and new methods which fit our general analysis framework. VR = variance reduced method, AS = arbitrary sampling, Quant = supports gradient quantization, RCD = randomized coordinate descent type method. ^a Special case of SVRG with 1 outer loop only; ^b Special case of DIANA with 1 node and quantization of exact gradient.



Eduard Gorbunov, Filip Hanzely and P.R.

A unified theory of SGD: variance reduction, sampling, quantization and coordinate descent

AISTATS 2020

DIANA: Further Sources

09/2018
SEGA



Filip Hanzely, Konstantin Mishchenko and P.R.
SEGA: Variance reduction via gradient sketching
NeurIPS 2018

First variance reduction for CGD
(compressor = sketch)

01/2019
DIANA



Konstantin Mishchenko, Eduard Gorbunov, Martin Takáč and P.R.
Distributed learning with compressed gradient differences
arXiv:1901.09269, 2019

First variance reduction for DCGD
DIANA first proposed & analyzed for
a special (ternary) compressor only

01/2019
IBCD, ISAGA,
ISEGA, ...



Konstantin Mishchenko, Filip Hanzely and P.R.
**99% of distributed optimization is a waste of time: the
issue and how to fix it**
UAI 2020

**Unlike original DIANA, which focused on
ternary quantization, here we focus on
sparsification**

10/2019
DIANA



Samuel Horváth, Dmitry Kovalev, Konstantin Mishchenko, P.R.
and Sebastian Stich
**Stochastic distributed learning with gradient quantization and
double variance reduction**
GOMS 2023

2023 Charles
Broyden Prize

Generalized DIANA in several
ways (general compressors,
added variance reduction for
stoch gradients)

Acceleration of DCGD/DIANA



Zhize Li, Dmitry Kovalev, Xun Qian and P.R.

Acceleration for compressed gradient descent in distributed and federated optimization

ICML 2020



Zhize Li and P.R.

CANITA: Faster rates for distributed convex optimization with communication compression

NeurIPS 2021

ADIANA: Acceleration in the Strongly Convex Case

Table 1. Convergence results for the special case with $n = 1$ device (i.e., problem (4))

Algorithm	μ -strongly convex f	convex f
Compressed Gradient Descent (CGD (Khairat et al., 2018))	$O\left((1 + \omega) \frac{L}{\mu} \log \frac{1}{\epsilon}\right)$	$O\left((1 + \omega) \frac{L}{\epsilon}\right)$
ACGD (this paper)	$O\left((1 + \omega) \sqrt{\frac{L}{\mu}} \log \frac{1}{\epsilon}\right)$	$O\left((1 + \omega) \sqrt{\frac{L}{\epsilon}}\right)$

Table 2. Convergence results for the general case with n devices (i.e., problem (1)). Our results are always better than previous results.

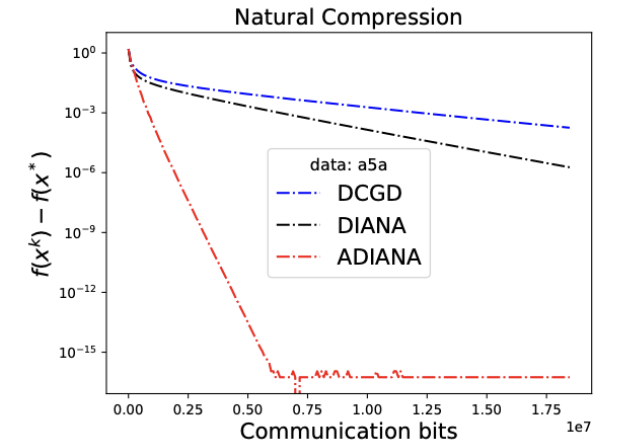
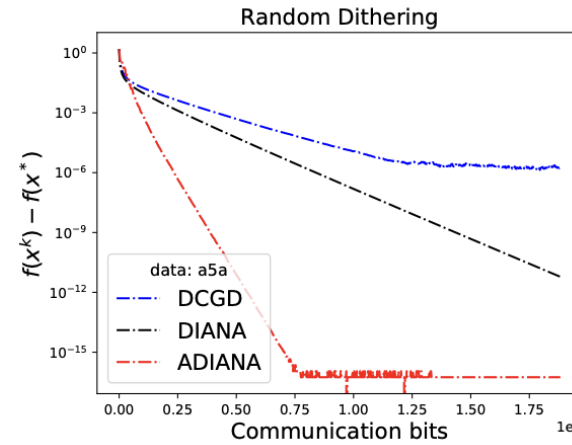
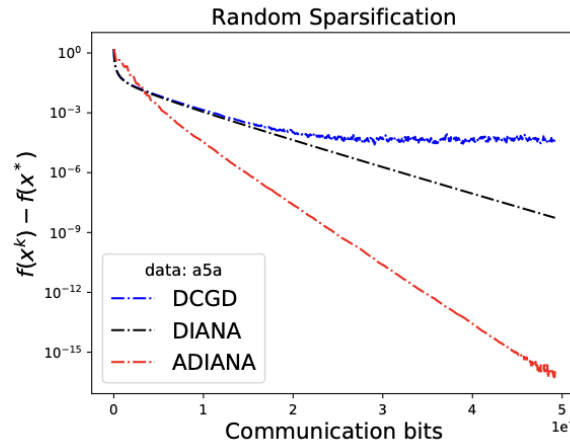
Algorithm	$n \leq \omega$ (few devices or high compression)	$n > \omega$ (lots of devices or low compression)
Distributed CGD (DIANA (Mishchenko et al., 2019))	$O\left(\omega \left(1 + \frac{L}{n\mu}\right) \log \frac{1}{\epsilon}\right)$	$O\left(\left(\omega + \frac{L}{\mu}\right) \log \frac{1}{\epsilon}\right)$
ADIANA (this paper)	$O\left(\omega \left(1 + \sqrt{\frac{L}{n\mu}}\right) \log \frac{1}{\epsilon}\right)$	$O\left(\left(\omega + \sqrt{\frac{L}{\mu}} + \sqrt{\frac{\omega L}{n\mu}}\right) \log \frac{1}{\epsilon}\right)$

Algorithm 3 Accelerated DIANA (ADIANA) by Li et al. (2020)

```

1: Parameters: Lipschitz-like parameter  $\bar{L} > 0$ , strong-convexity parameter  $\mu \geq 0$ , probability  $p$ , parameter  $\Gamma_0$ , initial point  $x^0 \in \mathbb{R}^d$ , initial gradient shifts  $h_1^0, \dots, h_n^0 \in \mathbb{R}^d$ 
2: Initialize  $\beta = 1/\omega + 1$  and  $z^0 = u^0 = x^0$ 
3: Initialize  $h^0 = \frac{1}{n} \sum_{i=1}^n h_i^0$ 
4: for  $t = 0, 1, \dots, T - 1$  do
5:    $\Gamma_{t+1}, \gamma_{t+1}, \theta_{t+1} = \text{CalculateOriginalLearningRates}(\dots)$ 
6:   for  $i = 1, \dots, n$  in parallel do
7:      $y^{t+1} = \theta_{t+1} u^t + (1 - \theta_{t+1}) z^t$ 
8:      $m_i^{t,y} = \mathcal{C}_i^{D,y}(\nabla f_i(y^{t+1}) - h_i^t)$  Worker  $i$  compresses the shifted gradient via the compressor  $\mathcal{C}_i^{D,y} \in \mathbb{U}(\omega)$ 
9:     Send compressed message  $m_i^{t,y}$  to the server
10:  end for
11:   $m^{t,y} = \frac{1}{n} \sum_{i=1}^n m_i^{t,y}$  Server averages the messages
12:   $g^{t+1} = h^t + m^{t,y} \equiv \frac{1}{n} \sum_{i=1}^n (h_i^t + m_i^{t,y})$ 
13:   $u^{t+1} = \arg \min_{x \in \mathbb{R}^d} \langle g^{t+1}, x \rangle + \frac{\bar{L} + \Gamma_{t+1} \mu}{2\gamma_{t+1}} \|x - u^t\|^2 + \frac{\mu}{2} \|x - y^{t+1}\|^2$  Server does a gradient-like descent step
14:  Flip a coin  $c^t \sim \text{Bernoulli}(p)$ 
15:  Broadcast non-compressed messages  $u^{t+1}$  to all  $n$  workers
16:  for  $i = 1, \dots, n$  in parallel do
17:     $x^{t+1} = \theta_{t+1} u^{t+1} + (1 - \theta_{t+1}) z^t$ 
18:     $z^{t+1} = \begin{cases} x^{t+1}, & c^t = 1 \\ z^t, & c^t = 0 \end{cases}$ 
19:     $m_i^{t,z} = \mathcal{C}_i^{D,z}(\nabla f_i(z^{t+1}) - h_i^t)$  Worker  $i$  compresses the shifted gradient via the compressor  $\mathcal{C}_i^{D,z} \in \mathbb{U}(\omega)$ 
20:     $h_i^{t+1} = h_i^t + \beta m_i^{t,z}$  Update the control variables
21:    Send compressed message  $m_i^{t,z}$  to the server
22:  end for
23:   $h^{t+1} = h^t + \beta \frac{1}{n} \sum_{i=1}^n m_i^{t,z}$  Server averages the messages
24: end for

```



Zhize Li, Dmitry Kovalev, Xun Qian and P.R.

Acceleration for compressed gradient descent in distributed and federated optimization

ICML 2020

CANITA: Acceleration of DCGD in the Convex Case

Table 1: Convergence rates for finding an ϵ -solution $\mathbb{E}[f(x^T)] - f(x^*) \leq \epsilon$ of distributed problem (1)

Algorithms	Strongly convex ^[1]	General convex	Remark
QSGD [1]	—	$O\left(\frac{L}{\epsilon} + \frac{\omega G^2}{n} \frac{1}{\epsilon^2}\right)$ ^[2]	✓ compression ✗ acceleration
DIANA [30]	$O\left(\left(1 + \frac{\omega}{n}\right)\kappa + \omega\right) \log \frac{1}{\epsilon}$	—	✓ compression ✗ acceleration
DIANA [9]	$O\left(\left(1 + \frac{\omega}{n}\right)\kappa + \omega\right) \log \frac{1}{\epsilon}$	$O\left(\left(1 + \frac{\omega}{n}\right)\frac{L}{\epsilon} + \frac{\omega}{\epsilon}\right)$	✓ compression ✗ acceleration
DIANA [12]	—	$O\left(\left(1 + \frac{\omega}{n}\right)\frac{L}{\epsilon} + \frac{\omega^2 + \omega}{\omega + n} \frac{1}{\epsilon}\right)$	✓ compression ✗ acceleration
ADIANA [26]	$O\left(\left(\sqrt{\kappa} + \sqrt{\left(\frac{\omega}{n} + \sqrt{\frac{\omega}{n}}\right)\omega\kappa} + \omega\right) \log \frac{1}{\epsilon}\right)$	—	✓ compression ✓ acceleration
CANITA (this paper)	—	$O\left(\sqrt{\left(1 + \sqrt{\frac{\omega^3}{n}}\right)\frac{L}{\epsilon}} + \omega\left(\frac{1}{\epsilon}\right)^{\frac{1}{3}}\right)$	✓ compression ✓ acceleration

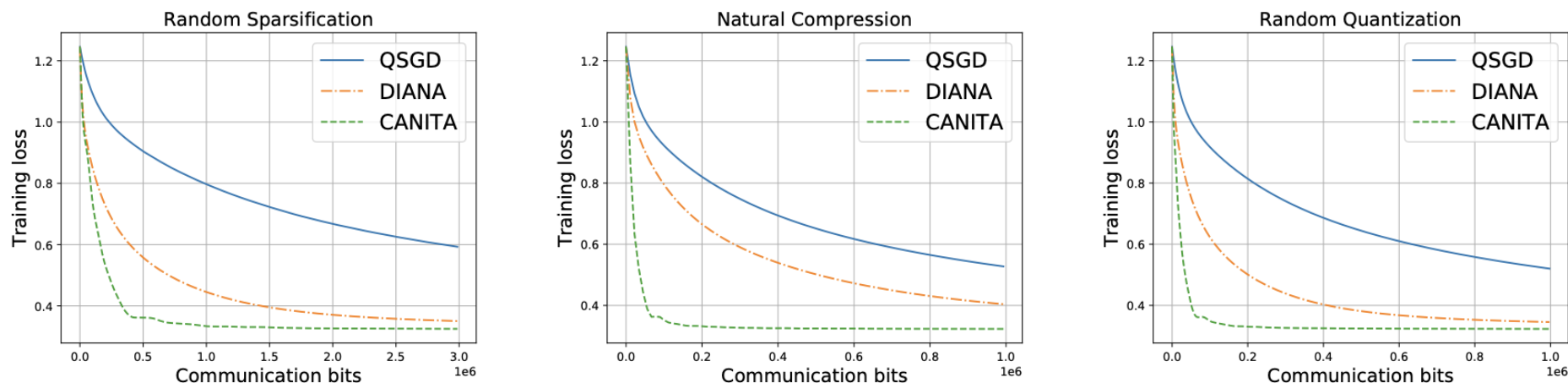


Figure 1: Performance of different methods for three different compressors (random sparsification, natural compression, and random quantization) on the a9a dataset.



Zhize Li and P.R.

CANITA: Faster rates for distributed convex optimization with communication compression

NeurIPS 2021

Error Feedback



Frank Seide, Hao Fu, Jasha Droppo, Gang Li, and Dong Yu

1-Bit Stochastic Gradient Descent and its Application to Data-Parallel Distributed Training of Speech DNNs

Interspeech 2014



Sebastian U. Stich, J.-B. Cordonnier, Martin Jaggi

Sparsified SGD with memory

NeurIPS 2018

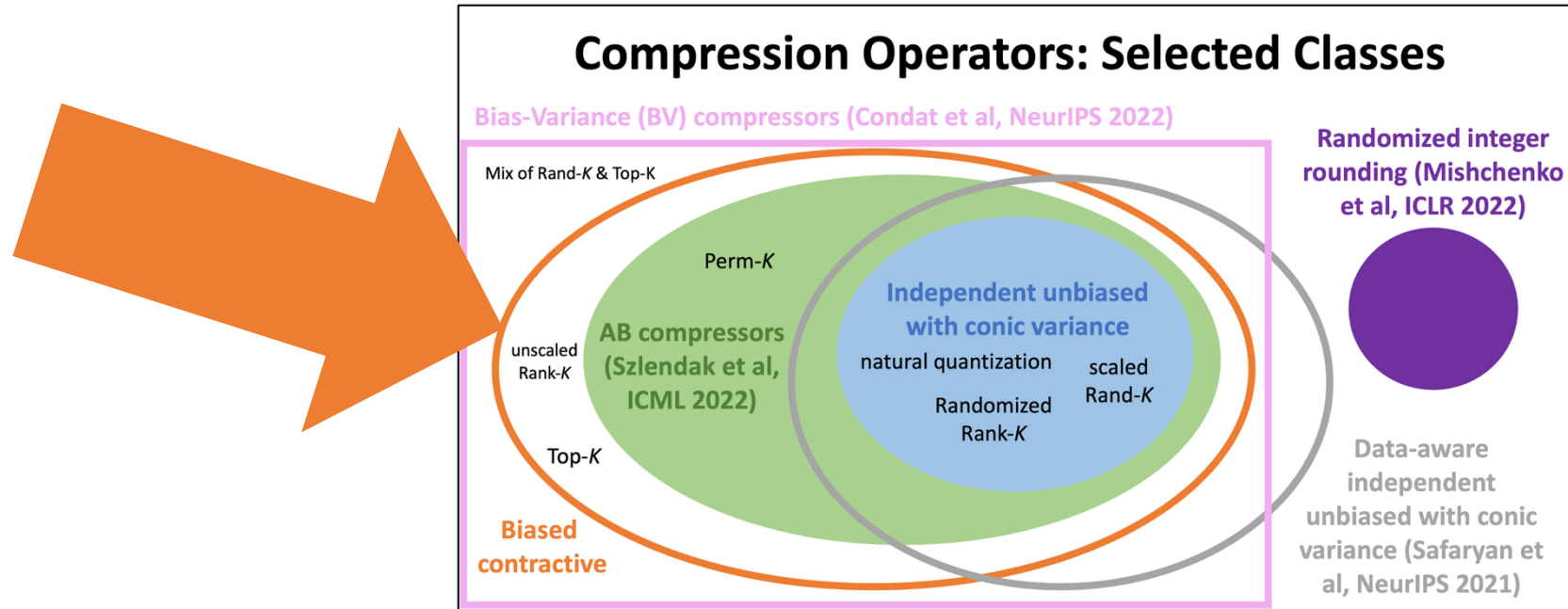


Peter Richtárik, Igor Sokolov and Ilyas Fatkhullin

EF21: A New, Simpler, Theoretically Better, and Practically Faster Error Feedback

NeurIPS 2021

Focus on Biased Contractive Compressors



For Contractive Compressors such as Top-K, DCGD can Diverge, and Exponentially so!!!

On Biased Compression for Distributed Learning

Aleksandr Beznosikov* Samuel Horváth Peter Richtárik Mher Safaryan

King Abdullah University of Science and Technology (KAUST)
Thuwal, Saudi Arabia

February 27, 2020

Abstract

In the last few years, various communication compression techniques have emerged as an indispensable tool helping to alleviate the communication bottleneck in distributed learning. However, despite the fact *biased* compressors often show superior performance in practice when compared to the much more studied and understood *unbiased* compressors, very little is known about them. In this work we study three classes of biased compression operators, two of which are new, and their performance when applied to (stochastic) gradient descent and distributed (stochastic) gradient descent. We show for the first time that biased compressors can lead to linear convergence rates both in the single node and distributed settings. Our *distributed* SGD method enjoys the ergodic rate $\mathcal{O}\left(\frac{\delta L \exp(-K)}{\mu} + \frac{(C+D)}{K\mu}\right)$, where δ is a compression parameter which grows when more compression is applied, L and μ are the smoothness and strong convexity constants, C captures stochastic gradient noise ($C = 0$ if full gradients are computed on each node) and D captures the variance of the gradients at the optimum ($D = 0$ for over-parameterized models). Further, via a theoretical study of several synthetic and empirical distributions of communicated gradients, we shed light on why and by how much biased compressors outperform their unbiased variants. Finally, we propose a new highly performing biased compressor—combination of Top- k and natural dithering—which in our experiments outperforms all other compression techniques.

Contents

1	Introduction	2
1.1	Distributed optimization	3
1.2	Contributions	3
1.3	Related work	4
1.4	Basic notation and definitions	5
2	Biased Compressors: Definitions & Theory	6
3	Biased Compressors: Old and New	6
4	Gradient Descent with Biased Compression	8
4.1	Complexity theory	8
4.2	$\mathbb{B}^3$ and $\mathbb{B}^2$ are better than $\mathbb{B}^1$	8
5	Superiority of Biased Compressors Under Statistical Assumptions	8
5.1	Top- k vs Rand- k	9
5.2	New compressor: Top- k combined with dithering	10

*This work was conducted while A. Beznosikov was a research intern in the Optimization and Machine Learning Lab of Peter Richtárik at KAUST; this visit was funded by the KAUST Baseline Research Funding Scheme. A. Beznosikov is affiliated with Moscow Institute of Physics and Technology, Russia and Sirius University of Science and Technology, Russia, and his research was also partially supported by RFBR, project number 19-31-51001. Alphabetical author order.

Example 1 Consider $n = d = 3$ and define

$$f_1(x) = \langle a, x \rangle^2 + \frac{1}{4} \|x\|_2^2, \quad f_2(x) = \langle b, x \rangle^2 + \frac{1}{4} \|x\|_2^2, \quad f_3(x) = \langle c, x \rangle^2 + \frac{1}{4} \|x\|_2^2,$$

where $a = (-3, 2, 2)$, $b = (2, -3, 2)$ and $c = (2, 2, -3)$. Let the starting iterate be $x^0 = (t, t, t)$, where $t > 0$. Then

$$\nabla f_1(x^0) = \frac{t}{2}(-11, 9, 9), \quad \nabla f_2(x^0) = \frac{t}{2}(9, -11, 9), \quad \nabla f_3(x^0) = \frac{t}{2}(9, 9, -11).$$

Using the Top-1 compressor, we get $\mathcal{C}(\nabla f_1(x^0)) = \frac{t}{2}(-11, 0, 0)$, $\mathcal{C}(\nabla f_2(x^0)) = \frac{t}{2}(0, -11, 0)$ and $\mathcal{C}(\nabla f_3(x^0)) = \frac{t}{2}(0, 0, -11)$. The next iterate of DCGD is

$$x^1 = x^0 - \eta \frac{1}{3} \sum_{i=1}^3 \mathcal{C}(\nabla f_i(x^0)) = \left(1 + \frac{11\eta}{6}\right) x^0.$$

Repeated application gives

$$x^k = \left(1 + \frac{11\eta}{6}\right)^k x^0.$$



Alexander Beznosikov, Samuel Horváth, Mher Safaryan, and Peter Richtárik

On biased compression for distributed learning

SpicyFL 2020: NeurIPS Workshop on Scalability, Privacy, and Security in Federated Learning

JMLR 2022

Error Feedback (EF14): Technique for Fixing the Divergence of DCGD

INTERSPEECH 2014



1-Bit Stochastic Gradient Descent and its Application to Data-Parallel Distributed Training of Speech DNNs

Frank Seide¹, Hao Fu^{1,2}, Jasha Droppo³, Gang Li¹, and Dong Yu³

¹ Microsoft Research Asia, 5 Danling Street, Haidian District, Beijing 100080, P.R.C.

² Institute of Microelectronics, Tsinghua University, 10084 Beijing, P.R.C

³ Microsoft Research, One Microsoft Way, Redmond, WA 98052, USA

{fseide, jdroppo, ganl, dongyu}@microsoft.com, fuhao9202@hotmail.com

Abstract

We show empirically that in SGD training of deep neural networks, one can, at no or nearly no loss of accuracy, quantize the gradients aggressively—to but *one bit* per value—if the quantization error is carried forward across minibatches (error feedback). This size reduction makes it feasible to parallelize SGD through data-parallelism with fast processors like recent GPUs.

We implement data-parallel deterministically distributed SGD by combining this finding with AdaGrad, automatic minibatch-size selection, double buffering, and model paral-

lizing efficiency for data parallelism are to increase the minibatch size and to reduce how much data gets exchanged [8].

We focus on the latter and propose to reduce bandwidth by *aggressively quantizing* the sub-gradients—to but *one bit per value*. We show that this does not or almost not reduce word accuracies—but only if the *quantization error is carried forward across minibatches*, i.e. the error in quantizing the gradient in one minibatch is added (fed back) to the gradient of the next minibatch. This is a common technique in other areas, such as sigma-delta modulation for DACs [9], or image rasterization. It is a key difference to the well-known R-prop method [27].

Error Feedback (EF14)

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

$$e^0 = 0$$
$$w^t = \mathcal{C}(e^t + \gamma \nabla f(x^t))$$

$$x^{t+1} = x^t - w^t$$

$$e^{t+1} = \underbrace{e^t + \gamma \nabla f(x^t)}_{\text{Update we wanted to apply}} - \underbrace{w^t}_{\text{Update we actually applied}}$$

Update we wanted to apply

Update we actually applied

Error Feedback (Distributed Form)

Algorithm 4 EF (Original error feedback)

- 1: Each node $i = 1, \dots, n$ sets the initial error to zero: $e_i^0 = 0$
 - 2: Each node $i = 1, \dots, n$ computes $w_i^0 = \mathcal{C}(\gamma \nabla f_i(x^0))$ and sends this to the master
 - 3: **for** $t = 0, 1, 2, \dots, T - 1$ **do**
 - 4: Master computes $x^{t+1} = x^t - \frac{1}{n} \sum_{i=1}^n w_i^t$
 - 5: **for all nodes** $i = 1, \dots, n$ **in parallel do**
 - 6: Compute current error: $e_i^{t+1} = e_i^t + \gamma \nabla f_i(x^t) - w_i^t$
 - 7: Compute new local gradient $\nabla f_i(x^{t+1})$
 - 8: Compute error-compensated (stepsize-scaled) gradient $w_i^{t+1} = \mathcal{C}(e_i^{t+1} + \gamma \nabla f_i(x^{t+1}))$
 - 9: Send w_i^{t+1} to the master
 - 10: **end for**
 - 11: **end for**
-

EF14 Theory is Very Limited: Stronger Assumptions than Needed for GD, and Rates are Worse!

Strong and/or Unreasonable Assumptions

- Bounded gradients
- Bounded compression error
- Single node setup
- Homogeneous data regime

Weak Rates

- No linear rate in the strongly convex regime
- Slow $\mathcal{O}(1/T^{2/3})$ sublinear rate in the nonconvex regime

Error Feedback Theory is Very Limited!

Algorithm	sCVX	nCVX	DIST	key limitation
EF Stich et al. [2018]	✓	✗	✓	bounded gradients; sublinear rate in sCVX case
EF-SGD Stich and Karimireddy [2019]	✓	✓	✗	single node only
EF Ajallooeian and Stich [2020]	✓	✓	✗	single node only
SignSGD Karimireddy et al. [2019]	✗	✓	✗	moment bound; single node only
EC-SGD Beznosikov et al. [2020]	✓	✗	✓	linear rate only if $\nabla f_i(x^*) = 0 \forall i$
EC-SGD Gorbunov et al. [2020b]	✓	✗	✓	linear rate only using an extra unbiased compressor
DoubleSqueeze Tang et al. [2020]	✗	✓	✓	bounded compression error; slow $O(1/T^{2/3})$ rate in nCVX case
Qsparse-SGD, CSER Basu et al. [2019], Xie et al. [2020]	✓	✓	✓	bounded gradients; slow $O(1/T^{1/2})$ rate in nCVX case
EC-SGD Koloskova et al. [2020]	✗	✓	✓ [†]	bounded gradients; slow $O(1/T^{2/3})$ rate in nCVX case

Table 1: Known results for first order methods using biased compressors. sCVX = supports strongly convex functions, nCVX= supports nonconvex functions, DIST = works in the distributed regime. [†]decentralized method

Solving All These Issues via Error Feedback 2021 (EF21)

EF21: A New, Simpler, Theoretically Better, and Practically Faster Error Feedback

Peter Richtárik
KAUST*

Igor Sokolov
KAUST

Ilyas Fatkhullin
KAUST[†] & TU Munich

Abstract

Error feedback (EF), also known as error compensation, is an immensely popular convergence stabilization mechanism in the context of distributed training of supervised machine learning models enhanced by the use of contractive communication compression mechanisms, such as Top- k . First proposed by Seide et al. [2014] as a heuristic, EF resisted any theoretical understanding until recently [Stich et al., 2018, Alistarh et al., 2018]. While these early breakthroughs were followed by a steady stream of works offering various improvements and generalizations, the current theoretical understanding of EF is still very limited. Indeed, to the best of our knowledge, all existing analyses either i) apply to the single node setting only, ii) rely on very strong and often unreasonable assumptions, such as global boundedness of the gradients, or iterate-dependent assumptions that cannot be checked a-priori and may not hold in practice, or iii) circumvent these issues via the introduction of additional unbiased compressors, which increase the communication cost. In this work we fix all these deficiencies by proposing and analyzing a new EF mechanism, which we call EF21, which consistently and substantially outperforms EF in practice. Moreover, our theoretical analysis relies on standard assumptions only, works in the distributed heterogeneous data setting, and leads to better and more meaningful rates. In particular, we prove that EF21 enjoys a fast $\mathcal{O}(1/T)$ convergence rate for smooth nonconvex problems, beating the previous bound of $\mathcal{O}(1/T^{2/3})$, which was shown under a strong bounded gradients assumption. We further improve this to a fast linear rate for Polyak-Łojasiewicz functions, which is the first linear convergence result for an error feedback method not relying on unbiased compressors. Since EF has a large number of applications where it reigns supreme, we believe that our 2021 variant, EF21, can have a large impact on the practice of communication efficient distributed learning.

1 Introduction

In order to obtain state-of-the-art performance, modern machine learning models rely on elaborate architectures, need to be trained on data sets of enormous sizes, and involve a very large number of parameters. Some of the most successful models are heavily over-parameterized, which means that they involve more parameters than the number of available training data points [Arora et al., 2018]. Naturally, these circumstances should inform the design of optimization methods that could be most efficient to perform the training.

First, the reliance on sophisticated model architectures, as opposed to simple linear models, generally leads to **nonconvex** optimization problems, which are more challenging than convex problems [Jain and Kar, 2017]. Second, the need for very large training data sizes necessitates the use of **distributed**

*King Abdullah University of Science and Technology, Thuwal, Saudi Arabia.

[†]This paper was written while Ilyas Fatkhullin was an intern at KAUST.



Peter Richtárik, Igor Sokolov and Ilyas Fatkhullin
EF21: A New, Simpler, Theoretically Better, and Practically Faster Error Feedback
NeurIPS 2021



Algorithm 1 EF-BV

proposed method

Input: $x^0, h_1^0, \dots, h_n^0 \in \mathbb{R}^d$,
 $h^0 = \frac{1}{n} \sum_{i=1}^n h_i^0, \gamma > 0$,
 $\lambda \in (0, 1], \nu \in (0, 1]$

for $t = 0, 1, \dots$ **do**

for $i = 1, 2, \dots, n$ in parallel **do**

$$d_i^t := C_i^t(\nabla f_i(x^t) - h_i^t)$$

$$h_i^{t+1} := h_i^t + \lambda d_i^t$$

 send d_i^t to master

end for

 at master:

$$d^t := \frac{1}{n} \sum_{i=1}^n d_i^t$$

$$h^{t+1} := h^t + \lambda d^t$$

$$g^{t+1} := h^t + \nu d^t$$

$$x^{t+1} := \text{prox}_{\gamma R}(x^t - \gamma g^{t+1})$$

 broadcast x^{t+1} to all workers

end for

Algorithm 2 EF21

(Richtárik et al., 2021)

Input: $x^0, h_1^0, \dots, h_n^0 \in \mathbb{R}^d$,
 $h^0 = \frac{1}{n} \sum_{i=1}^n h_i^0, \gamma > 0$,

for $t = 0, 1, \dots$ **do**

for $i = 1, 2, \dots, n$ in parallel **do**

$$d_i^t := C_i^t(\nabla f_i(x^t) - h_i^t)$$

$$h_i^{t+1} := h_i^t + d_i^t$$

 send d_i^t to master

end for

 at master:

$$d^t := \frac{1}{n} \sum_{i=1}^n d_i^t$$

$$h^{t+1} := h^t + d^t$$

$$g^{t+1} := h^t + d^t$$

$$x^{t+1} := \text{prox}_{\gamma R}(x^t - \gamma g^{t+1})$$

 broadcast x^{t+1} to all workers

end for

Algorithm 3 DIANA

(Mishchenko et al., 2019)

Input: $x^0, h_1^0, \dots, h_n^0 \in \mathbb{R}^d$,
 $h^0 = \frac{1}{n} \sum_{i=1}^n h_i^0, \gamma > 0$,
 $\lambda \in (0, 1]$

for $t = 0, 1, \dots$ **do**

for $i = 1, 2, \dots, n$ in parallel **do**

$$d_i^t := C_i^t(\nabla f_i(x^t) - h_i^t)$$

$$h_i^{t+1} := h_i^t + \lambda d_i^t$$

 send d_i^t to master

end for

 at master:

$$d^t := \frac{1}{n} \sum_{i=1}^n d_i^t$$

$$h^{t+1} := h^t + \lambda d^t$$

$$g^{t+1} := h^t + d^t$$

$$x^{t+1} := \text{prox}_{\gamma R}(x^t - \gamma g^{t+1})$$

 broadcast x^{t+1} to all workers

end for

A Whole Bunch of Optional Slides

Towards Error Feedback 2021

Contractive compressor

$$\mathbb{E} \left[\|\mathcal{C}(x) - x\|^2 \right] \leq (1 - \alpha) \|x\|^2$$

$$x^{t+1} = x^t - \gamma \frac{1}{n} \sum_{i=1}^n \mathcal{C}(\nabla f_i(x^t))$$

Distributed CGD can diverge exponentially when used with a contractive compressor!

Example [Beznosikov et al, 2020]:

$n=d=3$, $\mathcal{C} = \text{Top-1}$

Wishful thinking?

A new compressor which “learns” and produces smaller and smaller errors...

$$\lim_{t \rightarrow \infty} \mathbb{E} \left[\|\mathcal{C}_i^t(v_i^t) - v_i^t\|^2 \mid v_i^t \right] \rightarrow 0$$

$$x^{t+1} = x^t - \gamma \frac{1}{n} \sum_{i=1}^n \mathcal{C}_i^t(\nabla f_i(x^t))$$

Attempt 1: Naïve Idea

$$\mathcal{C}^t(v^t) \equiv \mathcal{C}(v^t)$$



There is no reason why the distortion should vanish in the limit

$$\lim_{t \rightarrow \infty} \mathbb{E} \left[\left\| \mathcal{C}^t(v^t) - v^t \right\|^2 \right] \not\rightarrow 0$$

Attempt 2: A Better Idea, but with a Problem

Assumptions:

$$v^* = \lim_{t \rightarrow \infty} v^t$$

v^* is known by the server

$$\mathcal{C}^t(v^t) \equiv v^* + \mathcal{C}(v^t - v^*)$$



Compression error vanishes in the limit!

$$\mathbb{E} \left[\|\mathcal{C}^t(v^t) - v^t\|^2 \mid v^t \right] = \mathbb{E} \left[\|v^* + \mathcal{C}(v^t - v^*) - v^t\|^2 \mid v^t \right] \leq (1 - \alpha) \|v^t - v^*\|^2 \rightarrow 0$$



The limit vector is not known!

Contraction property:

$$\mathbb{E} \left[\|\mathcal{C}(v) - v\|^2 \right] \leq (1 - \alpha) \|v\|^2$$

Attempt 3: Solving the Problem

Assumptions:

$$v^* = \lim_{t \rightarrow \infty} v^t$$

v^* is known by the server

$$\mathcal{C}^t(v^t) \equiv \cancel{v^*} + \mathcal{C}(v^t - \cancel{v^*})$$

$$v^* \approx \mathcal{C}^{t-1}(v^{t-1})$$



$$\mathcal{C}^t(v^t) \equiv \mathcal{C}^{t-1}(v^{t-1}) + \mathcal{C}(v^t - \mathcal{C}^{t-1}(v^{t-1}))$$

Attempt 3: Markov Compressor

Markov property:

$\mathcal{C}^t(v^t)$ depends on $\mathcal{C}^{t-1}(v^{t-1})$

Definition (Markov compressor)

$$\mathcal{C}^0(v^0) \stackrel{\text{def}}{=} \mathcal{C}(v^0)$$

$$\mathcal{C}^t(v^t) \stackrel{\text{def}}{=} \mathcal{C}^{t-1}(v^{t-1}) + \mathcal{C}(v^t - \mathcal{C}^{t-1}(v^{t-1})), \quad t \geq 1$$



Compression error vanishes in the limit!

EF21 = DCGD with Markov Compressor Applied to the Sequence of Gradients

$$x^{t+1} = x^t - \gamma \frac{1}{n} \sum_{i=1}^n \mathcal{C}_i^t(\nabla f_i(x^t))$$

Stepsize

Markov compressor:

$$v_i^t = \nabla f_i(x^t)$$

$$\mathcal{C}_i^0(v_i^0) \stackrel{\text{def}}{=} \mathcal{C}(v_i^0)$$

$$\mathcal{C}_i^t(v_i^t) \stackrel{\text{def}}{=} \mathcal{C}_i^{t-1}(v_i^{t-1}) + \mathcal{C}(v_i^t - \mathcal{C}_i^{t-1}(v_i^{t-1})), \quad t \geq 1$$

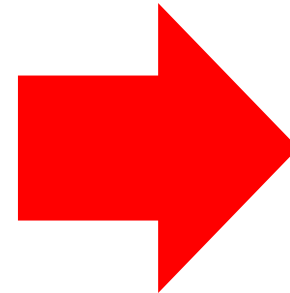
Theory of EF21

Restricted Equivalence of EF21 and EF14

Theorem (NEW)

If $\mathcal{C}$ is

- 1 deterministic
- 2 positively homogeneous
 $\mathcal{C}(tx) = t\mathcal{C}(x)$ for $t \geq 0$
- 3 additive
 $\mathcal{C}(x + y) = \mathcal{C}(x) + \mathcal{C}(y)$



$$\mathbf{EF21 = EF14}$$

Convergence of EF21: Summary of Results

Standard assumptions!

Strong $\mathcal{O}(1/T)$
sublinear rate!

Assumptions	Complexity	Theorem
f_i is L_i -smooth f is lower bounded by $f^{\inf}$	$\mathbb{E} \left[\ \nabla f(\hat{x}^T)\ ^2 \right] \leq \frac{2(f(x^0) - f^{\inf})}{\gamma T} + \frac{\mathbb{E}[G^0]}{\theta T}$	1
f_i is L_i -smooth f is lower bounded by $f^{\inf}$ f satisfies PL condition	$\mathbb{E} [\Psi^T] \leq (1 - \gamma\mu)^T \mathbb{E} [\Psi^0]$	2

Table 2: Summary of complexity results obtained in this paper. Quantities: μ = PL constant; γ = stepsize; G^0 = see (14); Ψ^t = Lyapunov function defined in Theorem 2.

First linear rate for
Error Feedback!

Convergence of EF21: General Nonconvex Case

Theorem (NEW)

Iterate chosen from $\{x^0, x^1, \dots, x^{T-1}\}$
uniformly at random

$$f^{\text{inf}} \stackrel{\text{def}}{=} \inf_{x \in \mathbb{R}^d} f(x)$$

$$G^t \stackrel{\text{def}}{=} \frac{1}{n} \sum_{i=1}^n \|g_i^t - \nabla f_i(x^t)\|^2$$

$$\mathbb{E} \left[\|\nabla f(\hat{x}^T)\|^2 \right] \leq \frac{2 \left(f(x^0) - f^{\text{inf}} \right)}{\gamma T} + \frac{\mathbb{E} [G^0]}{\theta T}$$

Stepsize:

$$0 < \gamma \leq \left(L + \left(\frac{1}{n} \sum_{i=1}^n L_i^2 \right)^{1/2} \cdot \left(\frac{(1 + \sqrt{1 - \alpha})}{\alpha} - 1 \right) \right)^{-1}$$

$$\begin{aligned} \|\nabla f(x) - \nabla f(y)\| &\leq L\|x - y\| \\ \|\nabla f_i(x) - \nabla f_i(y)\| &\leq L_i\|x - y\| \end{aligned}$$

$$\mathbb{E} \left[\|\mathcal{C}(v) - v\|^2 \right] \leq (1 - \alpha) \|v\|^2$$

$$1 - \sqrt{1 - \alpha}$$

Convergence of EF21: General Nonconvex Case

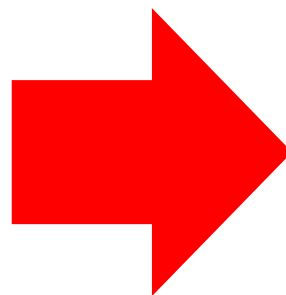
Corollary

Assuming that $g_i^0 = \nabla f_i(x^0)$ for all i

$$\tilde{L} \stackrel{\text{def}}{=} \left(\frac{1}{n} \sum_{i=1}^n L_i^2 \right)^{1/2}$$

$$\mathbb{E} \left[\|\mathcal{C}(v) - v\|^2 \right] \leq (1 - \alpha) \|v\|^2$$

$$T \geq \frac{2 \left(f(x^0) - f^{\text{inf}} \right) \left(L + \tilde{L} \left(\frac{1 + \sqrt{1 - \alpha}}{\alpha} - 1 \right) \right)}{\varepsilon}$$



$$\mathbb{E} \left[\|\nabla f(\hat{x}^T)\|^2 \right] \leq \varepsilon$$

For $\alpha = 1$ (no compression), we recover the rate of Gradient Descent!

EF21 Experiments: Problem and Data

$$f(x) = \frac{1}{n} \sum_{i=1}^n \log \left(1 + \exp \left(-y_i a_i^\top x \right) \right) + \lambda \sum_{j=1}^d \frac{x_j^2}{1 + x_j^2}$$

Dataset	n	N (total # of datapoints)	d (# of features)	N_i (# of datapoints per client)
phishing	20	11,055	68	552
mushrooms	20	8,120	112	406
a9a	20	32,560	123	1,628
w8a	20	49,749	300	2,487

Table 3: Summary of the datasets and splitting of the data among clients.

EF21 and EF21+ Tolerate Higher Stepsizes than EF14

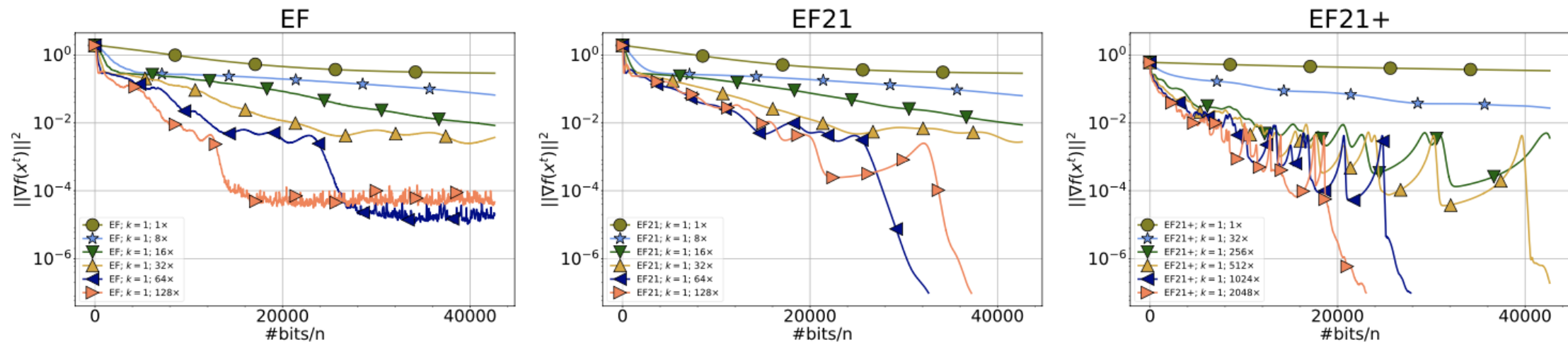


Figure 1: The performance of EF, EF21, and EF21+ with Top-1 compressor, and for increasing stepsizes. Representative dataset used: a9a. By 1 $\times$, 2 $\times$, 4 $\times$ (and so on) we indicate that the stepsize was set to a multiple of the largest stepsize predicted by our theory.

EF14, EF21 and EF21+ with Best Fixed Stepsizes

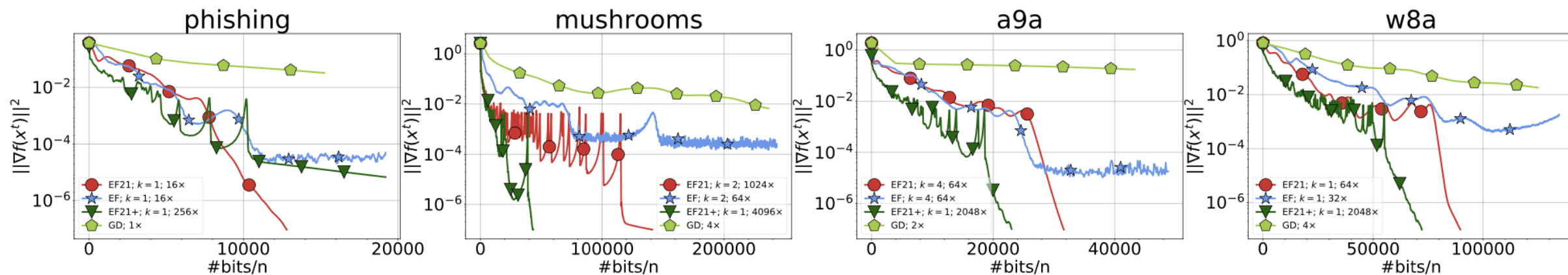


Figure 2: Comparison of EF21, EF21+ to EF with Top- k for individually fine-tuned k and fine-tuned stepsizes for all methods.



EF21: Extensions & Further Reading

EF21: Extensions & Further Reading 1

6/2021



P.R., Igor Sokolov and Ilyas Fatkhullin

EF21: A new, simpler, theoretically better, and practically faster error feedback
NeurIPS 2021

7/2021

EF21-PP, EF21-BC,
EF21-SGD, ...



Ilyas Fatkhullin, Igor Sokolov, Eduard Gorbunov, Zhize Li and P.R.

EF21 with bells & whistles: practical algorithmic extensions of modern error feedback
JMLR 2025

2/2022



P.R., Igor Sokolov, Ilyas Fatkhullin, Elnur Gasanov, Zhize Li and Eduard Gorbunov

3PC: Three point compressors for communication-efficient distributed training and a better theory for lazy aggregation
ICML 2022

5/2022



Laurent Condat, Kai Yi and P.R.

EF-BV: A unified theory of error feedback and variance reduction mechanisms for biased and unbiased compression in distributed optimization
NeurIPS 2022

9/2022



Kaja Grunkowska, Alexander Tyurin and P.R.

EF21-P and friends: Improved theoretical communication complexity for distributed optimization with bidirectional compression
ICML 2023

*6 extensions of EF21,
previously not possible for EF14
(e.g., partial participation, prox,
bidirectional compression,
momentum)*

- *Tight **generalization of contractive compressors & EF21***
- *Explains why **lazy aggregation** works*

- ***Unification of EF21 & DIANA***
- ***New class of compressors (2 parameters instead of 1) generalizing contractive & unbiased compressors***

EF21: Extensions & Further Reading 2

9/2022



Kaja Gruntkowska, Alexander Tyurin and P.R.
EF21-P and friends: Improved theoretical communication complexity for distributed optimization with bidirectional compression
ICML 2023

- *Discovery that EF14 should be run on the server to compress models (and this is equiv to EF21 after a change of variables)*
- *GD with bidirectional compression can beat GD in theoretical comm complexity*

11/2022



Maksim Makarenko, Elnur Gasanov, Rustem Islamov, Abdurakhmon Sadiev, P.R.
Adaptive compression for communication-efficient distributed training
TMLR 2023

EF21-P + EF21 used to fine-tune foundation models in practice

2023



Jue Wang, Yucheng Lu, Binhang Yuan, Beidi Chen, Percy Liang, Christopher De Sa, Christopher Re, Ce Zhang
CocktailSGD: Fine-tuning Foundation Models over 500Mbps Networks
ICML 2023

Momentum provably helps EF21 when one uses stoch gradients rather than gradients!

5/2023



Ilyas Fatkhullin, Alexander Tyurin and P.R.
Momentum provably improves error feedback!
NeurIPS 2023

EF21: Extensions & Further Reading 3

10/2023



Ahmad Rammal, Kaja Gruntkowska, Nikita Fedin, Eduard Gorbunov, P.R.
Communication compression for Byzantine robust learning: New efficient algorithms and improved rates
AISTATS 2024

2/2024



P.R., Elnur Gasanov, Konstantin Burlachenko
Error feedback reloaded: From quadratic to arithmetic mean of smoothness constants
ICLR 2024

*Improved theory for
EF21*

10/2024



Sarit Khirirat, Abdurakhmon Sadiev, Artem Riabinin, Eduard Gorbunov, P.R.
Error feedback under (L_0, L_1) -smoothness: normalization and momentum
NeurIPS 2025

*EF21 for
nonsmooth
problems requires
normalization*

5/2025

Safe-EF



Rustem Islamov, Yarden As, Ilyas Fatkhullin
Safe-EF: Error Feedback for Nonsmooth Constrained Optimization

EF21: Extensions & Further Reading 4

10/2025



Yuan Gao, Anton Rodomanov, Jeremy Rack, Sebastian U. Stich
Composite Optimization with Error Feedback: the Dual Averaging Approach

10/2025

EF21-Muon



Kaja Grunkowska, Alexander Gaponov, Zhirayr Tovmasyan, P.R.
Error feedback for Muon and friends
ICLR 2026

*1st error-feedback
for Muon*

11/2025

||EF21-SGDM||



Abdurakhmon Sadiev, Yury Demidovich, Igor Sokolov, Grigory Malinovsky, Sarit Khirirat, and P.R.
Improved convergence in parameter-agnostic error feedback through momentum

What About the Nonconvex Regime?

MARINA



Eduard Gorbunov, Konstantin Burlachenko, Zhize Li and P.R.
MARINA: Faster non-convex distributed learning with compression
ICML 2021



Rafał Szlendak, Alexander Tyurin and P.R.
Permutation compressors for provably faster distributed nonconvex optimization
ICLR 2022

Smooth Nonconvex Regime

- EF21 has bad guarantees
 - no better than GD
- DCGD has good guarantees
 - better than GD in the interpolation regime
- DIANA has very good guarantees
 - better than GD, no need for interpolation regime
- **MARINA has even better guarantees!**

Federated Training of a Supervised Machine Learning Model

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

devices /
machines

model parameters / features

Loss on local data $\mathcal{D}_i$ stored on device i

$$f_i(x) = \mathbb{E}_{\xi \sim \mathcal{D}_i} f_{i,\xi}(x)$$

The datasets $\mathcal{D}_1, \dots, \mathcal{D}_n$ can be arbitrarily heterogeneous

Assumptions: Very Weak!

A1

The gradient of f_i is L_i -Lipschitz, i.e.,

$$\|\nabla f_i(x) - \nabla f_i(y)\| \leq L_i \|x - y\| \quad \forall x, y \in \mathbb{R}^d$$

A2

$f = \frac{1}{n} \sum_{i=1}^n f_i$ is lower bounded, i.e.,

$$\inf_{x \in \mathbb{R}^d} f(x) \stackrel{\text{def}}{=} f^{\text{inf}} > -\infty$$

Goal: Find an Approximately Stationary Point in Expectation

Find a (possibly random) vector $x \in \mathbb{R}^d$ such that

Expectation w.r.t.
the randomness of
the algorithm



$$\mathbb{E} \left[\left\| \nabla f(x) \right\|^2 \right] \leq \varepsilon$$

with minimal communication



bits transferred from devices to the orchestrating server

MARINA: Current SOTA in Theoretical Communication Complexity

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

$$x^{t+1} = x^t - \gamma g^t$$

$$g^t = \frac{1}{n} \sum_{i=1}^n g_i^t$$

$$g_i^{t+1} = \begin{cases} \nabla f_i(x^{t+1}) & \text{with probability } p \\ g^t + \mathcal{C}_i(\nabla f_i(x^{t+1}) - \nabla f_i(x^t)) & \text{with probability } 1 - p \end{cases}$$

Unbiased compression operator $\mathcal{C}_i : \mathbb{R}^d \rightarrow \mathbb{R}^d$

$$\mathbb{E} [\mathcal{C}_i(v)] = v \quad \forall v \in \mathbb{R}^d$$

$$\mathbb{E} [\|\mathcal{C}_i(v) - v\|^2] \leq \omega_i \|v\|^2 \quad \forall v \in \mathbb{R}^d$$



Eduard Gorbunov, Konstantin Burlachenko, Zhize Li and P.R.
MARINA: Faster non-convex distributed learning with compression
ICML 2021

1-Slide Detour:

DASHA: Addressing Some Issues with MARINA

- MARINA does not apply to infinite-sum problems, **DASHA does!**
- MARINA occasionally sends non-compressed messages, **DASHA does not!**
- MARINA works also when stochastic gradients are used instead of gradients, but **DASHA has better communication complexity in this regime!**



Alexander Tyurin and P.R.

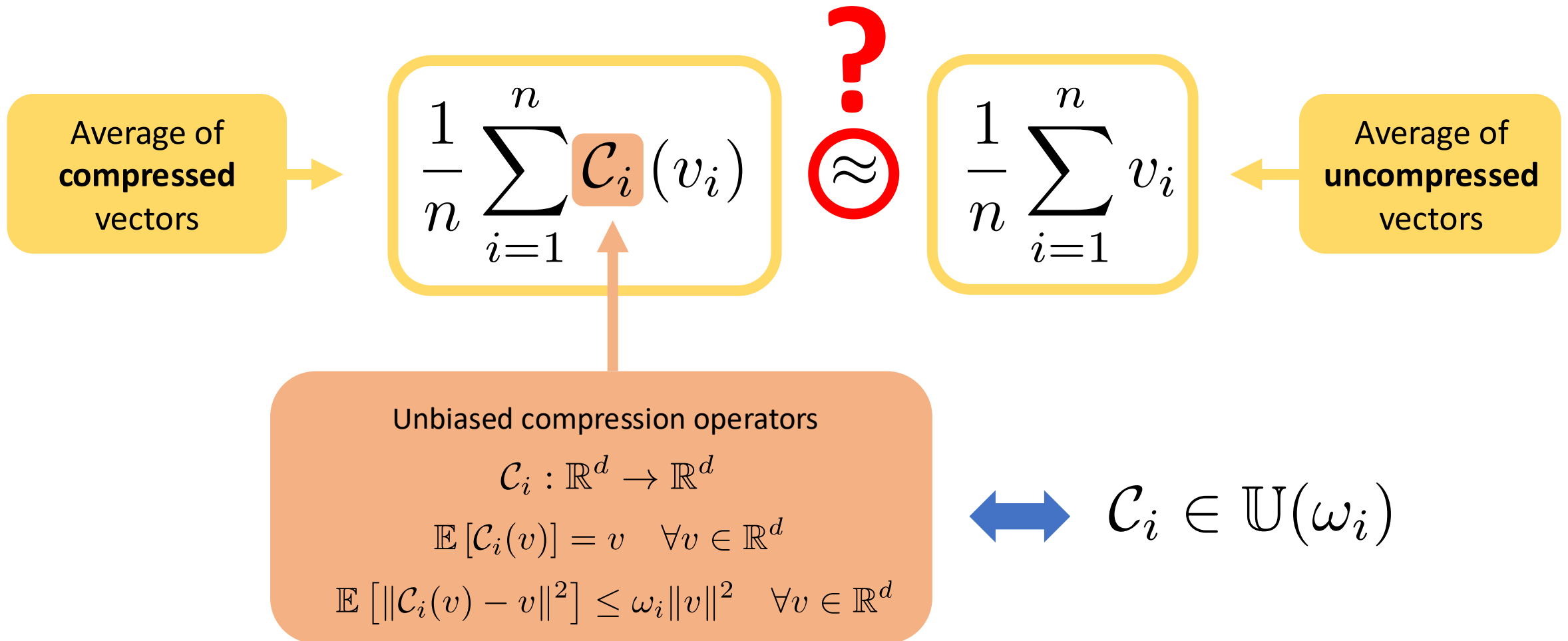
DASHA: Distributed nonconvex optimization with communication compression and optimal oracle complexity
ICLR 2023



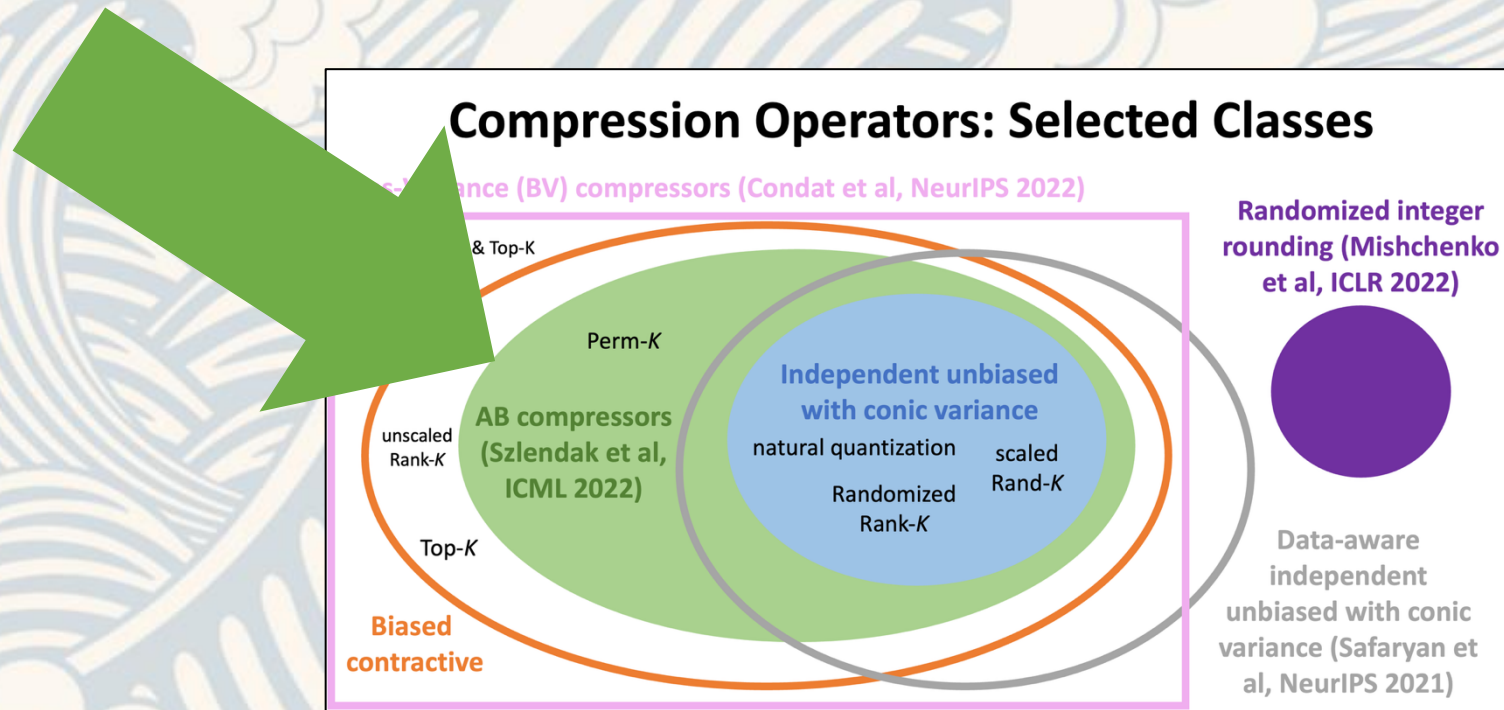
Alexander Tyurin and P.R.

A computation and communication efficient method for distributed nonconvex problems in the partial participation setting
NeurIPS 2023

Aggregation by Averaging Compressed Messages



Joint Design of Compression Operators



Two Compressor Design Principles

STANDARD LITERATURE

Design the compressor

$$\mathcal{C}_i : \mathbb{R}^d \rightarrow \mathbb{R}^d$$

used on device i

Independently or greedily
of all other devices

PROPOSED APPROACH

Design the compressor

$$\mathcal{C}_i : \mathbb{R}^d \rightarrow \mathbb{R}^d$$

used on device i

jointly
with all other devices

**Key Weakness of Independent
Compressors: Bad Performance
When Aggregating Similar Vectors**

Aggregation of Identical Vectors Using Rand1

$$\begin{pmatrix} 4 \\ -7 \\ 2 \end{pmatrix}$$

v_1



$$\begin{pmatrix} 4 \\ -7 \\ 2 \end{pmatrix}$$

v_2



$$\begin{pmatrix} 4 \\ -7 \\ 2 \end{pmatrix}$$

v_3



Aggregation of Identical Vectors Using Rand1

$$3 \times \begin{pmatrix} 4 \\ 0 \\ 0 \end{pmatrix}$$

$\mathcal{C}_1(v_1)$



$$3 \times \begin{pmatrix} 0 \\ -7 \\ 0 \end{pmatrix}$$

$\mathcal{C}_2(v_2)$



$$3 \times \begin{pmatrix} 0 \\ -7 \\ 0 \end{pmatrix}$$

$\mathcal{C}_3(v_3)$



Aggregation of Identical Vectors Using Rand1

$$\frac{1}{3} \sum_{i=1}^3 \mathcal{C}_i(v_i) = \begin{pmatrix} 4 \\ -14 \\ 0 \end{pmatrix} \neq \begin{pmatrix} 4 \\ -7 \\ 2 \end{pmatrix} = \frac{1}{3} \sum_{i=1}^3 v_i$$

The average of the compressed vectors **is not equal** to the average of the original vectors!

Trick: Design the Compressors Jointly!

Aggregation of Identical Vectors Using The Permutation Sparsifier Perm-1

$$\begin{pmatrix} 4 \\ -7 \\ 2 \end{pmatrix}$$

v_1



$$\begin{pmatrix} 4 \\ -7 \\ 2 \end{pmatrix}$$

v_2

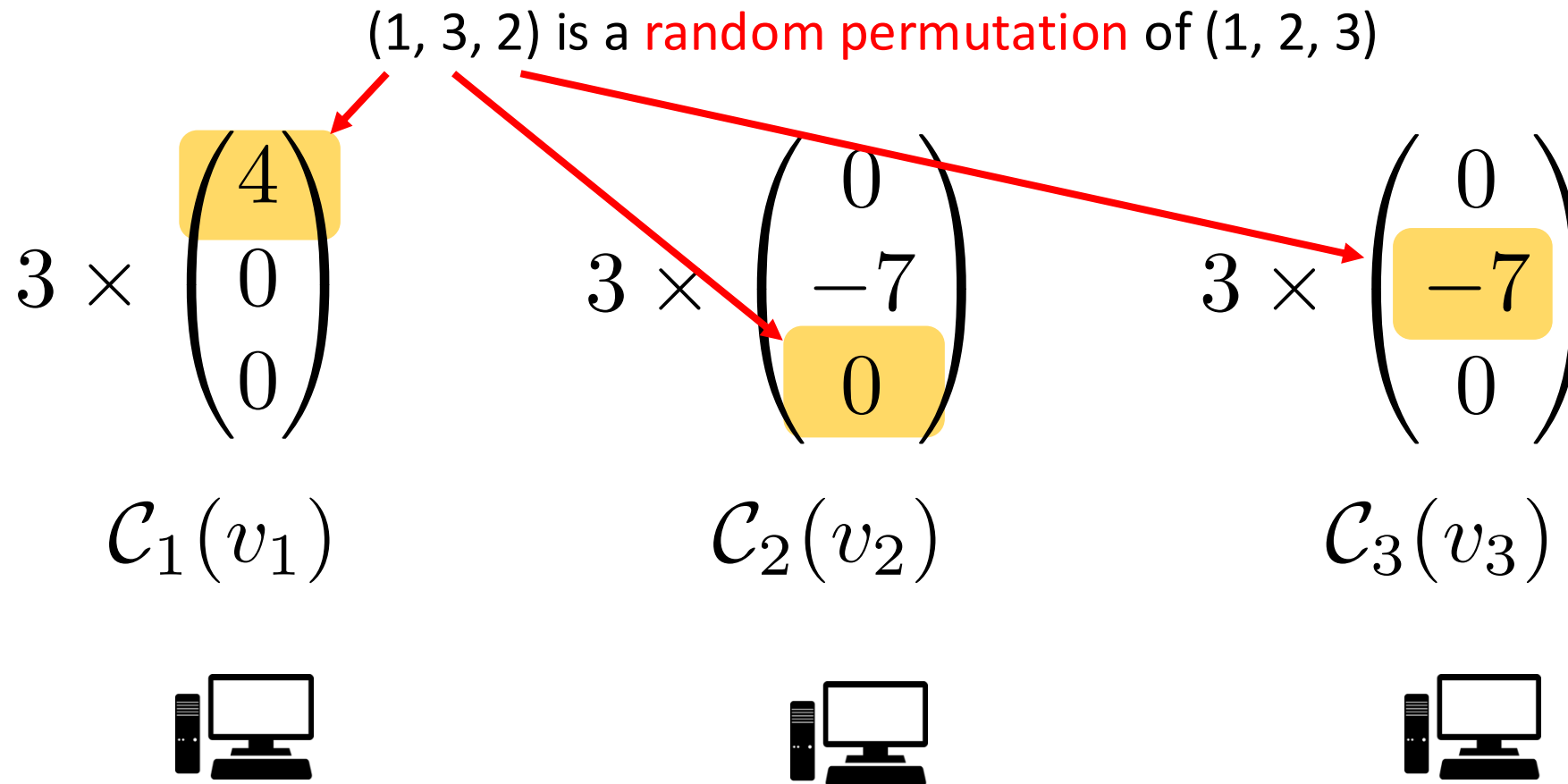


$$\begin{pmatrix} 4 \\ -7 \\ 2 \end{pmatrix}$$

v_3



Aggregation of Identical Vectors Using The Permutation Sparsifier Perm-1



Aggregation of Identical Vectors Using Perm-1

$$\frac{1}{3} \sum_{i=1}^3 \mathcal{C}_i(v_i) = \begin{pmatrix} 4 \\ -7 \\ 2 \end{pmatrix} = \begin{pmatrix} 4 \\ -7 \\ 2 \end{pmatrix} = \frac{1}{3} \sum_{i=1}^3 v_i$$

The average of the compressed vectors **is equal** to the average of the original vectors!

From Intuition to Formalism

$$\{\mathcal{C}_i\}_{i=1}^n \in \mathbb{U}(A, B)$$

New Concept: AB Inequality

Compressors $\mathcal{C}_1, \dots, \mathcal{C}_n$ satisfy the AB inequality if

$$\mathbb{E} \left[\left\| \frac{1}{n} \sum_{i=1}^n \mathcal{C}_i(v_i) - \frac{1}{n} \sum_{i=1}^n v_i \right\|^2 \right] \leq A \left(\frac{1}{n} \sum_{i=1}^n \|v_i\|^2 \right) - B \left\| \frac{1}{n} \sum_{i=1}^n v_i \right\|^2$$

clients

for all $v_1, \dots, v_n \in \mathbb{R}^d$

Key parameters capturing the **joint quality**
of the collection of compression operators

$$\mathcal{C}_1, \dots, \mathcal{C}_n : \mathbb{R}^d \rightarrow \mathbb{R}^d$$

Observation 1

$$\mathcal{C} \stackrel{\text{def}}{=} \frac{1}{n} \sum_{i=1}^n \mathcal{C}_i$$

Follows from the AB inequality by choosing
 $v_1 = v_2 = \dots = v_n = v$

If the AB inequality holds, then

$$\mathbb{E} \left[\|\mathcal{C}(v) - v\|^2 \right] \leq (A - B) \|v\|^2 \quad \forall v \in \mathbb{R}^d$$

AB Inequality

$$\mathbb{E} \left[\left\| \frac{1}{n} \sum_{i=1}^n \mathcal{C}_i(v_i) - \frac{1}{n} \sum_{i=1}^n v_i \right\|^2 \right] \leq A \left(\frac{1}{n} \sum_{i=1}^n \|v_i\|^2 \right) - B \left\| \frac{1}{n} \sum_{i=1}^n v_i \right\|^2$$

Observation 2

Follows from
Observation 1
by choosing
 $v \neq 0$



If the AB inequality holds, then

$$A \geq B$$

AB Inequality

$$\mathbb{E} \left[\left\| \frac{1}{n} \sum_{i=1}^n \mathcal{C}_i(v_i) - \frac{1}{n} \sum_{i=1}^n v_i \right\|^2 \right] \leq A \left(\frac{1}{n} \sum_{i=1}^n \|v_i\|^2 \right) - B \left\| \frac{1}{n} \sum_{i=1}^n v_i \right\|^2$$

New Concept: Input Variance Compressors

$$\bar{v} \stackrel{\text{def}}{=} \frac{1}{n} \sum_{i=1}^n v_i$$

$\mathcal{C}_1, \dots, \mathcal{C}_n$ are input variance compressors if

the AB inequality holds with $A = B$

$$\mathbb{E} \left[\left\| \frac{1}{n} \sum_{i=1}^n \mathcal{C}_i(v_i) - \frac{1}{n} \sum_{i=1}^n v_i \right\|^2 \right] \leq C \cdot \left(\frac{1}{n} \sum_{i=1}^n \|v_i - \bar{v}\|^2 \right)$$

AB Inequality

$\text{Var}(v_1, \dots, v_n)$

$$\mathbb{E} \left[\left\| \frac{1}{n} \sum_{i=1}^n \mathcal{C}_i(v_i) - \frac{1}{n} \sum_{i=1}^n v_i \right\|^2 \right] \leq A \left(\frac{1}{n} \sum_{i=1}^n \|v_i\|^2 \right) - B \left\| \frac{1}{n} \sum_{i=1}^n v_i \right\|^2$$

Examples

Table 1: Examples of compressors $\{\mathcal{C}_i\}_{i=1}^n \in \mathbb{U}(A, B)$. See the appendix for many more.

Compressors	A	B	Calculation of A, B	Reference
$\mathcal{C}_i \in \mathbb{U}(\omega_i)$	$\max_i \omega_i$	0	Lemma 1	standard
$\mathcal{C}_i \in \mathbb{U}(\omega_i)$, independent	$\frac{1}{n} \max_i \omega_i$	0	Lemma 1	standard
PermK ($d \geq n$); Def 2	1	1	Theorem 1	new
PermK ($d \leq n$); Def 3	$1 - \frac{n-d}{n-1}$	$1 - \frac{n-d}{n-1}$	Theorem 2	new

Hessian Variance

Three Smoothness Parameters

1

Bound on **gradient differences**

$$\|\nabla f_i(x) - \nabla f_i(y)\|^2 \leq L_i^2 \|x - y\|^2$$

∇f_i is L_i -Lipschitz

2

Bound on the **2nd moment of gradient differences**

$$\frac{1}{n} \sum_{i=1}^n \|\nabla f_i(x) - \nabla f_i(y)\|^2 \leq L_+^2 \|x - y\|^2$$

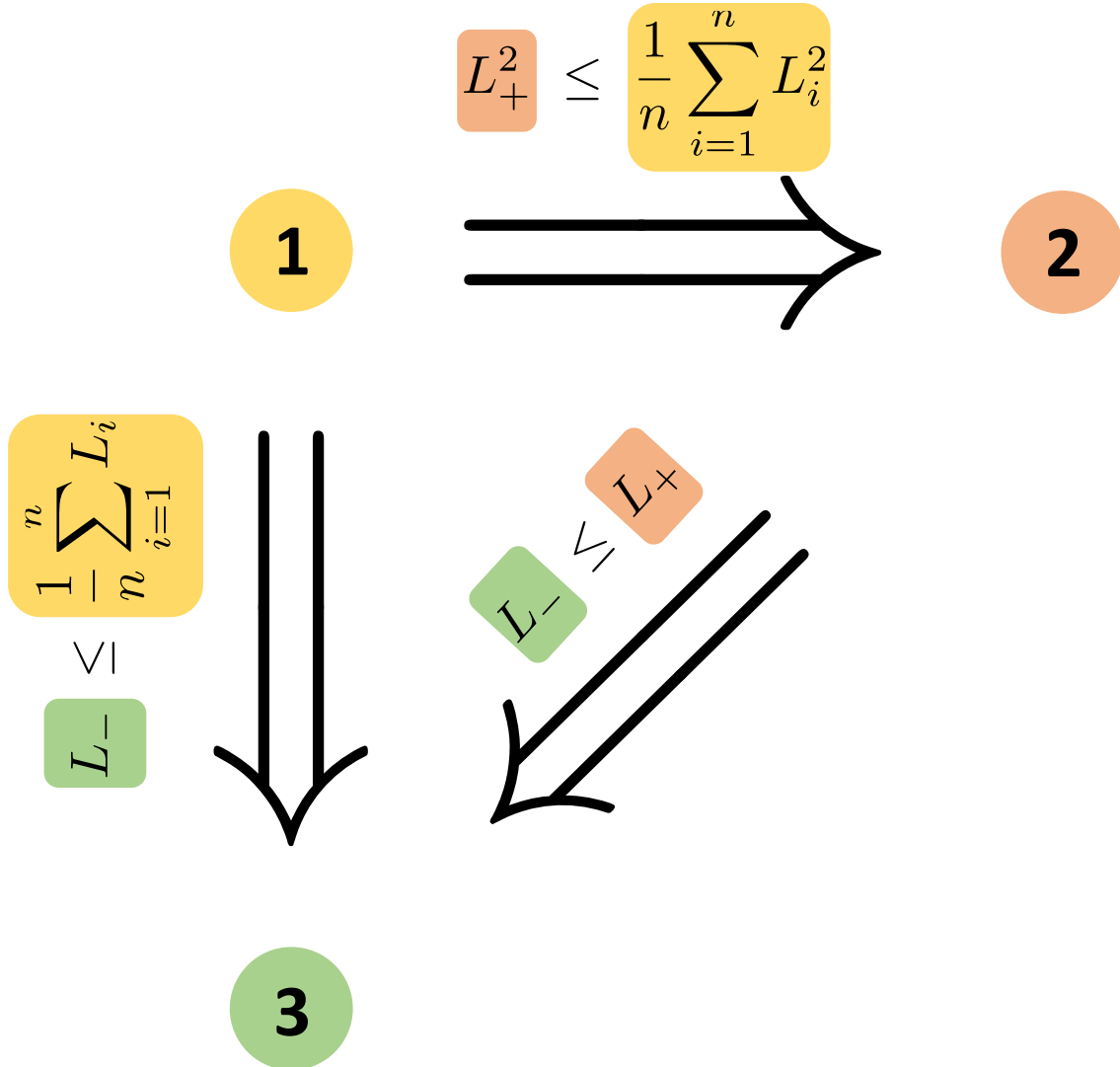
3

Bound on the **mean of gradient differences**

$$\left\| \frac{1}{n} \sum_{i=1}^n \nabla f_i(x) - \frac{1}{n} \sum_{i=1}^n \nabla f_i(y) \right\|^2 \leq L_-^2 \|x - y\|^2$$

∇f is L_- -Lipschitz

Three Smoothness Parameters: Relationships



1

Bound on **gradient differences**

$$\|\nabla f_i(x) - \nabla f_i(y)\|^2 \leq L_i^2 \|x - y\|^2$$

2

Bound on the **2nd moment of gradient differences**

$$\frac{1}{n} \sum_{i=1}^n \|\nabla f_i(x) - \nabla f_i(y)\|^2 \leq L_+^2 \|x - y\|^2$$

3

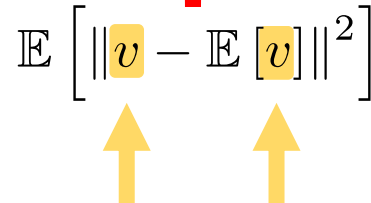
Bound on the **mean of gradient differences**

$$\left\| \frac{1}{n} \sum_{i=1}^n \nabla f_i(x) - \frac{1}{n} \sum_{i=1}^n \nabla f_i(y) \right\|^2 \leq L_-^2 \|x - y\|^2$$

Hessian Variance

Bound on the **variance of gradient differences**

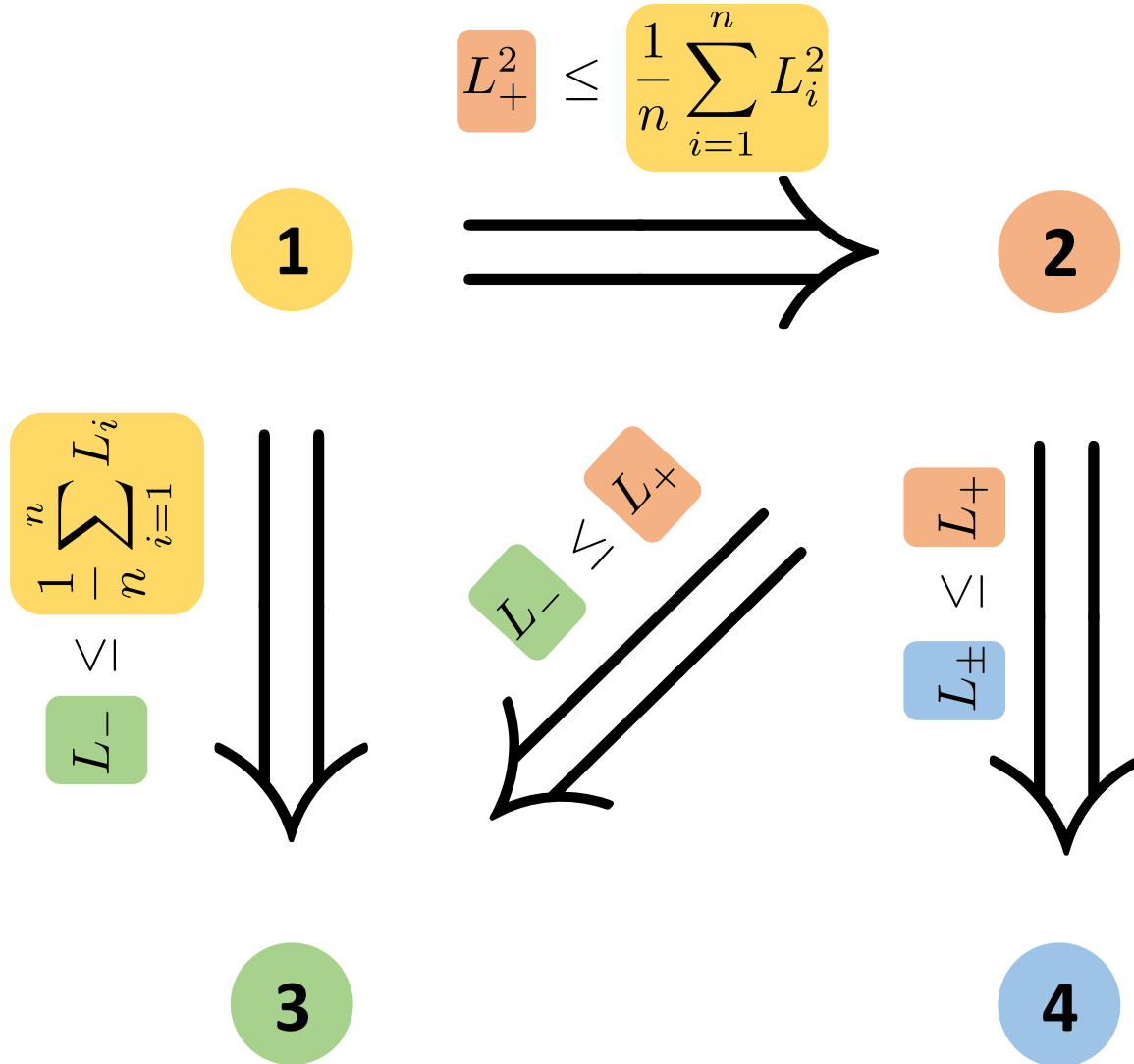
$$\frac{1}{n} \sum_{i=1}^n \|\nabla f_i(x) - \nabla f_i(y)\|^2 - \left\| \frac{1}{n} \sum_{i=1}^n \nabla f_i(x) - \frac{1}{n} \sum_{i=1}^n \nabla f_i(y) \right\|^2 \leq L_{\pm}^2 \|x - y\|^2$$


$$\mathbb{E} \left[\|v - \mathbb{E}[v]\|^2 \right]$$

v is a random vector defined as follows: $v = v_i$ with probability $\frac{1}{n}$

$$v_i \stackrel{\text{def}}{=} \nabla f_i(x) - \nabla f_i(y)$$

Four Smoothness Parameters



1	Bound on gradient differences $\ \nabla f_i(x) - \nabla f_i(y)\ ^2 \leq L_i^2 \ x - y\ ^2$
2	Bound on the 2nd moment of gradient differences $\frac{1}{n} \sum_{i=1}^n \ \nabla f_i(x) - \nabla f_i(y)\ ^2 \leq L_+^2 \ x - y\ ^2$
3	Bound on the mean of gradient differences $\left\ \frac{1}{n} \sum_{i=1}^n \nabla f_i(x) - \frac{1}{n} \sum_{i=1}^n \nabla f_i(y) \right\ ^2 \leq L_-^2 \ x - y\ ^2$
4	Bound on the variance of gradient differences $\frac{1}{n} \sum_{i=1}^n \ \nabla f_i(x) - \nabla f_i(y)\ ^2 - \left\ \frac{1}{n} \sum_{i=1}^n \nabla f_i(x) - \frac{1}{n} \sum_{i=1}^n \nabla f_i(y) \right\ ^2 \leq L_{\pm}^2 \ x - y\ ^2$

Hessian Variance can be Small

Table 2: Value of $L_{\pm}^2$ in cases when $f_i(x) = \phi(x) + \phi_i(x)$, where $\phi : \mathbb{R}^d \rightarrow \mathbb{R}$ is an arbitrary differentiable function and $\phi_i : \mathbb{R}^d \rightarrow \mathbb{R}$ is twice continuously differentiable. The matrices $\mathbf{A}_i \in \mathbb{R}^{d \times d}$ are assumed (without loss of generality) to be symmetric. The matrix-valued function $\mathbf{L}_{\pm}(x, y)$ is defined in Theorem 3.

$\phi(x)$	$\phi_i(x)$	Hessian variance $L_{\pm}^2$
any	0	0
any	$b_i^{\top} x + c_i$	0
0	$\frac{1}{2} x^{\top} \mathbf{A}_i x + b_i^{\top} x + c_i$	$\lambda_{\max} \left(\frac{1}{n} \sum_{i=1}^n \mathbf{A}_i^2 - \left(\frac{1}{n} \sum_{i=1}^n \mathbf{A}_i \right)^2 \right)$
0	smooth	$\sup_{x, y \in \mathbb{R}^d, x \neq y} \frac{(x-y)^{\top} \mathbf{L}_{\pm}(x, y)(x-y)}{\ x-y\ ^2}$

Hessian Variance: Second Order Characterization

Hessian variance
parameter

$L_{\pm}^2$

$= \sup_{x, y \in \mathbb{R}^d, x \neq y}$

$$\frac{(x - y)^{\top} \mathbf{L}_{\pm}(x, y)(x - y)}{\|x - y\|^2}$$

“Variance” of the (average) Hessians

$$\mathbf{L}_{\pm}(x, y) \stackrel{\text{def}}{=} \frac{1}{n} \sum_{i=1}^n \mathbf{H}_i^2(x, y) - \left(\frac{1}{n} \sum_{i=1}^n \mathbf{H}_i(x, y) \right)^2$$

Average Hessian of f_i between x and y

$$\mathbf{H}_i(x, y) \stackrel{\text{def}}{=} \int_0^1 \nabla^2 f_i(x + t(y - x)) dt$$

Hessian Variance: Second Order Characterization

Theorem 3. Assume that for each $i \in \{1, 2, \dots, n\}$, the function f_i is twice continuously differentiable. Fix any $x, y \in \mathbb{R}^d$ and define⁹

$$\mathbf{H}_i(x, y) := \int_0^1 \nabla^2 f_i(x + t(y - x)) dt, \quad \mathbf{H}(x, y) := \frac{1}{n} \sum_{i=1}^n \mathbf{H}_i(x, y). \quad (9)$$

Then the matrices $\mathbf{L}_i(x, y) := \mathbf{H}_i^2(x, y)$, $\mathbf{L}_-(x, y) := \mathbf{H}^2(x, y)$, $\mathbf{L}_+(x, y) := \frac{1}{n} \sum_{i=1}^n \mathbf{H}_i^2(x, y)$ and $\mathbf{L}_\pm(x, y) := \mathbf{L}_+(x, y) - \mathbf{L}_-(x, y)$ are symmetric and positive semidefinite. Moreover,

$$L_i^2 = \sup_{x, y \in \mathbb{R}^d, x \neq y} \frac{(x-y)^\top \mathbf{L}_i(x, y)(x-y)}{\|x-y\|^2},$$

$$L_-^2 = \sup_{x, y \in \mathbb{R}^d, x \neq y} \frac{(x-y)^\top \mathbf{L}_-(x, y)(x-y)}{\|x-y\|^2},$$

$$L_+^2 = \sup_{x, y \in \mathbb{R}^d, x \neq y} \frac{(x-y)^\top \mathbf{L}_+(x, y)(x-y)}{\|x-y\|^2},$$

$$L_\pm^2 = \sup_{x, y \in \mathbb{R}^d, x \neq y} \frac{(x-y)^\top \mathbf{L}_\pm(x, y)(x-y)}{\|x-y\|^2}.$$

Theoretical Analysis

A Refined and Generalized Convergence Analysis of MARINA

$$T \geq \frac{2(f(x^0) - f^{\inf})}{\varepsilon} \left(L_- + \sqrt{\frac{1-p}{p}} \left((A-B)L_+^2 + BL_{\pm}^2 \right) \right)$$

Hessian variance

AB inequality constants

$$\mathbb{E} \left[\left\| \frac{1}{n} \sum_{i=1}^n \mathcal{C}_i(v_i) - \frac{1}{n} \sum_{i=1}^n v_i \right\|^2 \right] \leq A \left(\frac{1}{n} \sum_{i=1}^n \|v_i\|^2 \right) - B \left\| \frac{1}{n} \sum_{i=1}^n v_i \right\|^2$$

$$\mathbb{E} \left[\left\| \nabla f(\hat{x}^T) \right\|^2 \right] \leq \varepsilon$$

Key: $L_{\pm} \leq L_+$

1

Bound on gradient differences

$$\|\nabla f_i(x) - \nabla f_i(y)\|^2 \leq L_i^2 \|x - y\|^2$$

2

Bound on the 2nd moment of gradient differences

$$\frac{1}{n} \sum_{i=1}^n \|\nabla f_i(x) - \nabla f_i(y)\|^2 \leq L_+^2 \|x - y\|^2$$

3

Bound on the mean of gradient differences

$$\left\| \frac{1}{n} \sum_{i=1}^n \nabla f_i(x) - \frac{1}{n} \sum_{i=1}^n \nabla f_i(y) \right\|^2 \leq L_-^2 \|x - y\|^2$$

4

Bound on the variance of gradient differences

$$\frac{1}{n} \sum_{i=1}^n \|\nabla f_i(x) - \nabla f_i(y)\|^2 - \left\| \frac{1}{n} \sum_{i=1}^n \nabla f_i(x) - \frac{1}{n} \sum_{i=1}^n \nabla f_i(y) \right\|^2 \leq L_{\pm}^2 \|x - y\|^2$$

A Refined and Generalized Convergence Analysis of MARINA

Input variance compressors remove the dependence on L_+ !
Get further boost when Hessian variance $L_{\pm}$ is small!

MARINA with Rand-1 vs Perm-1 ($d > n$)

Approach	Compressors	A	B	$(A - B)L_+^2 + BL_\pm^2$
Gorbunov et al 2021	Rand1	$\frac{d-1}{n}$	0	$\frac{d-1}{n}L_+^2$
THIS WORK	Perm1	1	1	$L_\pm^2$

$L_\pm^2$ can be much much smaller than $\frac{d-1}{n}L_+^2$

Optimized Communication Complexity of MARINA with our Perm- K Compressors

Table 4: Optimized communication complexity of MARINA and EF21 with particular compressors.

Method + Compressor	Communication Complexity	
	$d \geq n$ (Lemma 13)	$d \leq n$ (Lemma 14)
MARINA $\cap$ Perm K	$\mathcal{O}\left(\frac{\Delta_0}{\varepsilon} \min\left\{dL_-, \frac{d}{n}L_- + \frac{d}{\sqrt{n}}L_{\pm}\right\}\right)$	$\mathcal{O}\left(\frac{\Delta_0}{\varepsilon} \min\left\{dL_-, L_- + \frac{d}{\sqrt{n}}L_{\pm}\right\}\right)$
MARINA $\cap$ Rand K	$\mathcal{O}\left(\frac{\Delta_0}{\varepsilon} \min\left\{dL_-, \frac{d}{\sqrt{n}}L_+\right\}\right)$	$\mathcal{O}\left(\frac{\Delta_0}{\varepsilon} \min\left\{dL_-, L_- + \frac{d}{\sqrt{n}}L_+\right\}\right)$
EF21 $\cap$ Top K	$\mathcal{O}\left(\frac{\Delta_0}{\varepsilon}dL_-\right)$	$\mathcal{O}\left(\frac{\Delta_0}{\varepsilon}dL_-\right)$

$d \geq n \implies$ up to $\mathcal{O}(\sqrt{n})$ improvement

$d \leq n \implies$ up to $\mathcal{O}\left(1 + \frac{d}{\sqrt{n}}\right)$ improvement

Experiments

Synthetic Nonconvex Quadratics

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

features: $d = 1000$

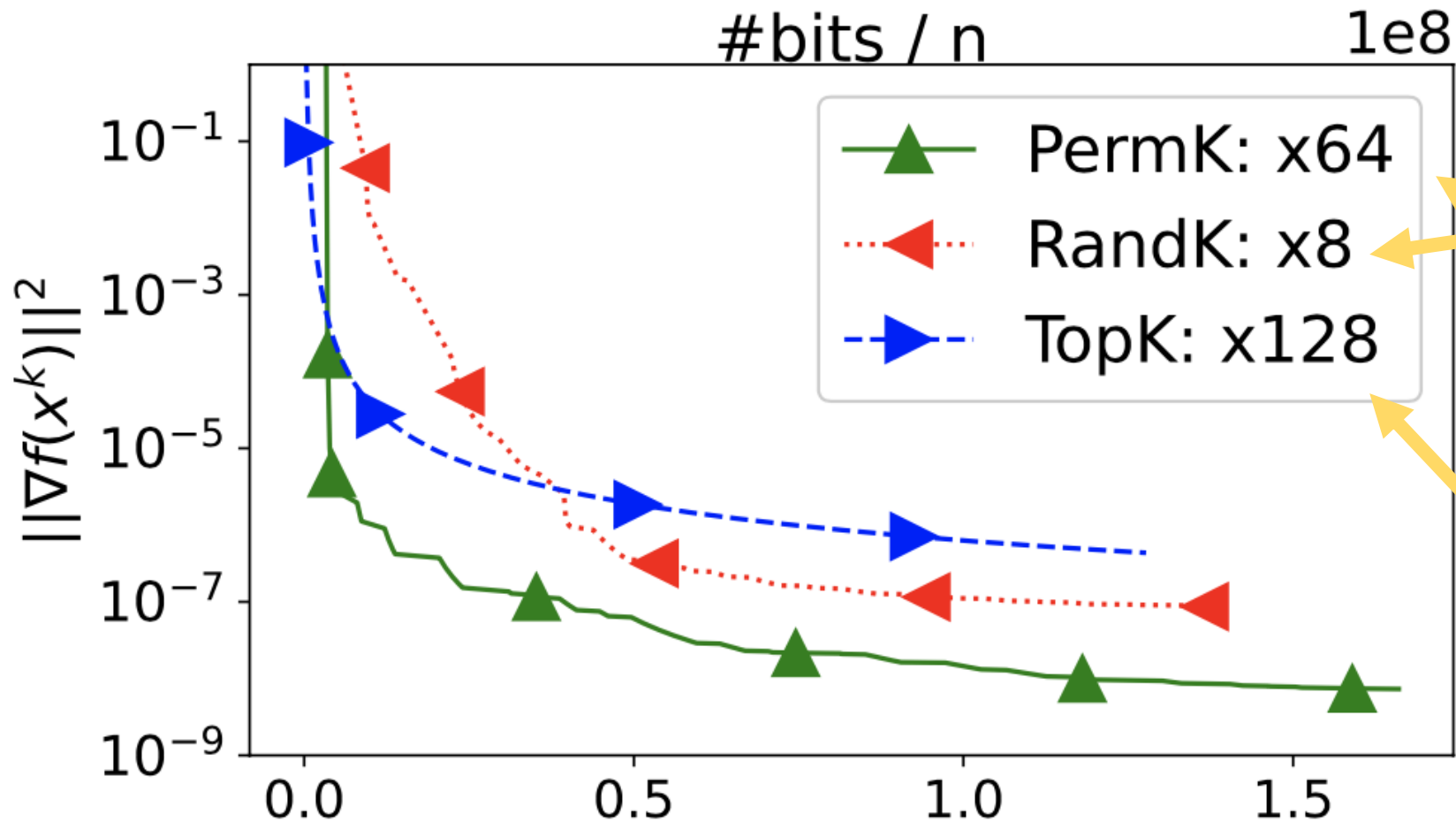
clients: $n = 1000$

$$f_i(x) = \frac{1}{2} x^\top \mathbf{A}_i x - b_i^\top x$$

Zero Hessian Variance

$$L_{\pm} = 0$$

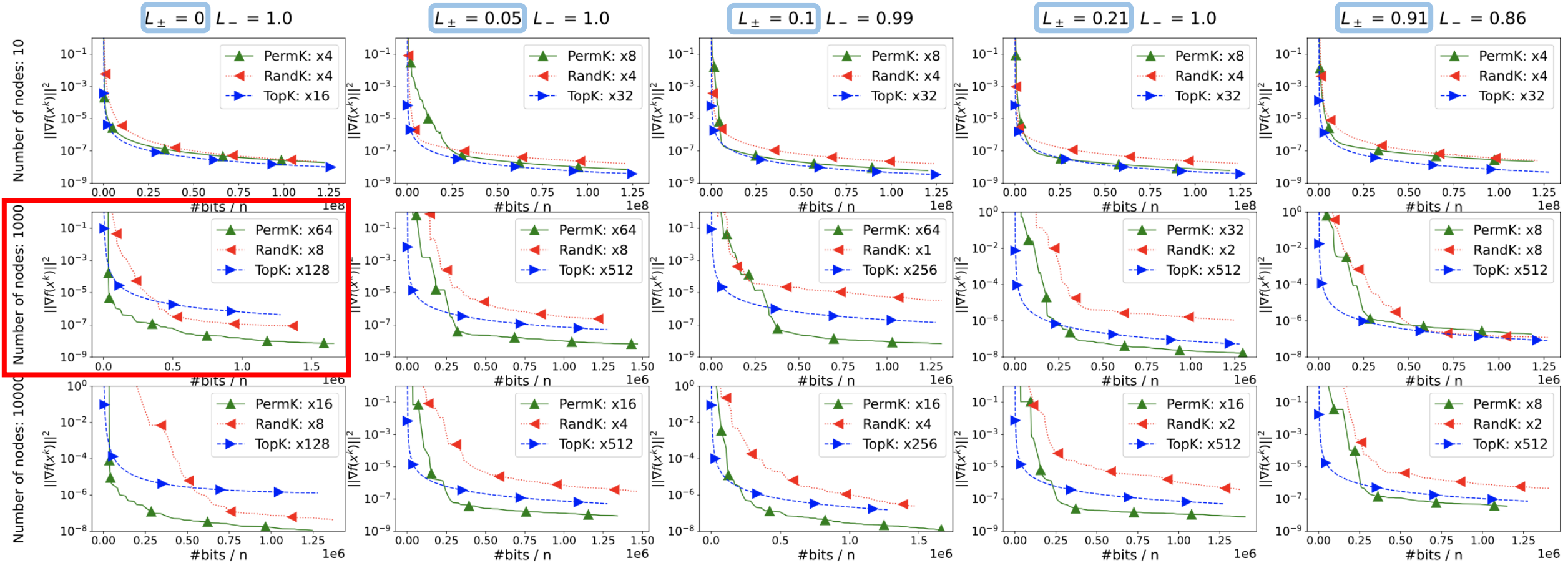
Homogeneous
data regime



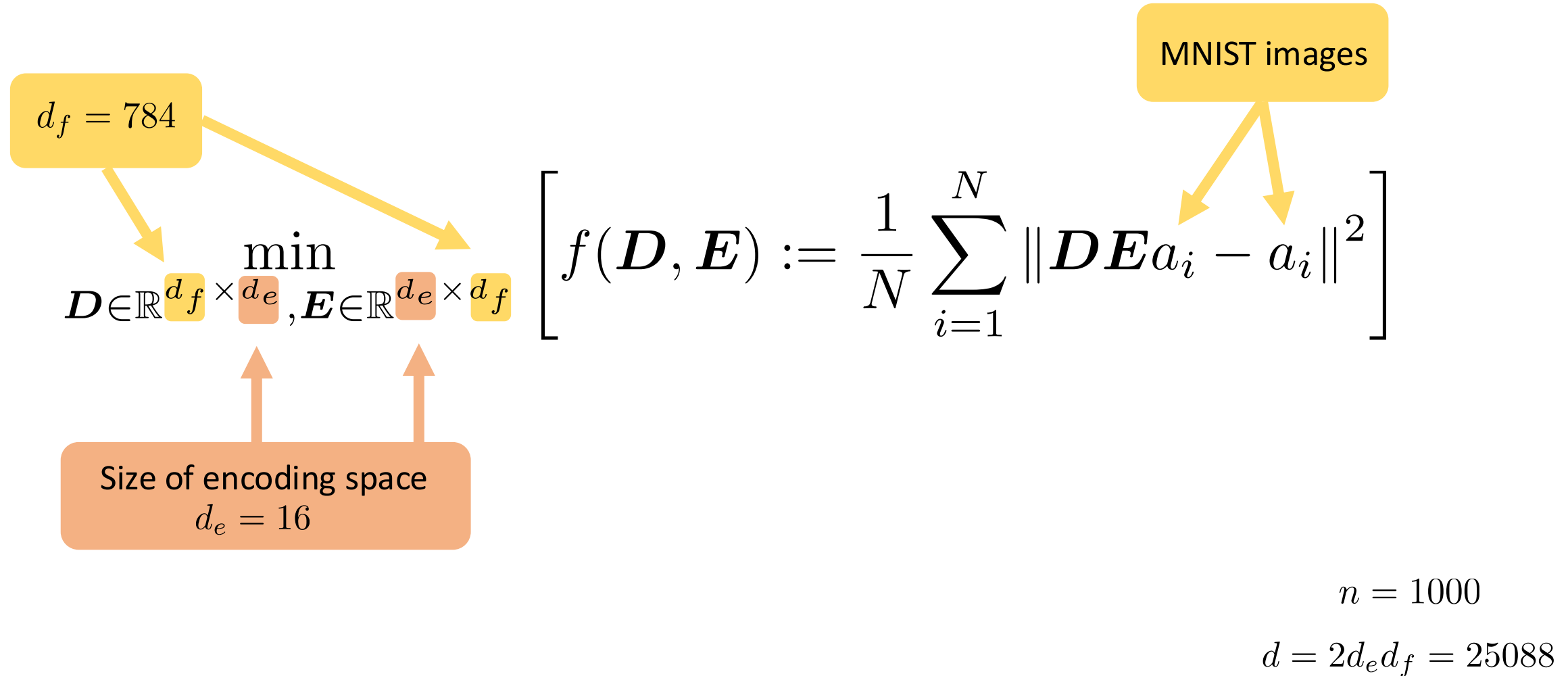
$$f_i(x) = \frac{1}{2}x^\top \mathbf{A}_i x - b_i^\top x$$

$$n = 1000, d = 1000$$

Synthetic Experiments

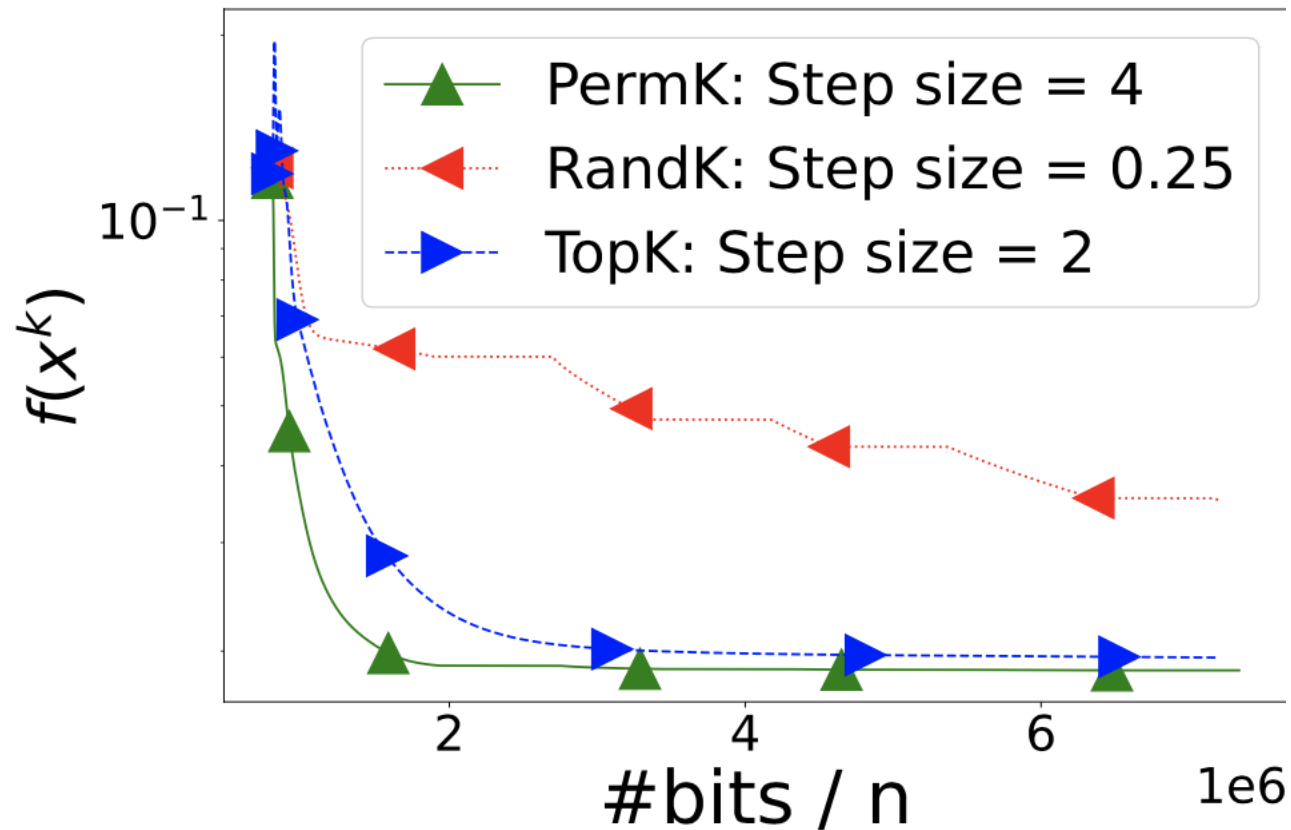


Training an Autoencoder with MNIST



Training an Autoencoder with MNIST

Homogeneous data partition

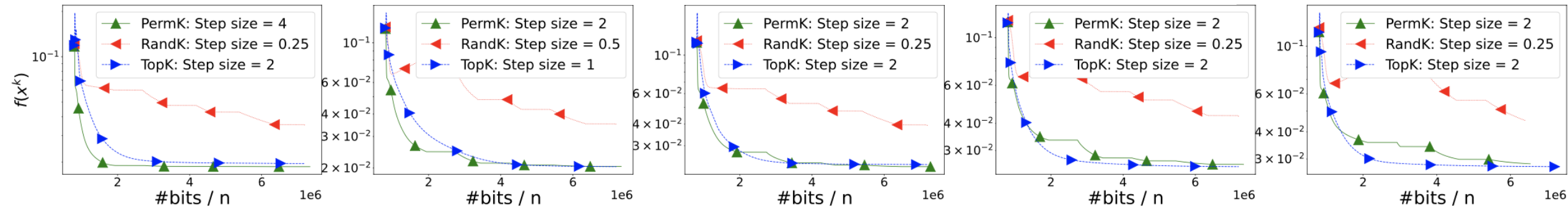


$$n = 1000$$

$$d = 2d_e d_f = 25088$$

Training an Autoencoder with MNIST

Homogeneous
data partition



Heterogeneous
data partition

As data heterogeneity increases, PermK does worse, as predicted by theory. Still much better than RandK!!!

MARINA – Further Sources

02/2021

MARINA



Eduard Gorbunov, Konstantin Burlachenko, Zhize Li and P.R.
MARINA: Faster non-convex distributed learning with compression
ICML 2021

Original MARINA paper

07/2021



Rafał Szlendak, Alexander Tyurin and P.R.
Permutation compressors for provably faster distributed nonconvex optimization
ICLR 2022



Improved MARINA analysis:
introduced **correlated compressors**
(e.g. AB inequality, permutation
compressors) and **Hessian variance** (2nd
order similarity)

06/2022



Eduard Gorbunov, Samuel Horváth, P.R. and Gauthier Gidel
Variance reduction is an antidote to Byzantines: better rates, weaker assumptions and communication compression as a cherry on the top
ICLR 2023

Better rates under **Byzantine workers** (VR can help to fight Byz workers)

10/2023



Hanmin Li, Avetik Karagulyan, and P.R.
MARINA meets matrix stepsizes: Variance reduced distributed non-convex optimization
arXiv:2310.04614

Theoretical
improvement under
matrix stepsizes

MARINA – Further Sources

10/2023



Ahmad Rammal, Kaja Grutkowska, Nikita Fedin, Eduard Gorbunov, and P.R.
Communication compression for Byzantine robust learning: New efficient algorithms and improved rates
AISTATS 2024

Further improvements

11/2023



Grigory Malinovsky, P.R., Samuel Horváth, and Eduard Gorbunov
Byzantine robustness and partial participation can be achieved simultaneously: just clip gradient differences
NeurIPS 2024

First Byz robustness results under partial participation

09/2024

MARINA-P



Kaja Grutkowska, Alexander Tyurin, and P.R.
Improving the worst-case bidirectional communication complexity for nonconvex distributed optimization under function similarity
NeurIPS 2024

MARINA-P = MARINA on the server side

12/2024

MARINA-P



Igor Sokolov and P.R.
MARINA-P: Superior performance in non-smooth federated optimization with adaptive stepsizes
arXiv:2412.17082

Non-smooth analysis of MARINA-P



Bidirectional Compression

Cost Model

- Uplink (w2s) communication cost = 1 (per bit)
- Worker computation cost = 0
- Server aggregation cost = 0
- Downlink (=s2w) communication cost = r (per bit)

Bidirectional Compression: Early Works

5/2019

DQSGD



Samuel Horváth, Chen-Yu Ho, L'udovít Horváth, Atal Narayan Sahu, Marco Canini and P.R.

Natural compression for distributed deep learning

MSML 2022

1st time a theoretical benefit of bidirectional compression was identified (for some compressors)

5/2019

DoubleSqueeze



Hanlin Tang, Xiangru Lian, Chen Yu, Tong Zhang, Ji Liu

DoubleSqueeze: Parallel Stochastic Gradient Descent with Double-Pass Error-Compensated Compression

ICML 2019

02/2021

MCM



Constantin Philippenko and Aymeric Dieuleveut

Preserved central model for faster bidirectional compression in distributed settings

NeurIPS 2021

7/2021

EF21-PP



Ilyas Fatkhullin, Igor Sokolov, Eduard Gorbunov, Zhize Li and Peter Richtárik

EF21 with bells & whistles: practical algorithmic extensions of modern error feedback

JMLR 2025

Bidirectional Compression: From Multiplicative to Additive Dependence on Server & Client Variances

09/2022

EF21P-DCGD
EF21P-DIANA



Kaja Gruntkowska, Alexander Tyurin, Peter Richtárik
EF21-P and Friends: Improved Theoretical Communication Complexity for Distributed Optimization with Bidirectional Compression
ICML 2023

- EF14 is used all wrong!
- Use it on the server to compress the model (=EF21-P)
- First additive dependence on server & worker compression variances

05/2023

2-Direction



Alexander Tyurin and P.R.
2Direction: Theoretically faster distributed training with bidirectional communication compression
NeurIPS 2023

Acceleration + bidirectional compression can be combined

09/2024

MARINA-P



Kaja Gruntkowska, Alexander Tyurin, and P.R.
Improving the worst-case bidirectional communication complexity for nonconvex distributed optimization under function similarity
NeurIPS 2024

Server sends a different compressed version of the same message to each client; this results in theoretical benefit!

01/2026

BiCoLoR



Laurent Condat, Artavazd Maranjyan, P.R.
BiCoLoR: Communication-Efficient Optimization with Bidirectional Compression and Local Training
arXiv:2601.12400, 2026

Server Compression (via EF21-P) Helps

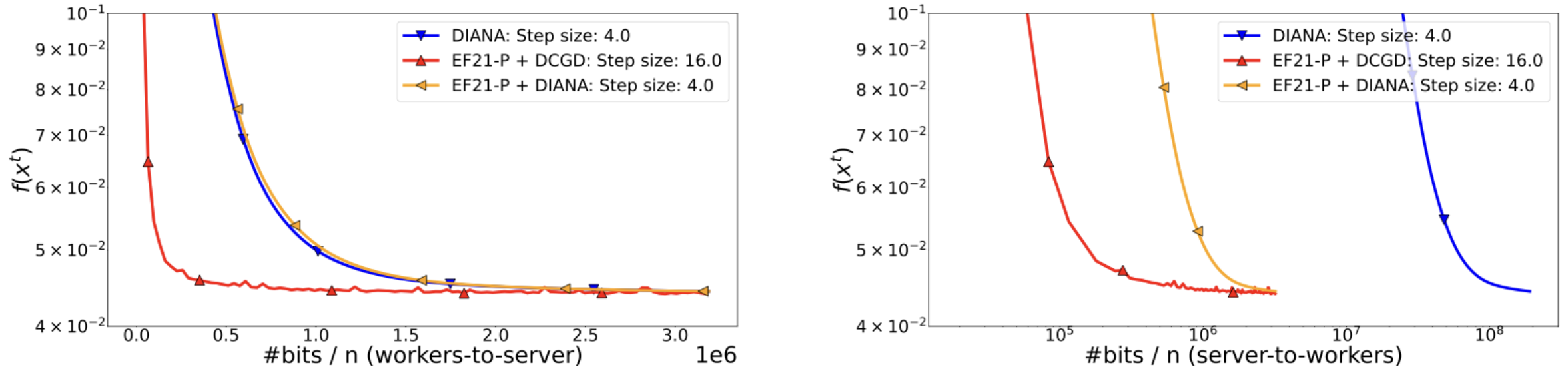


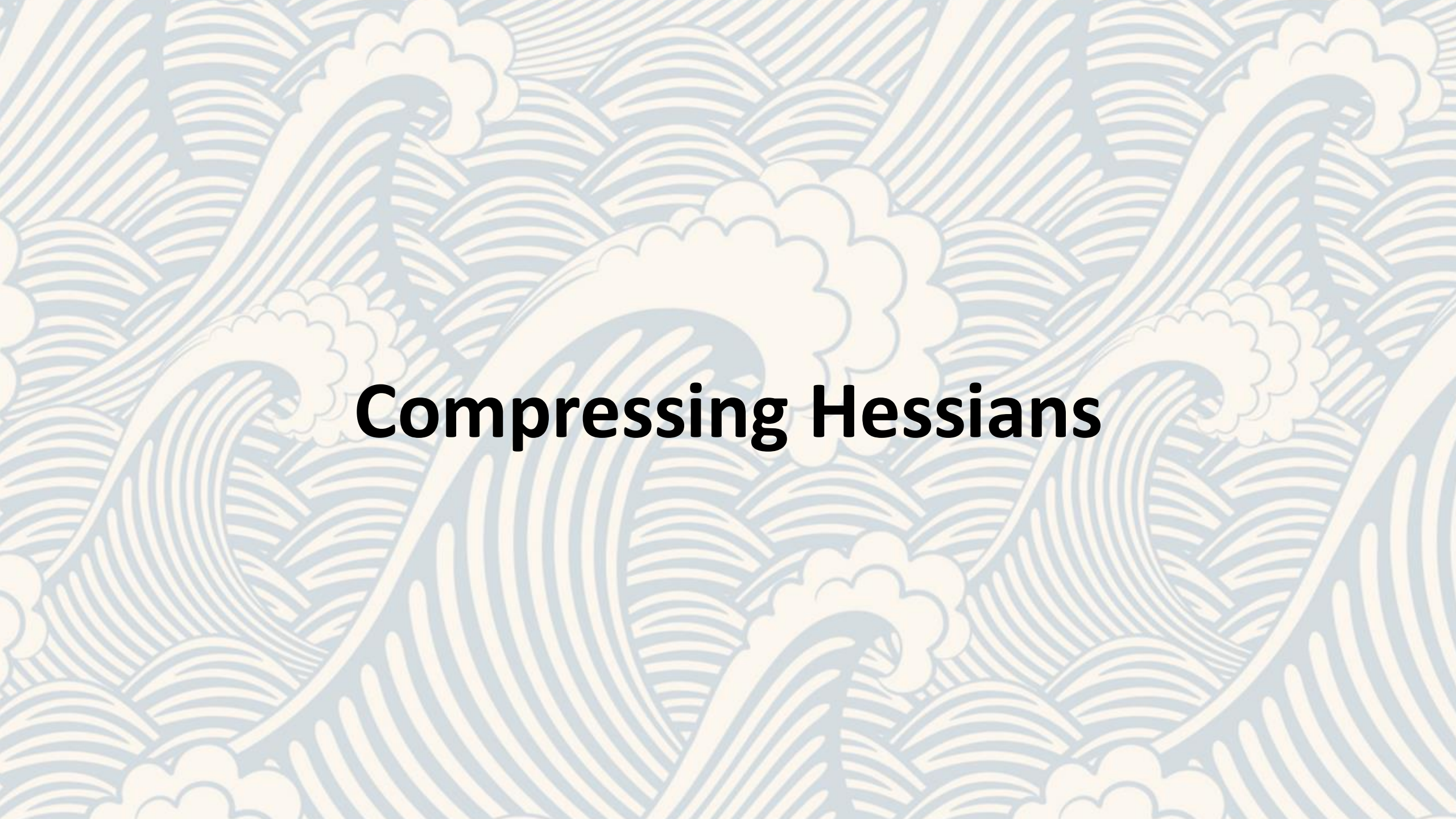
Figure 3: Logistic Regression with *w8a* dataset. # of workers $n = 10$. $K = 10$ in all compressors.



Kaja Grunkowska, Alexander Tyurin, Peter Richtárik

EF21-P and Friends: Improved Theoretical Communication Complexity for Distributed Optimization with Bidirectional Compression

ICML 2023

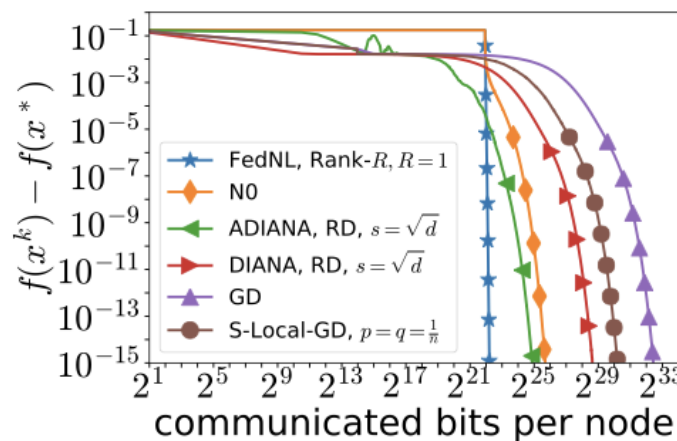


Compressing Hessians

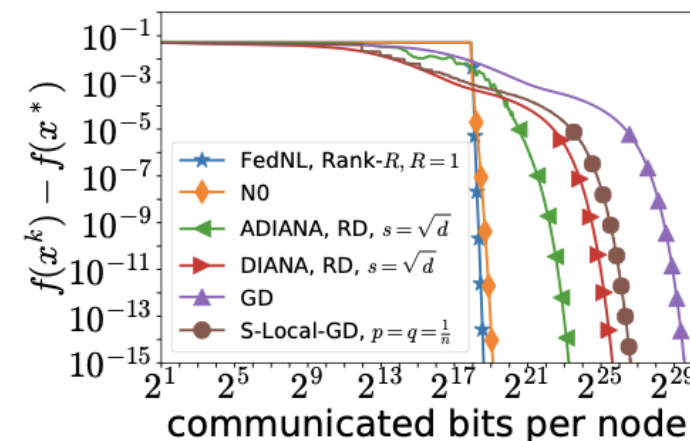
Towards Comm.-Efficient 2nd-Order Methods

KEY IDEAS:

- Replace local training by Hessian computation!
- Compress Hessians!
- Apply EF21!
- Do not compress gradients

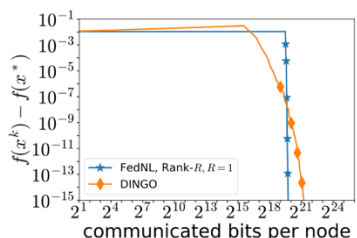


(a) *madelon*, $\lambda = 10^{-3}$

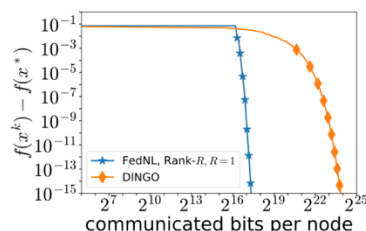


(b) *ala*, $\lambda = 10^{-4}$

Theory: strong!



(c) *w8a*, $\lambda = 10^{-3}$



(d) *phishing*, $\lambda = 10^{-4}$



Rustem Islamov, Xun Qian and P.R.

Distributed second order methods with fast rates and compressed communication

ICML 2021



Mher Safaryan, Rustem Islamov, Xun Qian and P.R.

FedNL: Making Newton-type methods applicable to federated learning

ICML 2022

Towards Comm.-Efficient 2nd-Order Methods

FedNL: Making Newton-Type Methods Applicable to Federated Learning

Table 2: Comparison of the main features of our family of FedNL algorithms and results with those of Islamov et al. (2021), which we used as an inspiration. We have made numerous and significant modifications and improvements in order to obtain methods applicable to federated learning.

#	Feature	(Islamov et al., 2021)	This Work
[hd]	supports <i>h</i> eterogeneous <i>d</i> ata setting	✓	✓
[fs]	applies to general <i>f</i> inite- <i>s</i> um problems	✗	✓
[as]	uses <i>a</i> daptive <i>s</i> tepsizes	✓	✓
[pe]	<i>p</i> rivacy is <i>e</i> nhanced (training data is not sent to the server)	✗	✓
[uc]	supports <i>u</i> nbiased Hessian <i>c</i> ompression (e.g., Rand- K)	✓	✓
[cc]	supports <i>c</i> ontractive Hessian <i>c</i> ompression (e.g., Top- K)	✗	✓
[fr]	<i>f</i> ast local <i>r</i> ate: independent of the condition number	✓	✓
[fr]	<i>f</i> ast local <i>r</i> ate: independent of the # of training data points	✗	✓
[fr]	<i>f</i> ast local <i>r</i> ate: independent of the compressor variance	✗	✓
[pp]	supports <i>p</i> artial <i>p</i> articipation	✗	✓(Alg 2)
[gg]	has <i>g</i> lobal convergence <i>g</i> uarantees via line search	✗	✓(Alg 3)
[gg]	has <i>g</i> lobal convergence <i>g</i> uarantees via cubic regularization	✓	✓(Alg 4)
[gc]	supports smart uplink <i>g</i> radient <i>c</i> ompression at the devices	✗	✓(Alg 5)
[mc]	supports smart downlink <i>m</i> odel <i>c</i> ompression by the master	✗	✓(Alg 5)
[lc]	performs useful <i>l</i> ocal <i>c</i> omputation	✓	✓



Towards Comm.-Efficient 2nd-Order Methods

2/2021



Rustem Islamov, Xun Qian and P.R.
Distributed second order methods with fast rates and compressed communication
ICML 2021

- The 1st method applying EF to Hessians
- Local convergence rate independent of condition number!

6/2021



Mher Safaryan, Rustem Islamov, Xun Qian and P.R.
FedNL: Making Newton-type methods applicable to federated learning
ICML 2022

Extension to FL scenarios
(e.g., partial participation)

11/2021



Xun Qian, Rustem Islamov, Mher Safaryan and P.R.
Basis matters: better communication-efficient second order methods for federated learning
AISTATS 2022

6/2022



Rustem Islamov, Xun Qian, Slavomír Hanzely, Mher Safaryan and P.R.
Distributed Newton-type methods with communication compression and Bernoulli aggregation
TMLR 2023

10/2024



Konstantin Burlachenko and P.R.
Unlocking FedNL: Self-contained compute-optimized implementation

System/software tuning:
1000x speedup

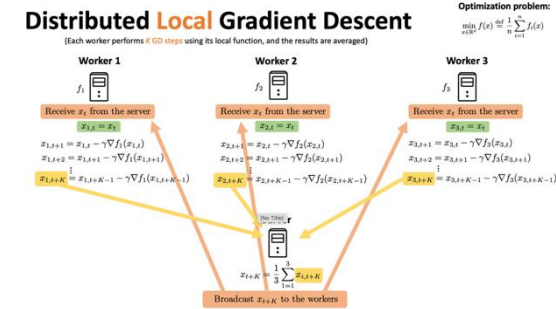
The background is a traditional Chinese ink wash painting. It features a misty, layered mountain landscape with soft, hazy peaks. In the foreground, dark, silhouetted mountain ridges are visible. Delicate, dark brown branches with clusters of pink cherry blossoms and falling petals frame the top and sides of the image. The overall style is elegant and serene, typical of classical Chinese art.

Lecture 3

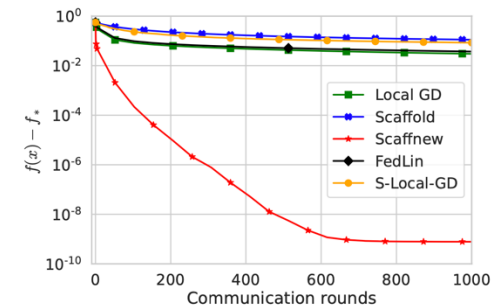
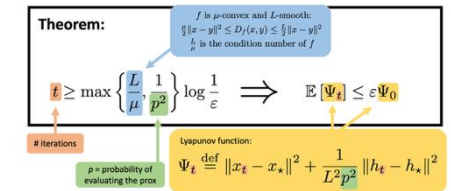
Local Training

Outline

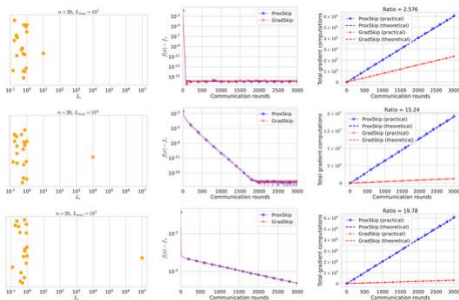
1. Brief History of Local Training
2. 5th Generation of Local Training Methods
3. ProxSkip
4. GradSkip
5. RandProx
6. Further Approaches (similarity, Acc-S-DANE, ...)



ProxSkip: Bounding the # of Iterations



(c) theoretical hyper-parameters



Algorithm 1 ProxSkip

```

1: stepsize  $\gamma > 0$ , probability  $p > 0$ , initial iterate  $x_0 \in \mathbb{R}^d$ , initial control variate  $h_0 \in \mathbb{R}^d$ , number of iterations  $T \geq 1$ 
2: for  $t = 0, 1, \dots, T - 1$  do
3:    $\hat{x}_{t+1} = x_t - \gamma (\nabla f(x_t) - h_t)$   $\diamond$  Take a gradient-type step adjusted via the control variate  $h_t$ 
4:   Flip a coin  $\theta_t \in \{0, 1\}$  where  $\text{Prob}(\theta_t = 1) = p$   $\diamond$  Flip a coin that decides whether to skip the prox or not
5:   if  $\theta_t = 1$  then
6:      $x_{t+1} = \text{prox}_{\frac{\gamma}{p}}(\hat{x}_{t+1} - \frac{\gamma}{p} h_t)$   $\diamond$  Apply prox, but only very rarely! (with small probability  $p$ )
7:   else
8:      $x_{t+1} = \hat{x}_{t+1}$   $\diamond$  Skip the prox!
9:   end if
10:   $h_{t+1} = h_t + \frac{\gamma}{p} (x_{t+1} - \hat{x}_{t+1})$   $\diamond$  Update the control variate  $h_t$ 
11: end for
  
```

Optimization Formulation of Federated Learning

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

model parameters / features

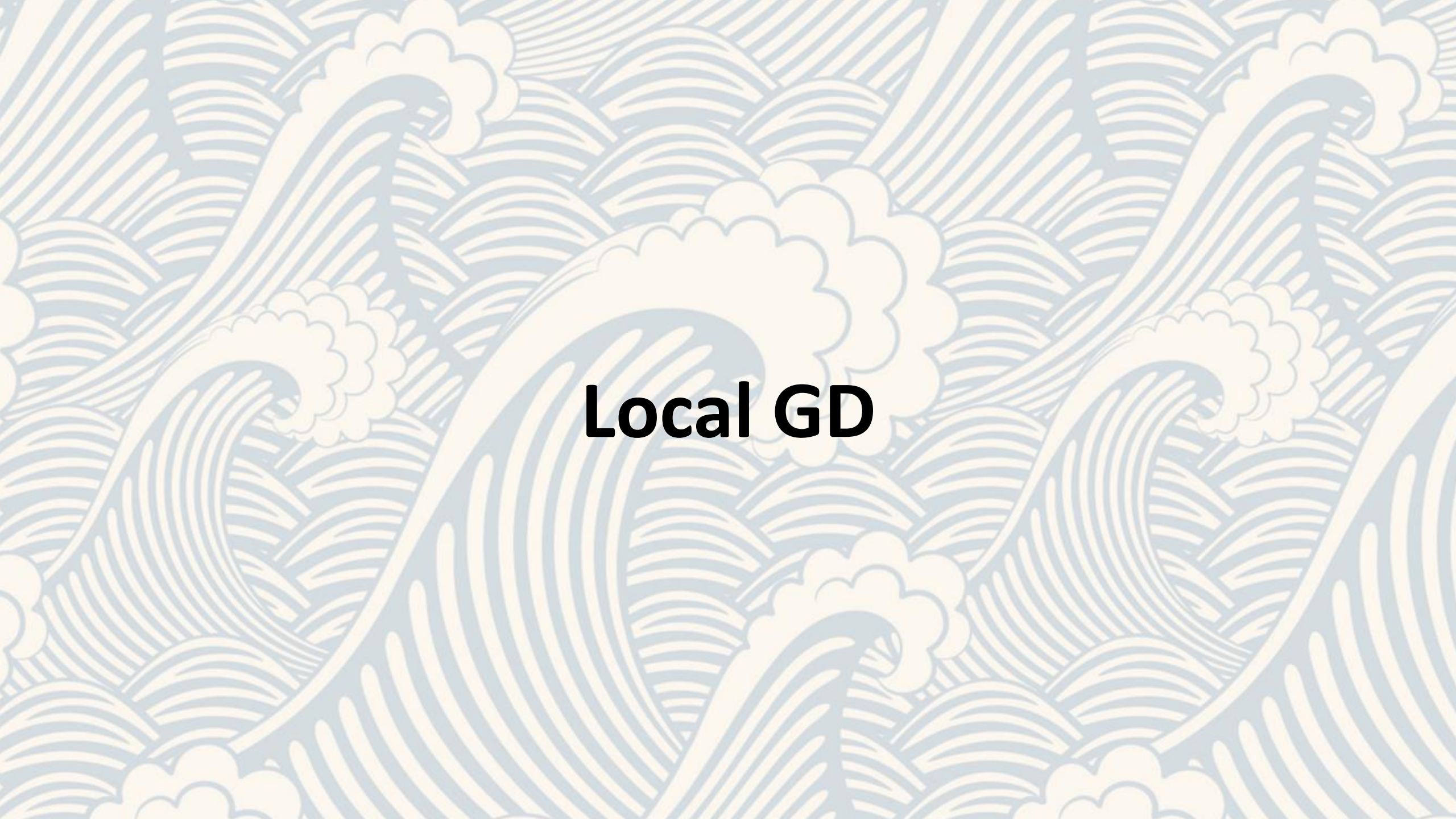
$$\frac{1}{n} \sum_{i=1}^n f_i(x)$$

devices /
machines

Loss on local data $\mathcal{D}_i$ stored on device i

$$f_i(x) = \mathbb{E}_{\xi \sim \mathcal{D}_i} f_{i,\xi}(x)$$

The datasets $\mathcal{D}_1, \dots, \mathcal{D}_n$ can be arbitrarily heterogeneous

The background of the slide is a repeating pattern of stylized waves. The waves are rendered in a light blue color against a white background. Each wave is composed of multiple curved lines, creating a sense of movement and depth. The pattern is dense and covers the entire area of the slide.

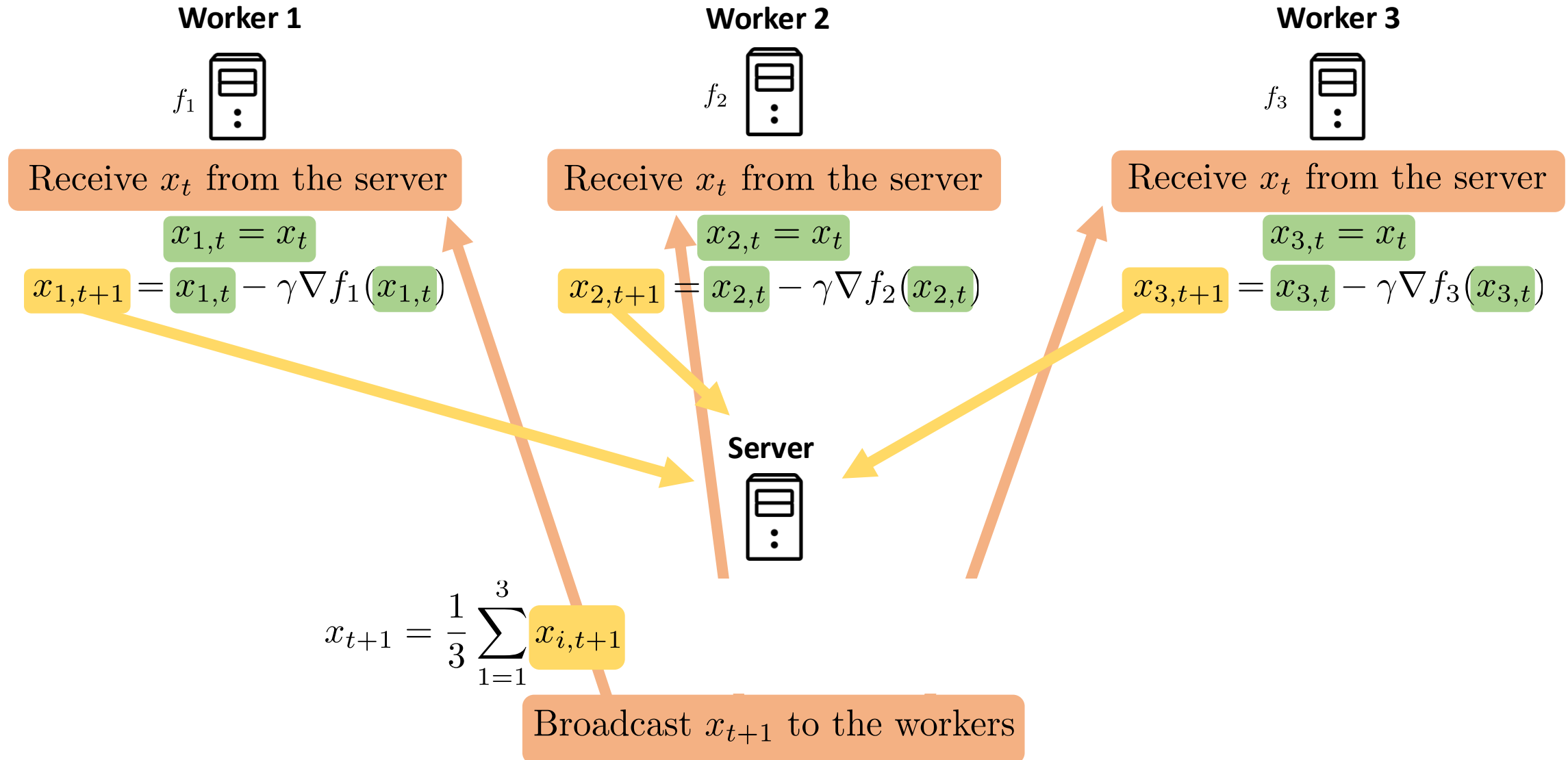
Local GD

Distributed Gradient Descent

(Each worker performs 1 GD step using its local function, and the results are averaged)

Optimization problem:

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

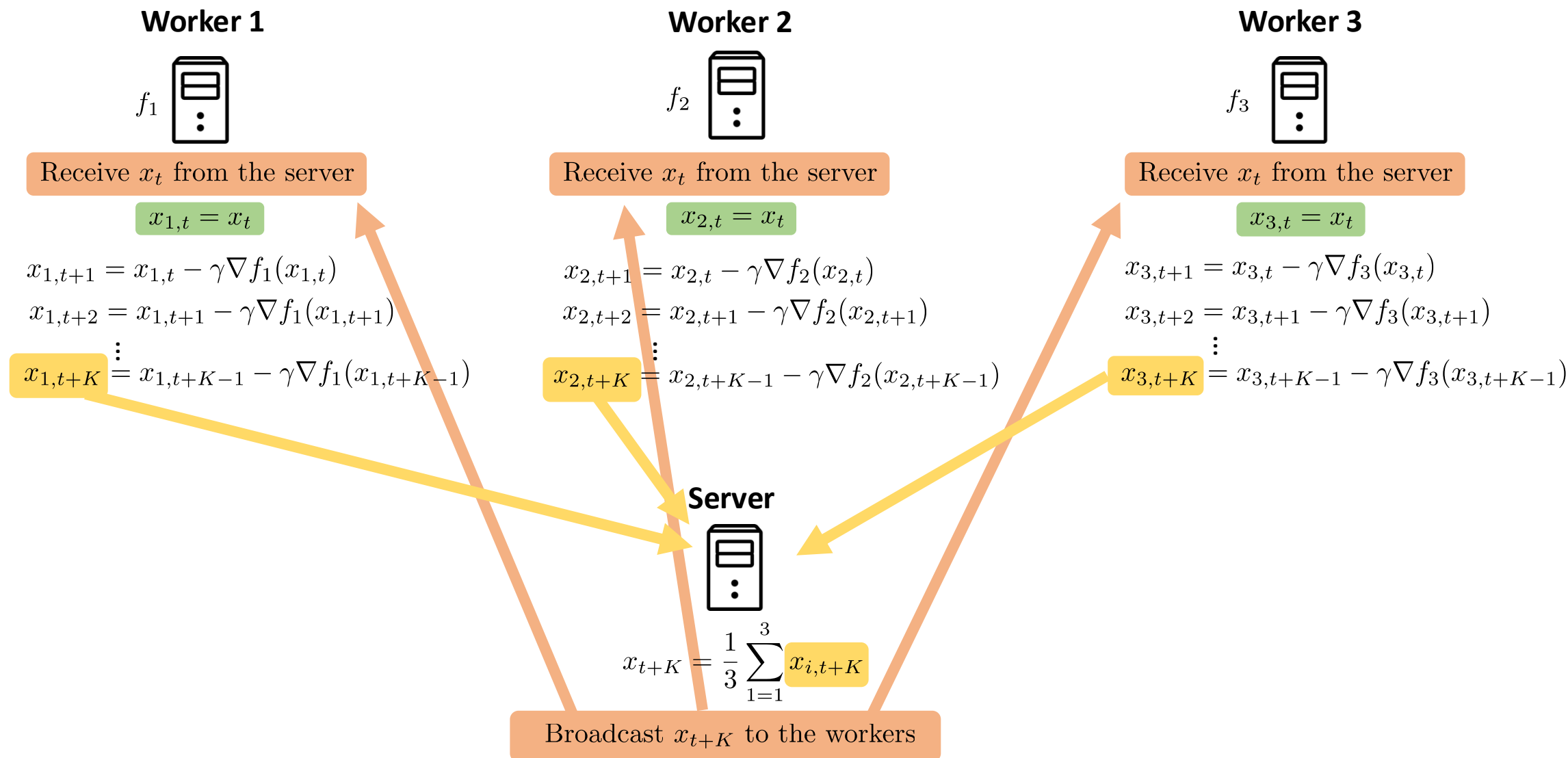


Distributed Local Gradient Descent

(Each worker performs K GD steps using its local function, and the results are averaged)

Optimization problem:

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



Brief History of Local GD



Grigory Malinovsky, Kai Yi and P.R.

Variance reduced ProxSkip: algorithm, theory and application to federated learning

NeurIPS 2022

Brief History of Local Training Methods

Table 1: Five generations of local training (LT) methods summarizing the progress made by the ML/FL community over the span of 7+ years in the understanding of the *communication acceleration properties of LT*.

Generation ^(a)	Theory	Assumptions	Comm. Complexity ^(b)	Selected Key References
1. Heuristic	✗	—	empirical results only	LocalSGD [Povey et al., 2015]
	✗	—	empirical results only	SparkNet [Moritz et al., 2016]
	✗	—	empirical results only	FedAvg [McMahan et al., 2017]
2. Homogeneous	✓	bounded gradients	sublinear	FedAvg [Li et al., 2020b]
	✓	bounded grad. diversity ^(c)	linear but worse than GD	LFGD [Haddadpour and Mahdavi, 2019]
3. Sublinear	✓	standard ^(d)	sublinear	LGD [Khaled et al., 2019]
	✓	standard	sublinear	LSGD [Khaled et al., 2020]
4. Linear	✓	standard	linear but worse than GD	Scaffold [Karimireddy et al., 2020]
	✓	standard	linear but worse than GD	S-Local-GD [Gorbunov et al., 2020a]
	✓	standard	linear but worse than GD	FedLin [Mitra et al., 2021]
5. Accelerated	✓	standard	linear & better than GD	ProxSkip/Scaffnew [Mishchenko et al., 2022]
	✓	standard	linear & better than GD	ProxSkip-VR [THIS WORK]

^(a) Since client sampling (CS) and data sampling (DS) can only *worsen* theoretical communication complexity, our historical breakdown of the literature into 5 generations of LT methods focuses on the full client participation (i.e., no CS) and exact local gradient (i.e., no DS) setting. While some of the referenced methods incorporate CS and DS techniques, these are irrelevant for our purposes. Indeed, from the viewpoint of communication complexity, all these algorithms enjoy best theoretical performance in the no-CS and no-DS regime.

^(b) For the purposes of this table, we consider problem (1) in the *smooth* and *strongly convex* regime only. This is because the literature on LT methods struggles to understand even in this simplest (from the point of view of optimization) regime.

^(c) *Bounded gradient diversity* is a uniform bound on a specific notion of gradient variance depending on client sampling probabilities. However, this assumption (as all homogeneity assumptions) is very restrictive. For example, it is not satisfied the standard class of smooth and strongly convex functions.

^(d) The notorious FL challenge of handling non-i.i.d. data by LT methods was solved by Khaled et al. [2019] (from the viewpoint of *optimization*). From generation 3 onwards, there was no need to invoke any data/gradient homogeneity assumptions. Handling non-i.i.d. data remains a challenge from the point of view of *generalization*, typically by considering *personalized* FL models.



Grigory Malinovsky, Kai Yi and P.R.

Variance Reduced ProxSkip: Algorithm, Theory and Application to Federated Learning

NeurIPS 2022

Brief History of Local Training Methods

Generation 1: Heuristic

“No theory”

10/2014



Daniel Povey, Xiaohui Zhang, and Sanjeev Khudanpur
Parallel Training of DNNs with Natural Gradient and Parameter Averaging
ICLR Workshops 2015

11/2015



Philipp Moritz, Robert Nishihara, Ion Stoica, Michael I. Jordan
SparkNet: Training Deep Networks in Spark
ICLR 2015

02/2016



H. Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, Blaise Agüera y Arcas
Communication-Efficient Learning of Deep Networks from Decentralized Data
AISTATS 2017

Accepted as a workshop contribution at ICLR 2015

PARALLEL TRAINING OF DNNs WITH NATURAL GRADIENT AND PARAMETER AVERAGING

Daniel Povey, Xiaohui Zhang & Sanjeev Khudanpur
Carnegie Mellon University, Pittsburgh, PA, 15213, USA
(dpovey@cmu.edu), (xazhang), (skhudanp@cmu.edu)

ABSTRACT

We describe the neural-network training framework used in the Kaldi speech-recognition results, which is based around training DNNs with large amounts of training data using multiple GPUs equipped on multi-core machines, in order to be as hardware-efficient as possible, but without a way to use multiple machines without generating excessive network traffic. Our method is to average the neural-network parameters periodically (typically every minute or so), and redistribute them to the original parameters in the machines for further training. Each machine uses different data. By itself, this method does not work very well. However, we have further refined an approximate and efficient implementation of Natural Gradient for Stochastic Gradient Descent (NG-SGD), which seems to allow one parallel-averaging method to work well, as well as substantially improving the convergence of SGD on a single machine.

1 INTRODUCTION

Parallel training of neural networks generally makes use of some combination of model-parallelism and data-parallelism (Dean et al., 2015), and the natural approach to data-parallelism involves communication of model parameters to each locality. Here we describe our current best training framework which uses a different version of data-parallelism: we have multiple SGD processes on separate machines, and only infrequently (every minute or so) average the model parameters and redistribute them to the individual machines. This is very efficient for us in large-scale training of recurrent neural networks, but in our case it only works well when combined with an efficient implementation of natural gradient methods: gradient descent (NG-SGD) that we have developed. We don't attempt in this paper to develop a framework that explains why parameter averaging should work well despite non-convexity of DNNs, or why NG-SGD is so helpful. The point of this paper is to describe our methods and to establish that respectively they work well. The significance of this work is that we show that it is possible to get a linear speedup when increasing the number of GPUs, without requiring frequent data transfer (because data only needs to go to about 4 or 5 GPUs).

In Section 2 we describe our problem setting, which is Deep Neural Networks (DNNs) applied to speech recognition, although we show our ideas are more general than this. In Section 3 we describe the parallel training method. In Section 4 we describe the general ideas behind our natural gradient method, although most of the technical details have been relegated to appendices. In this paper we also give our proofs. In the end we discuss in Section 5 what we think we can and can't be proven about our methods. Section 6 has experiments on the convergence of SGD with and without natural gradient and parallelism. We conclude in Section 7.

There are two versions of our NG-SGD method: a “simple” version and an “enhanced” one. Technical details for these are in Appendices A and B respectively. Appendix C has background information on our DNN implementation.

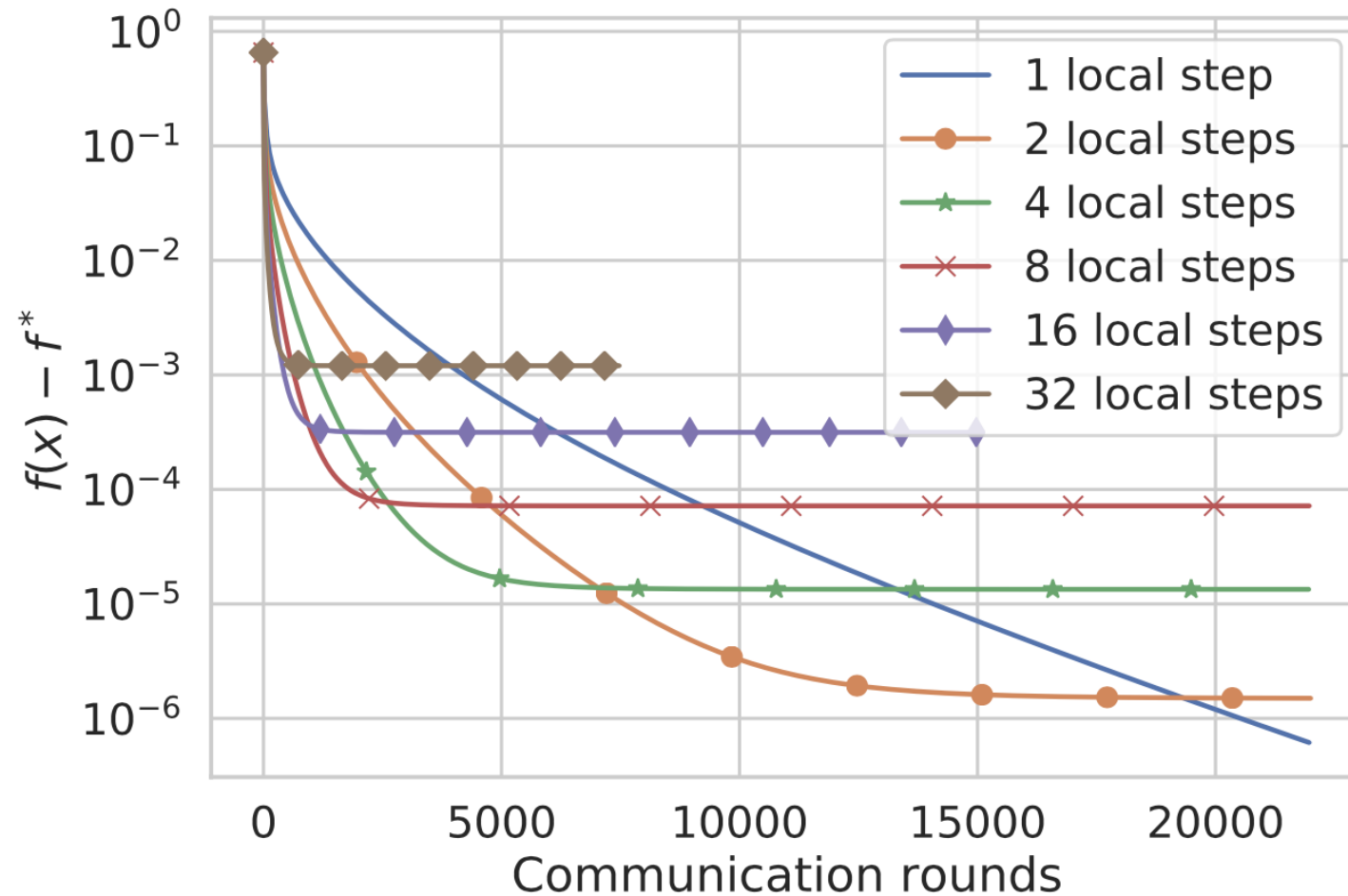
2 PROBLEM SETTING

When training DNNs for speech recognition, the fundamental problem is that of classifying vectors $x \in \mathbb{R}^D$ as corresponding to classes $1, \dots, C$. The dimension D is typically around 4000.

1

Brief History of Local Training Methods

Generation 1: Heuristic



L2-regularized logistic regression
LibSVM mushrooms dataset

Brief History of Local Training Methods

Generation 2: Homogeneous

“Theory requires data to be similar/homogeneous across the clients”

12/2018

FedProx



T. Li, A K Sahu, M. Zaheer, M. Sanjabi,
A. Talwalkar, V. Smith
**Federated Optimization in
Heterogeneous Networks**
MLSys 2020

$$x_i^{k+1} \approx \arg \min_x f_i(x) + \frac{\lambda}{2} \|x - x^k\|^2 \quad x^{k+1} = \frac{1}{n} \sum_{i=1}^n x_i^{k+1}$$

Bounded gradient diversity (aka strong growth):

$$\frac{1}{n} \sum_{i=1}^n \|\nabla f_i(x)\|^2 \leq C \|\nabla f(x)\|^2 \quad \forall x \in \mathbb{R}^d$$

07/2019



Xiang Li, Kaixuan Huang, Wenhao Yang, Shusen
Wang and Zhihua Zhang
On the Convergence of FedAvg on Non-IID Data
ICLR 2020

10/2019



Farzin Haddadpour and Mehrdad Mahdavi
**On the Convergence of Local Descent Methods in
Federated Learning**
arXiv:1910.14425, 2019

Bounded gradients:

$$\|\nabla f_i(x)\| \leq B \quad \forall x \in \mathbb{R}^d \quad \forall i \in \{1, 2, \dots, n\}$$

Brief History of Local Training Methods

Generation 3: Sublinear (or “Linear to a Neighborhood”)

“Heterogeneous data is allowed,
but # communications is worse than GD if high accuracy is required”

10/2019

Local GD



Ahmed Khaled, Konstantin Mishchenko, P.R.

First Analysis of Local GD on Heterogeneous Data

NeurIPS 2019 Workshop on Federated Learning for Data Privacy and Confidentiality, 2019

10/2019

Local SGD



Ahmed Khaled, Konstantin Mishchenko, P.R.

Tighter Theory for Local SGD on Identical and Heterogeneous Data

AISTATS 2020

04/2020

Local Fixed Point
Method



Grigory Malinovsky, Dmitry Kovalev, Elnur Gasanov, Laurent Condat, P.R.

From Local SGD to Local Fixed-Point Methods for Federated Learning

ICML 2020

Shows (faster, in
terms of #
communications)
linear convergence to
a non-optimal point

Brief History of Local Training Methods

Generation 4: Linear

“Heterogeneous data is allowed, but the rate at best matches that of GD”

10/2019

Scaffold



Sai P. Karimireddy, S. Kale, M. Mohri, S. J. Reddi, S. U. Stich, A. T. Suresh
SCAFFOLD: Stochastic Controlled Averaging for Federated Learning
ICML 2020

11/2020

S-Local-GD, Local-GD*
S-Local-SVRG



Eduard Gorbunov, Filip Hanzely and P.R.
Local SGD: Unified Theory and New Efficient Methods
AISTATS 2021

02/2021

FedLin



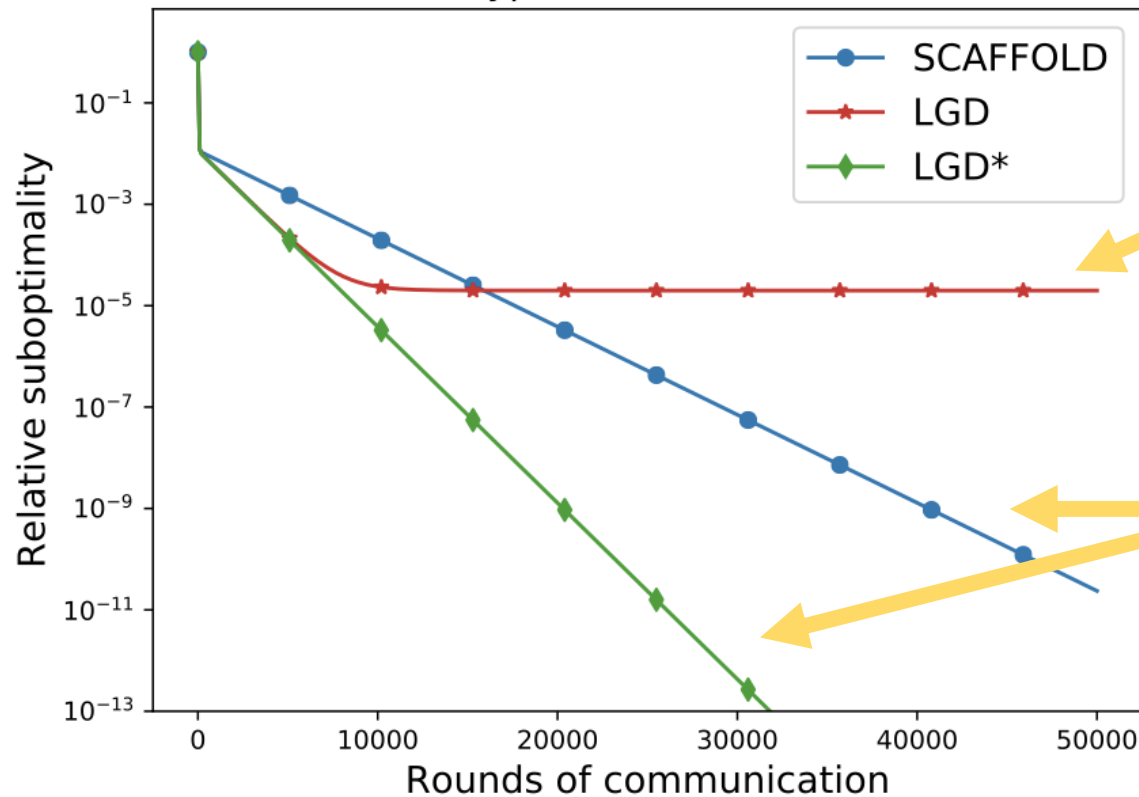
Aritra Mitra, Rayana Jaafar, George J. Pappas, Hamed Hassani
Linear Convergence in Federated Learning: Tackling Client Heterogeneity & Sparse Gradients
NeurIPS 2021

Method	$\mathbf{a}_t^*, \mathbf{h}_t^*, \mathbf{l}_t^*$	Complexity	Setting	Sec
Local-SGD, Alg. 1 (Woodworth et al., 2020a)	$f_t(x_t^*), 0, -$	$\frac{L}{\mu} + \frac{\sigma^2}{\mu^2} + \sqrt{\frac{L(\sigma^2 + \mu^2)}{\mu^2}}$	UBV, C-Het	G.1.1
Local-SGD, Alg. 1 (Kobayashi et al., 2020)	$f_t(x_t^*), 0, -$	$\frac{L}{\mu} + \frac{\sigma^2}{\mu^2} + \sqrt{\frac{L(\sigma^2 + \mu^2)}{\mu^2}}$	UBV, C-Het	G.1.1
Local-SGD, Alg. 1 (Khalid et al., 2020)*	$f_t(x_t^*), 0, -$	$\frac{L + \sigma^2}{\mu} + \sqrt{\frac{L(\sigma^2 + \mu^2)}{\mu^2}}$	ES, C-Het	G.1.2
Local-SGD, Alg. 1 (Khalid et al., 2020)*	$f_t(x_t^*), 0, -$	$\frac{L + \sigma^2}{\mu} + \sqrt{\frac{L(\sigma^2 + \mu^2)}{\mu^2}}$	ES, Het	G.1.2
Local-SVRG, Alg. 2 (NEW)	$\nabla f_{t, \mathbf{a}_t}(x_t^*) - \nabla f_{t, \mathbf{h}_t}(y_t^*)$	$m + \frac{L + \sigma^2}{\mu} + \sqrt{\frac{L(\sigma^2 + \mu^2)}{\mu^2}}$	simple, C-Het	G.2
Local-SVRG, Alg. 2 (NEW)	$\nabla f_{t, \mathbf{a}_t}(x_t^*) - \nabla f_{t, \mathbf{h}_t}(y_t^*)$	$m + \frac{L + \sigma^2}{\mu} + \sqrt{\frac{L(\sigma^2 + \mu^2)}{\mu^2}}$	simple, C-Het	G.2
S*-Local-SGD, Alg. 3 (NEW)	$f_t(x_t^*), \mathbf{h}_t^* - \frac{1}{n} \sum_{i=1}^n \mathbf{h}_i^*,$	$\frac{L}{\mu} + \frac{\sigma^2}{\mu^2} + \sqrt{\frac{L(\sigma^2 + \mu^2)}{\mu^2}}$	UBV, Het	G.3
SS-Local-SGD, Alg. 4 (Karimireddy et al., 2019a)	$f_t(x_t^*), \mathbf{h}_t^* - \frac{1}{n} \sum_{i=1}^n \mathbf{h}_i^*,$	$\frac{L}{\mu} + \frac{\sigma^2}{\mu^2} + \sqrt{\frac{L(\sigma^2 + \mu^2)}{\mu^2}}$	UBV, Het	G.4.1
SS-Local-SGD, Alg. 4 (NEW)	$f_t(x_t^*), \mathbf{h}_t^* - \frac{1}{n} \sum_{i=1}^n \mathbf{h}_i^*,$	$\frac{L}{\mu} + \frac{\sigma^2}{\mu^2} + \sqrt{\frac{L(\sigma^2 + \mu^2)}{\mu^2}}$	ES, Het	G.4.2
S*-Local-SGD*, Alg. 5 (NEW)	$\nabla f_{t, \mathbf{a}_t}(x_t^*) - \nabla f_{t, \mathbf{h}_t}(y_t^*)$	$\left(\frac{L}{\mu} + \frac{\sigma^2}{\mu^2} + \sqrt{\frac{L(\sigma^2 + \mu^2)}{\mu^2}} \right) \log \frac{1}{\epsilon}$	simple, Het	G.5
S-Local-SVRG, Alg. 6 (NEW)	$\nabla f_{t, \mathbf{a}_t}(x_t^*) - \nabla f_{t, \mathbf{h}_t}(y_t^*)$	$\left(m + \frac{L}{\mu} + \frac{\sigma^2}{\mu^2} + \sqrt{\frac{L(\sigma^2 + \mu^2)}{\mu^2}} \right) \log \frac{1}{\epsilon}$	simple, Het	G.6

Brief History of Local Training Methods

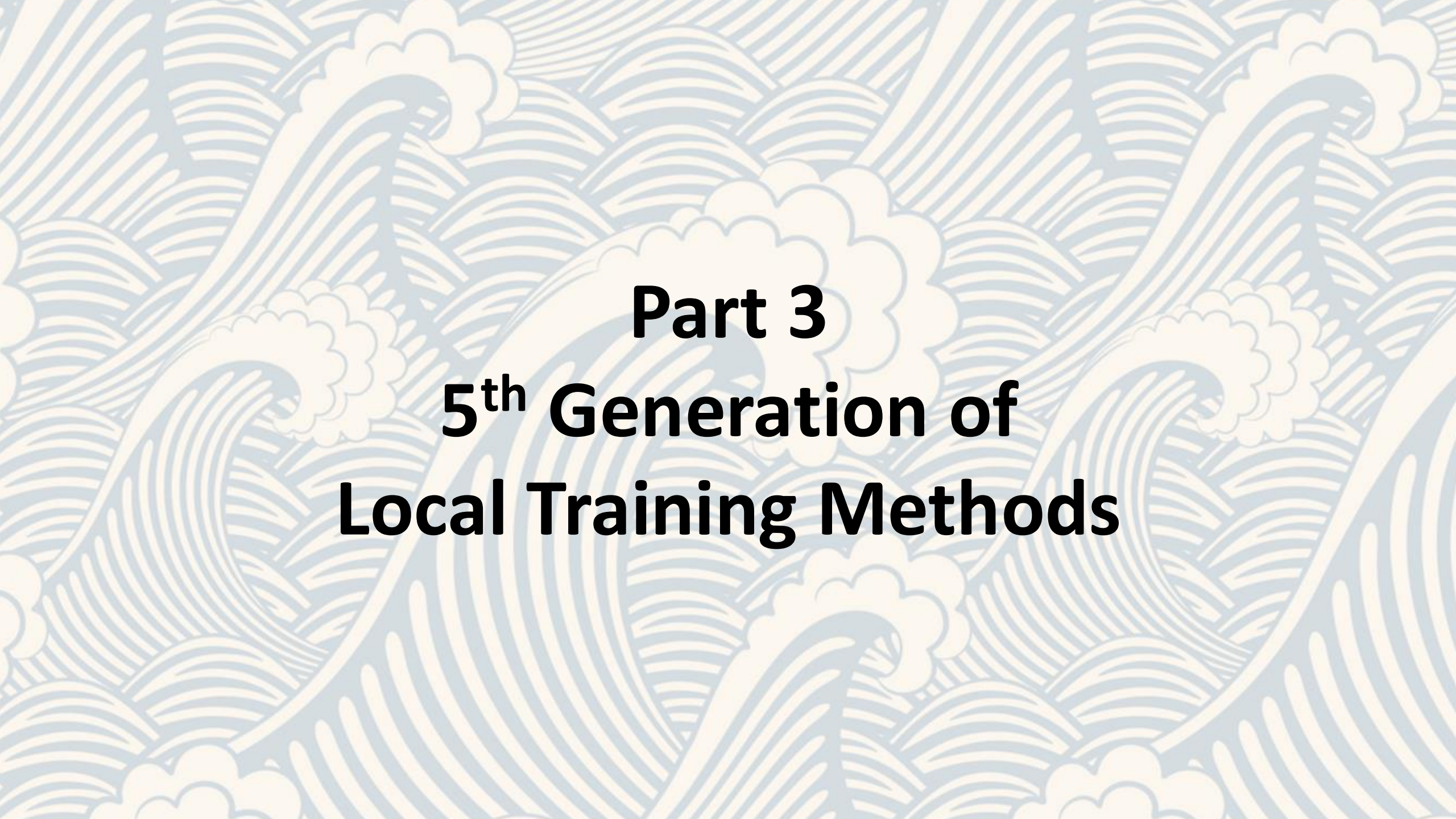
Generation 4: Linear

“Heterogeneous data is allowed, but the rate at best matches that of GD”



Generation 3

Generation 4



Part 3

5th Generation of Local Training Methods

Brief History of Local Training Methods

Generation 5: Accelerated

“Communication complexity is better than GD for heterogeneous data”



In practice, local training significantly improves communication efficiency.

However, there is no theoretical result explaining this!

Is the situation hopeless, or can we show/prove that local training helps?

Key Property of 5th Generation Local Training Methods

Communication complexity
of 4th generation
local training methods

$$\mathcal{O} \left(\frac{L}{\mu} \log \frac{1}{\varepsilon} \right)$$

Communication complexity
of 5th generation
local training methods

$$\mathcal{O} \left(\sqrt{\frac{L}{\mu}} \log \frac{1}{\varepsilon} \right)$$

ProxSkip: Yes! Local Gradient Steps Provably Lead to Communication Acceleration! Finally![†]

Konstantin Mishchenko¹ Grigory Malinovsky² Sebastian Stich³ Peter Richtárik²

Abstract

We introduce **ProxSkip**—a surprisingly simple and provably efficient method for minimizing the sum of a smooth (f) and an expensive nonsmooth proximal (ψ) function. The canonical approach to solving such problems is via the proximal gradient descent (**ProxGD**) algorithm, which is based on the evaluation of the gradient of f and the prox operator of ψ in each iteration. In this work we are specifically interested in the regime in which the evaluation of prox is costly relative to the evaluation of the gradient, which is the case in many applications. **ProxSkip** allows for the expensive prox operator to be skipped in most iterations: its iteration complexity is $\mathcal{O}(\kappa \log 1/\epsilon)$, where κ is the condition number of f , the number of evaluations is $\mathcal{O}(\sqrt{\kappa} \log 1/\epsilon)$ only. Our motivation comes from federated learning, where the evaluation of the gradient operator corresponds to a local GD step independently on all devices, and evaluation of prox corresponds to expensive communication in the form of gradient aggregation. In this context, **ProxSkip** offers a provable *acceleration* of communication complexity. Unlike other local gradient-type methods such as **FedAvg**, **SCAFFOLD**, **S-Local-GD** and others, whose theoretical communication complexity is worse than, or at best matching, that of **ProxGD** in the heterogeneous data regime, we achieve a provable and large improvement with heterogeneous data-bounding assumptions.

where $f: \mathbb{R}^d \rightarrow \mathbb{R}$ is a smooth function, and $\psi: \mathbb{R}^d \rightarrow \mathbb{R} \cup \{+\infty\}$ is a proper, closed and convex regularizer.

Such problem are ubiquitous, and appear in numerous applications associated with virtually all areas of science and engineering, including signal processing (Combettes & Pesquet, 2009), image processing (Luke, 2020), data science (Parikh & Boyd, 2014) and machine learning (Shalev-Shwartz & Ben-David, 2014).

1.1. Proximal gradient descent

[†] Please accept our apologies, our excitement apparently spilled over into the title. If we were to choose a more scholarly title for this work, it would be *ProxSkip: Breaking the Communication Barrier of Local Gradient Methods.*

1. Introduction

We study optimization problems of the form

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

¹CNRS, ENS, Inria Sierra, Paris, France ²Computer Science, King Abdullah University of Science and Technology, Thuwal, Saudi Arabia ³CISPA Helmholtz Center for Information Security, Saarbrücken, Germany. Correspondence to: Peter Richtárik <peter.richtarik@kaust.edu.sa>.

[†] Please accept our apologies, our excitement apparently spilled over into the title. If we were to choose a more scholarly title for this work, it would be *ProxSkip: Breaking the Communication Barrier of Local Gradient Methods.*

$\text{prox}_{\gamma\psi}$. This is the case for many regularizers, including the L_1 norm ($\psi(x) = \|x\|_1$), the L_2 norm ($\psi(x) = \|x\|_2^2$), and elastic net (Zhou & Hastie, 2005). For many further examples, we refer the reader to the books (Parikh & Boyd, 2014; Beck, 2017).

1.2. Expensive proximity operators

However, in this work we are interested in the situation when the evaluation of the *proximity operator* is *expensive*. That is, we assume that the computation of $\text{prox}_{\gamma\psi}$ (the backward step) is costly relative to the evaluation of the gradient of f (the forward step).

A conceptually simple yet rich class of expensive proximity operators arises from regularizers ψ encoding a

The Beginning



ICML
International Conference
On Machine Learning



Konstantin Mishchenko, Grigory Malinovsky, Sebastian Stich and P.R.
ProxSkip: Yes! Local Gradient Steps Provably Lead to Communication Acceleration! Finally!
ICML 2022

Brief History of Local Training Methods

Generation 5: Accelerated I

“Communication complexity is better than GD for heterogeneous data”

02/2022

ProxSkip



Konstantin Mishchenko, Grigory Malinovsky, Sebastian Stich and P.R.

ProxSkip: Yes! Local Gradient Steps Provably Lead to Communication Acceleration! Finally!

ICML 2022

07/2022

APDA; APDA-Inexact



Abdurakhmon Sadiev, Dmitry Kovalev and P.R.

Communication Acceleration of Local Gradient Methods via an Accelerated Primal-Dual Algorithm with Inexact Prox

NeurIPS 2022

Similar to FedProx (Li et al, 2018), but provably accelerates due to local training

07/2022

ProxSkip-LSVRG



Grigory Malinovsky, Kai Yi and P.R.

Variance Reduced ProxSkip: Algorithm, Theory and Application to Federated Learning

NeurIPS 2022

07/2022

RandProx



Laurent Condat and P.R.

RandProx: Primal-Dual Optimization Algorithms with Randomized Proximal Updates

ICLR 2023

Brief History of Local Training Methods

Generation 5: Accelerated II

“Communication complexity is better than GD for heterogeneous data”

10/2022

GradSkip



Artavazd Maranjyan, Mher Safaryan and P.R.

GradSkip: Communication-Accelerated Local Gradient Methods with Better Computational Complexity

TMLR 2025

10/2022

Compressed-
Scaffnew



Laurent Condat, Ivan Agarský and P.R.

Provably Doubly Accelerated Federated Learning: The First Theoretically Successful Combination of Local Training and Compressed Communication

arXiv:2210.13277, 2022

12/2022

5GCS



Michal Grudzien, Grigory Malinovsky and P.R.

Can 5th Generation Local Training Methods Support Client Sampling? Yes!

AISTATS 2023

02/2023

TAMUNA



Laurent Condat, Grigory Malinovsky and P.R.

TAMUNA: Accelerated Federated Learning with Local Training and Partial Participation

arXiv:2302.09832, 2023

Brief History of Local Training Methods

Generation 5: Accelerated III

“Communication complexity is better than GD for heterogeneous data”

07/2024

LoCoDL



Laurent Condat, Artavazd Maranjyan, P.R.

LoCoDL: Communication-Efficient Distributed Learning with Local Training and Compression

ICLR 2025

01/2026

BiCoLoR



Laurent Condat, Artavazd Maranjyan, P.R.

BiCoLoR: Communication-Efficient Optimization with Bidirectional Compression and Local Training

arXiv:2601.12400, 2026

Brief History of Local Training Methods: Overview

Generation 5: Accelerated

	# Comm. Rounds	Local Optimizer	# Local Training Steps	Total Complexity (Comm. + Comp.)	Client Sampling?	Comm. Compression?	Supports Decentralized Setup?	Key Insight
ProxSkip 2/22, ICML 22	$\mathcal{O}\left(\sqrt{\frac{L}{\mu}} \log \frac{1}{\varepsilon}\right)$	GD	$\sqrt{\frac{L}{\mu}}$	=	✗	✗	✓	First 5th generation local training method
APDA-Inexact 7/22, NeurIPS 22	$\mathcal{O}\left(\sqrt{\frac{L}{\mu}} \log \frac{1}{\varepsilon}\right)$	any	better	better	✗	✗	✓	Can use more powerful local solvers which take fewer local GD-type steps
VR-ProxSkip 7/22, NeurIPS 22	worse	VR-SGD	worse	better	✗	✗	✗	Running variance reduced SGD locally can lead to better total complexity than ProxSkip
RandProx 7/22	$\mathcal{O}\left(\sqrt{\frac{L}{\mu}} \log \frac{1}{\varepsilon}\right)$	GD	$\sqrt{\frac{L}{\mu}}$	=	✗	✗	✓	ProxSkip = VR mechanism for compressing the prox
GradSkip 10/22	$\mathcal{O}\left(\sqrt{\frac{L}{\mu}} \log \frac{1}{\varepsilon}\right)$	GD	better	better	✗	✗	✗	Workers containing less important data can do fewer local training steps!
Compressed Scaffnew 10/22 LoCoDL 7/24	worse	GD	better worse	better	✗	✓	✗	Can compress uplink, leads to better overall communication complexity than ProxSkip.
5GCS 10/22	worse	any	$\sqrt{\frac{L}{\mu}}$	worse	✓	✗	✗	Can do client sampling
TAMUNA 2/23	worse	GD / SGD	better worse	better	✓	✓	✗	First 5th gen LT method that can do CS + CC!
BiCoLoR 1/26				better	✓	bidirectional compression	✗	First 5th generation local training method supporting bidirectional compression

ProxSkip: Local Training Provably Leads to Communication Acceleration



Konstantin Mishchenko, Grigory Malinovsky, Sebastian Stich and P.R.

ProxSkip: Yes! Local Gradient Steps Provably Lead to Communication Acceleration! Finally!

ICML 2022

Federated Learning: ProxSkip vs Baselines

Table 1. The performance of federated learning methods employing multiple local gradient steps in the strongly convex regime.

	method	# local steps per round	# floats sent per round	stepsize on client i	linear rate?	# rounds	rate better than GD?
Gen 3	GD (Nesterov, 2004)	1	d	$\frac{1}{L}$	✓	$\tilde{\mathcal{O}}(\kappa)$ ^(c)	✗
	LocalGD (Khaled et al., 2019; 2020)	τ	d	$\frac{1}{\tau L}$	✗	$\mathcal{O}\left(\frac{G^2}{\mu n \tau \varepsilon}\right)$ ^(d)	✗
Gen 4	Scaffold (Karimireddy et al., 2020)	τ	$2d$	$\frac{1}{\tau L}$ ^(e)	✓	$\tilde{\mathcal{O}}(\kappa)$ ^(c)	✗
	S-Local-GD ^(a) (Gorbunov et al., 2021)	τ	$d < \# < 2d$ ^(f)	$\frac{1}{\tau L}$	✓	$\tilde{\mathcal{O}}(\kappa)$	✗
	FedLin ^(b) (Mitra et al., 2021)	τ_i	$2d$	$\frac{1}{\tau_i L}$	✓	$\tilde{\mathcal{O}}(\kappa)$ ^(c)	✗
Gen 5	Scaffnew ^(g) (this work) for any $p \in (0, 1]$	$\frac{1}{p}$ ^(h)	d	$\frac{1}{L}$	✓	$\tilde{\mathcal{O}}\left(p\kappa + \frac{1}{p}\right)$ ^(c)	✓ (for $p > \frac{1}{\kappa}$)
	Scaffnew ^(g) (this work) for optimal $p = \frac{1}{\sqrt{\kappa}}$	$\sqrt{\kappa}$ ^(h)	d	$\frac{1}{L}$	✓	$\tilde{\mathcal{O}}(\sqrt{\kappa})$ ^(c)	✓

^(a) This is a special case of S-Local-SVRG, which is a more general method presented in (Gorbunov et al., 2021). S-Local-GD arises as a special case when full gradient is computed on each client.

^(b) FedLin is a variant with a fixed but different number of local steps for each client. Earlier method S-Local-GD has the same update but random loop length.

^(c) The $\tilde{\mathcal{O}}$ notation hides logarithmic factors.

^(d) G is the level of dissimilarity from the assumption $\frac{1}{n} \sum_{i=1}^n \|\nabla f_i(x)\|^2 \leq G^2 + 2LB^2 (f(x) - f_*)$, $\forall x$.

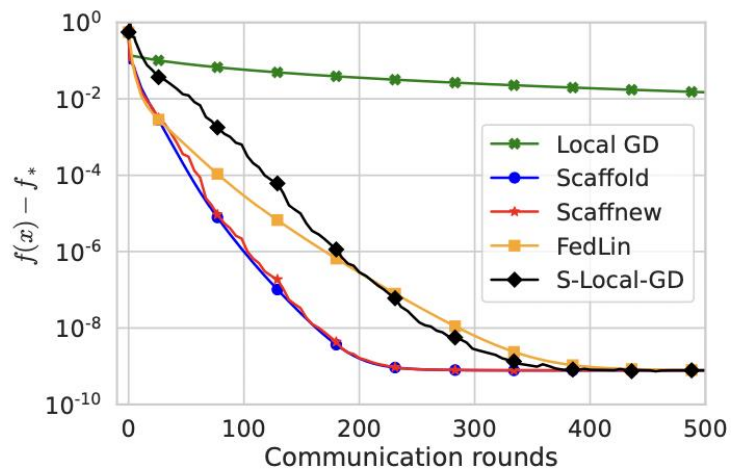
^(e) We use Scaffold's cumulative local-global stepsize $\eta_l \eta_g$ for a fair comparison.

^(f) The number of sent vectors depends on hyper-parameters, and it is randomized.

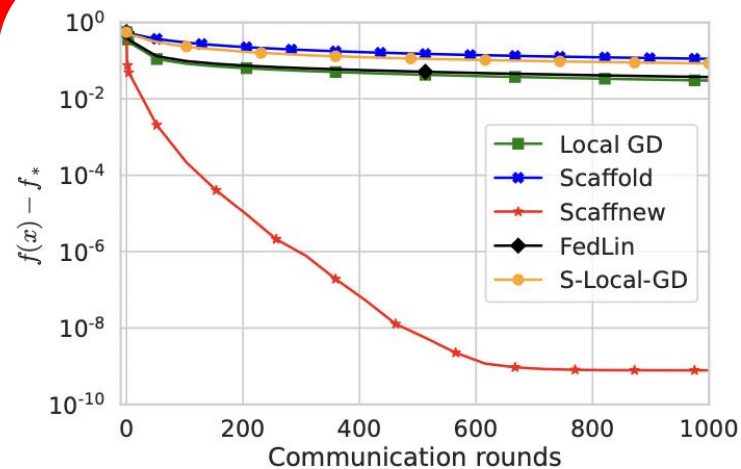
^(g) Scaffnew (Algorithm 2) = ProxSkip (Algorithm 1) applied to the consensus formulation (6) + (7) of the finite-sum problem (5).

^(h) ProxSkip (resp. Scaffnew) takes a *random* number of gradient (resp. local) steps before prox (resp. communication) is computed (resp. performed). What is shown in the table is the *expected* number of gradient (resp. local) steps.

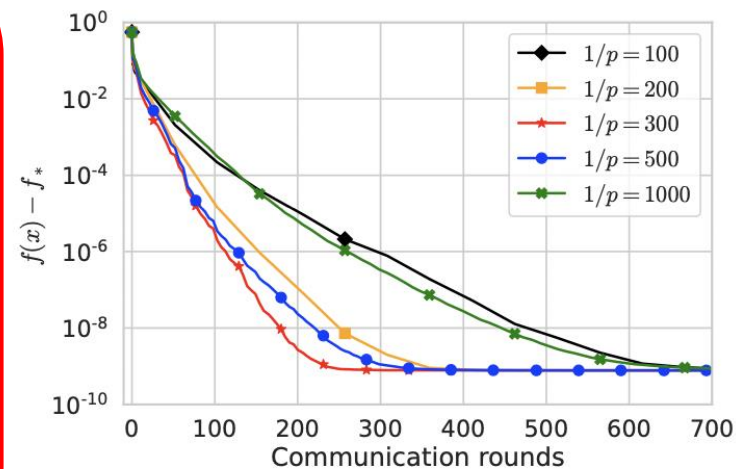
ProxSkip + Deterministic Gradients



(a) tuned hyper-parameters



(b) theoretical hyper-parameters



(c) different options of p

Figure 1. Deterministic Case. Comparison of **Scaffnew** to other local update methods that tackle data-heterogeneity and to **LocalGD**. In (a) we compare communication rounds with optimally tuned hyper-parameters. In (b), we compare communication rounds with the algorithm parameters set to the best theoretical stepsizes used in the convergence proofs. In (c), we compare communication rounds with the algorithm stepsize set to the best theoretical stepsize and different options of parameter p .

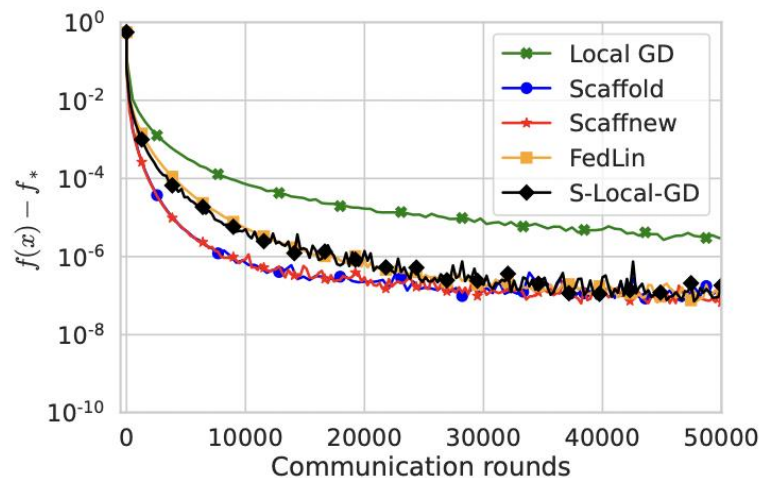
L2-regularized logistic regression:

$$f(x) = \frac{1}{n} \sum_{i=1}^n \log(1 + \exp(-b_i a_i^\top x)) + \frac{\lambda}{2} \|x\|^2$$

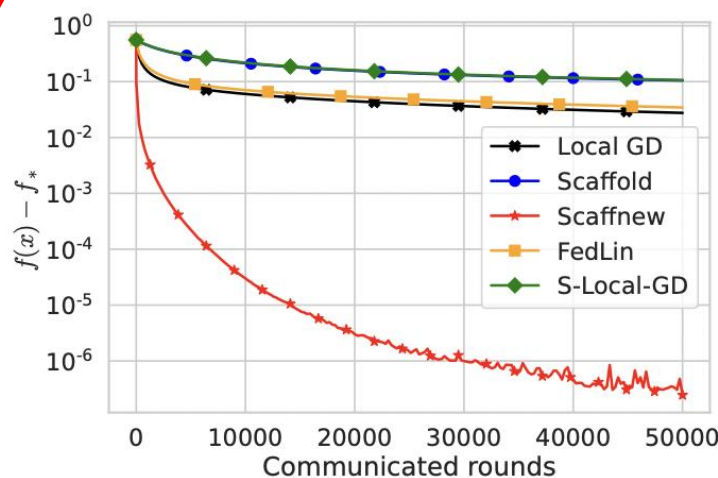
$$a_i \in \mathbb{R}^d, b_i \in \{-1, +1\}, \lambda = L/10^4$$

w8a dataset from LIBSVM library (Chang & Lin, 2011)

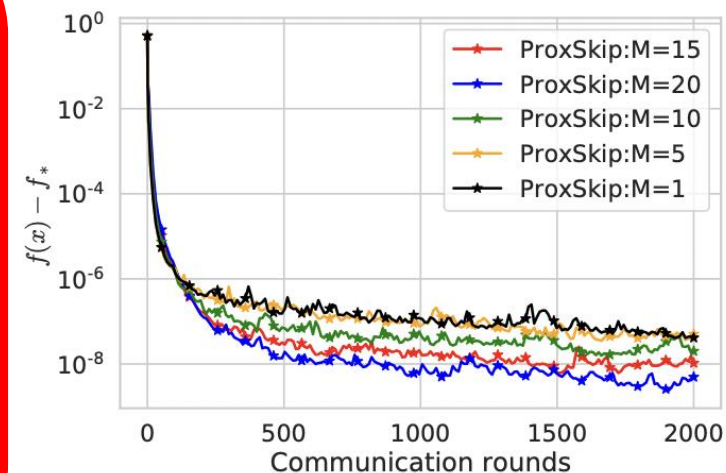
ProxSkip + Stochastic Gradients



(a) tuned hyper-parameters



(b) theoretical hyper-parameters



(c) different number of clients

Figure 2. Stochastic Case. Comparison of **Scaffnew** to other local update methods that tackle data-heterogeneity and to **LocalSGD**. In (a) we compare communication rounds with optimally tuned hyper-parameters. In (b), we compare communication rounds with the algorithm parameters set to the best theoretical stepsizes used in the convergence proofs. In (c), we compare communication rounds with the algorithm parameters set to the best theoretical stepsizes used in the convergence proofs and different number of clients.

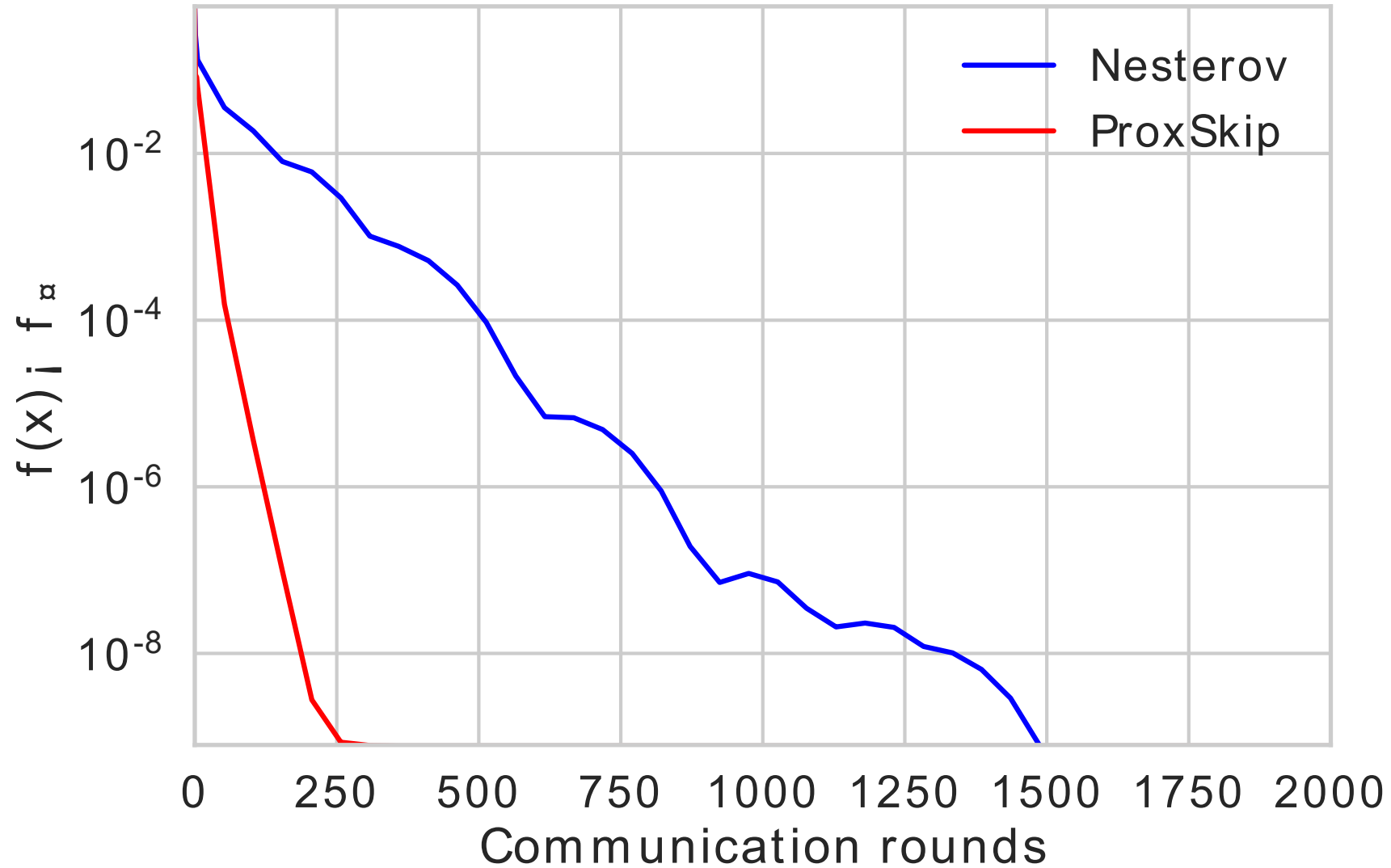
L2-regularized logistic regression:

$$f(x) = \frac{1}{n} \sum_{i=1}^n \log(1 + \exp(-b_i a_i^\top x)) + \frac{\lambda}{2} \|x\|^2$$

$$a_i \in \mathbb{R}^d, b_i \in \{-1, +1\}, \lambda = L/10^4$$

w8a dataset from LIBSVM library (Chang & Lin, 2011)

ProxSkip vs Nesterov's Accelerated GD



Consensus Reformulation

Original problem:
optimization in $\mathbb{R}^d$

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

Consensus reformulation:
optimization in $\mathbb{R}^{nd}$

$$\min_{x_1, \dots, x_n \in \mathbb{R}^d} \left\{ \frac{1}{n} \sum_{i=1}^n f_i(x_i) + \psi(x_1, \dots, x_n) \right\}$$

Bad: non-differentiable

Good: Indicator function of a nonempty closed convex set

$$\psi(x_1, \dots, x_n) \stackrel{\text{def}}{=} \begin{cases} 0, & \text{if } x_1 = \dots = x_n, \\ +\infty, & \text{otherwise.} \end{cases}$$

Consensus Reformulation

Original problem:
optimization in $\mathbb{R}^d$

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

Bad: non-differentiable

Good: proper closed convex

Consensus reformulation:
optimization in $\mathbb{R}^{nd}$

$$\min_{x_1, \dots, x_n \in \mathbb{R}^d} \left\{ \frac{1}{n} \sum_{i=1}^n f_i(x_i) + \psi(x_1, \dots, x_n) \right\}$$

$\psi(x_1, \dots, x_n) : \mathbb{R}^{nd} \rightarrow \mathbb{R} \cup \{+\infty\}$
is a proper **closed convex** function

$\text{epi}(\psi) \stackrel{\text{def}}{=} \{(x, t) \mid \psi(x) \leq t\}$ The epigraph of ψ is a closed and convex set

Three Assumptions

The epigraph of ψ is a closed and convex set

$$\text{epi}(\psi) \stackrel{\text{def}}{=} \{(x, t) \in \mathbb{R}^d \times \mathbb{R} \mid \psi(x) \leq t\}$$

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

A1 f is μ -convex and L -smooth:

$$\frac{\mu}{2} \|x - y\|^2 \leq D_f(x, y) \leq \frac{L}{2} \|x - y\|^2$$

A2 $\psi : \mathbb{R}^d \rightarrow \mathbb{R} \cup \{+\infty\}$ is proper, closed, and convex

A3 ψ is proximable

Bregman divergence of f :

$$D_f(x, y) \stackrel{\text{def}}{=} f(x) - f(y) - \langle \nabla f(y), x - y \rangle$$

The proximal operator $\text{prox}_\psi : \mathbb{R}^d \rightarrow \mathbb{R}^d$ defined by

$$\text{prox}_\psi(x) \stackrel{\text{def}}{=} \arg \min_{u \in \mathbb{R}^d} \left(\psi(u) + \frac{1}{2} \|u - x\|^2 \right)$$

can be evaluated exactly (e.g., in closed form)

Key Method: Proximal Gradient Descent

proximal operator:

$$\text{prox}_\psi(x) \stackrel{\text{def}}{=} \arg \min_{u \in \mathbb{R}^d} \left(\psi(u) + \frac{1}{2} \|u - x\|^2 \right)$$



stepsize

$$x_t - \gamma \nabla f(x_t)$$



gradient operator

$$x \mapsto x - \gamma \nabla f(x)$$

Proximal Gradient Descent: Theory

f is μ -convex and L -smooth:
 $\frac{\mu}{2} \|x - y\|^2 \leq D_f(x, y) \leq \frac{L}{2} \|x - y\|^2$
 $\frac{L}{\mu}$ is the condition number of f

Theorem:

$$t \geq \frac{L}{\mu} \log \frac{1}{\varepsilon} \implies \|x_t - x_\star\|^2 \leq \varepsilon \|x_0 - x_\star\|^2$$

(for stepsize $\gamma = \frac{1}{L}$)

iterations

Error tolerance

$$x_\star \stackrel{\text{def}}{=} \arg \min_{x \in \mathbb{R}^d} f(x) + \psi(x)$$

What to do When the Prox is Expensive?

Can we somehow get away with
fewer evaluations of the proximity operator
in the Proximal GD method?

Approach 1



We'll skip ALL prox evaluations!



The method is NOT implementable!



Serves as an inspiration for Approach 2

Approach 2 (ProxSkip)



We'll skip MANY prox evaluations!



The method is implementable!

Approach 1:
Simple, Extreme but
Practically Useless Variant

Removing ψ via a Reformulation

$$\min_{x \in \mathbb{R}^d} f(x) - \langle h_\star, x \rangle$$

$$\begin{aligned} h_\star &\stackrel{\text{def}}{=} \nabla f(x_\star) \\ x_\star &\stackrel{\text{def}}{=} \arg \min_{x \in \mathbb{R}^d} f(x) + \psi(x) \end{aligned}$$



$x_\star$ is a solution of the above problem!

By the 1st order optimality conditions, the solution satisfies $\nabla f(x) - \nabla f(x_\star) = 0$



We do not know $h_\star = \nabla f(x_\star)$!

Apply Gradient Descent to the Reformulation

$$\begin{aligned} h_{\star} &\stackrel{\text{def}}{=} \nabla f(x_{\star}) \\ x_{\star} &\stackrel{\text{def}}{=} \arg \min_{x \in \mathbb{R}^d} f(x) + \psi(x) \end{aligned}$$

$$x_{t+1} = x_t - \gamma (\nabla f(x_t) - h_{\star})$$



We do not need to evaluate the prox of ψ at all!



We do not know $h_{\star}$ and hence can't implement the method!

Approach 2:

The ProxSkip Method

Idea: Try to “Learn” the Optimal Gradient Shift

$$x_{t+1} = x_t - \gamma (\nabla f(x_t) - h_t)$$

Desire: $h_t \rightarrow h_\star$



Perhaps we can learn $h_\star$ with only occasional access to ψ ?

ProxSkip: Bird's Eye View

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

1

$$\hat{x}_{t+1} = x_t - \gamma (\nabla f(x_t) - h_t)$$

2a

with probability $1 - p$ do

$$1 - p \approx 1$$

$$x_{t+1} = \hat{x}_{t+1}$$

$$h_{t+1} = h_t$$

2b

with probability p do

$$p \approx 0$$

evaluate $\text{prox}_{\frac{\gamma}{p}\psi}(?)$

$$x_{t+1} = ?$$

$$h_{t+1} = ?$$

ProxSkip: The Algorithm (Detailed View)

Algorithm 1 ProxSkip

1: stepsize $\gamma > 0$, probability $p > 0$, initial iterate $x_0 \in \mathbb{R}^d$, initial control variate $h_0 \in \mathbb{R}^d$, number of iterations $T \geq 1$
2: **for** $t = 0, 1, \dots, T - 1$ **do**
3: $\hat{x}_{t+1} = x_t - \gamma(\nabla f(x_t) - h_t)$ ◇ Take a gradient-type step adjusted via the control variate h_t
4: Flip a coin $\theta_t \in \{0, 1\}$ where $\text{Prob}(\theta_t = 1) = p$ ◇ Flip a coin that decides whether to skip the prox or not
5: **if** $\theta_t = 1$ **then**
6: $x_{t+1} = \text{prox}_{\frac{\gamma}{p}\psi}(\hat{x}_{t+1} - \frac{\gamma}{p}h_t)$ ◇ Apply prox, but only very rarely! (with small probability p)
7: **else**
8: $x_{t+1} = \hat{x}_{t+1}$ ◇ Skip the prox!
9: **end if**
10: $h_{t+1} = h_t + \frac{p}{\gamma}(x_{t+1} - \hat{x}_{t+1})$ ◇ Update the control variate h_t
11: **end for**

ProxSkip: Bounding the # of Iterations

Theorem:

f is μ -convex and L -smooth:
 $\frac{\mu}{2} \|x - y\|^2 \leq D_f(x, y) \leq \frac{L}{2} \|x - y\|^2$
 $\frac{L}{\mu}$ is the condition number of f

$$t \geq \max \left\{ \frac{L}{\mu}, \frac{1}{p^2} \right\} \log \frac{1}{\varepsilon} \implies \mathbb{E} [\Psi_t] \leq \varepsilon \Psi_0$$

iterations

p = probability of
evaluating the prox

Lyapunov function:

$$\Psi_t \stackrel{\text{def}}{=} \|x_t - x_\star\|^2 + \frac{1}{L^2 p^2} \|h_t - h_\star\|^2$$

ProxSkip: Optimal Prox-Evaluation Probability

Since in each iteration we evaluate the prox with probability p , the expected number of prox evaluations after t iterations is:

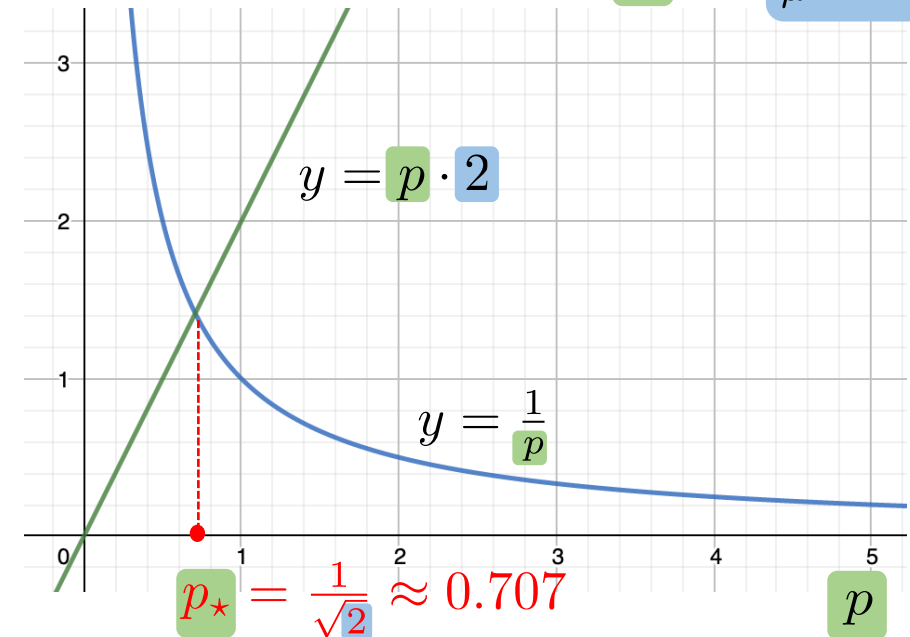
$\frac{L}{\mu}$ is the condition number of f

$$p \cdot t = p \cdot \max \left\{ \frac{L}{\mu}, \frac{1}{p^2} \right\} \cdot \log \frac{1}{\varepsilon} = \max \left\{ p \cdot \frac{L}{\mu}, \frac{1}{p} \right\} \cdot \log \frac{1}{\varepsilon}$$

Minimized for p satisfying $p \cdot \frac{L}{\mu} = \frac{1}{p}$

$$\Rightarrow p_{\star} = \frac{1}{\sqrt{L/\mu}}$$

Computation of optimal $p_{\star}$ for $\frac{L}{\mu} = 2$



From Gradients to Stochastic Gradients

- As described, in ProxSkip each worker computes the **full gradient** of its local function
- It's often better to consider a **cheap stochastic approximation of the gradient** instead
 - We consider this extension in the paper
 - We provide theoretical convergence rates

$$\nabla f_i(x_t) \Rightarrow g_i(x_t)$$

Full gradient Stochastic gradient

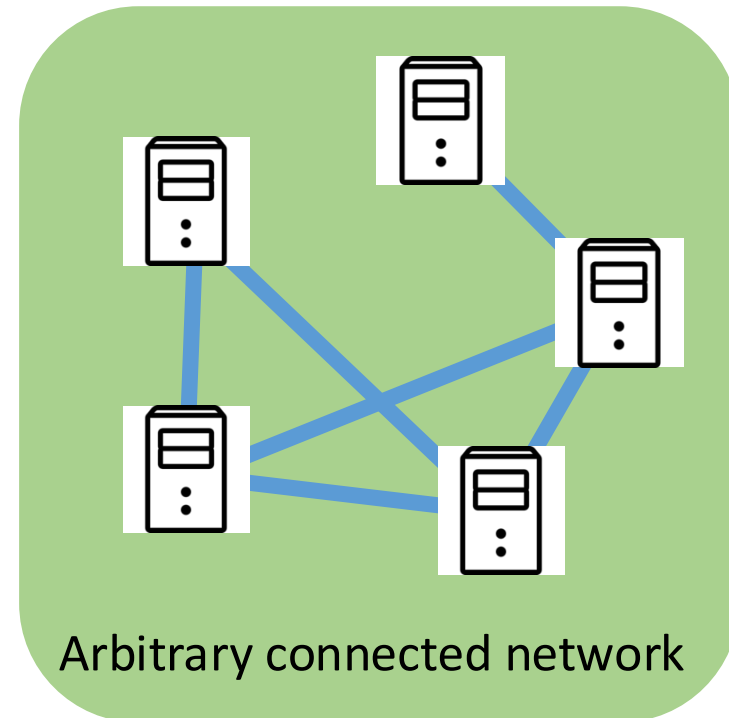
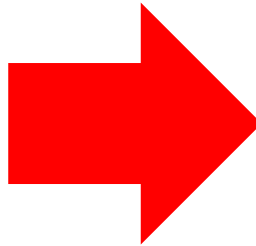
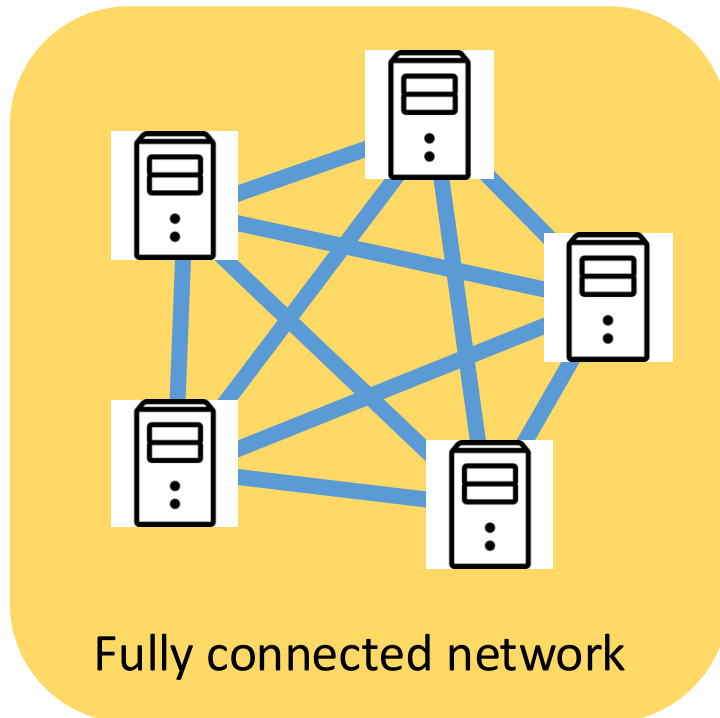
Assumptions:

(unbiasedness) $\mathbb{E}[g_{i,t}(x_t) \mid x_t] = \nabla f_i(x_t)$

(expected smoothness) $\mathbb{E}[\|g_{i,t}(x_t) - \nabla f(x_*)\|^2 \mid x_t] \leq 2AD_f(x_t, x_*) + C$
(Gower et al, 2019)

From Fully Connected Networks to Arbitrary Connected Networks

- In each communication round of ProxSkip, **each worker sends messages to all other workers** (e.g., through a server).
 - We can think of ProxSkip workers as the nodes of a **fully-connected network**.
 - In each communication round, all **workers communicate with their neighbors**.
- In the paper we provide extension to **arbitrary connected networks**.



“Philosophically” Related Literature

Gradient Sliding by **George Lan**



Guanghui Lan

Gradient Sliding for Composite Optimization

Mathematical Programming 2016



Guanghui Lan and Yi Zhou

Conditional Gradient Sliding for Convex Optimization

SIAM Journal on Optimization 2016



Guanghui Lan and Yuyuan Ouyang

Accelerated Gradient Sliding for Structured Convex Optimization

COAP 2022



Dmitry Kovalev, Aleksandr Beznosikov, Ekaterina Dmitrievna Borodich, Alexander Gasnikov, Gesualdo Scutari

Optimal Gradient Sliding and its Application to Optimal Distributed Optimization Under Similarity

NeurIPS 2022

GradSkip: Clients with Less Important Data can do Less Local Training

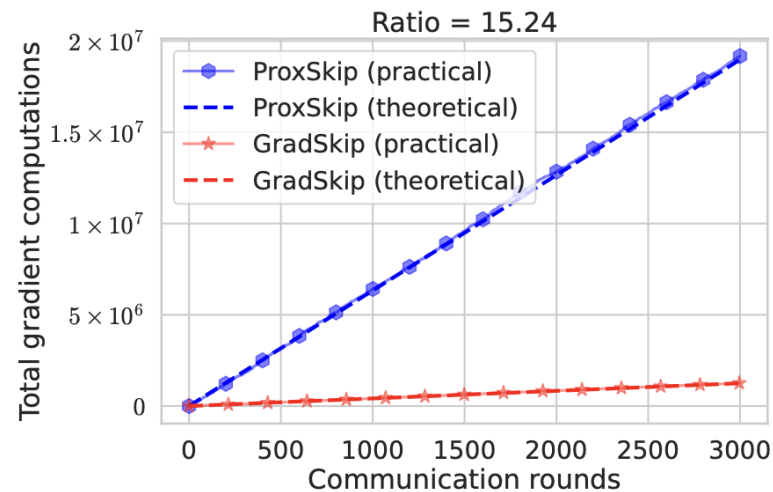
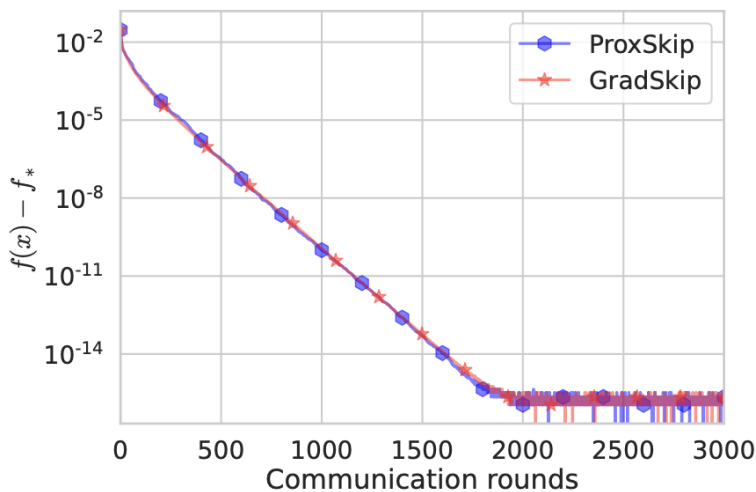
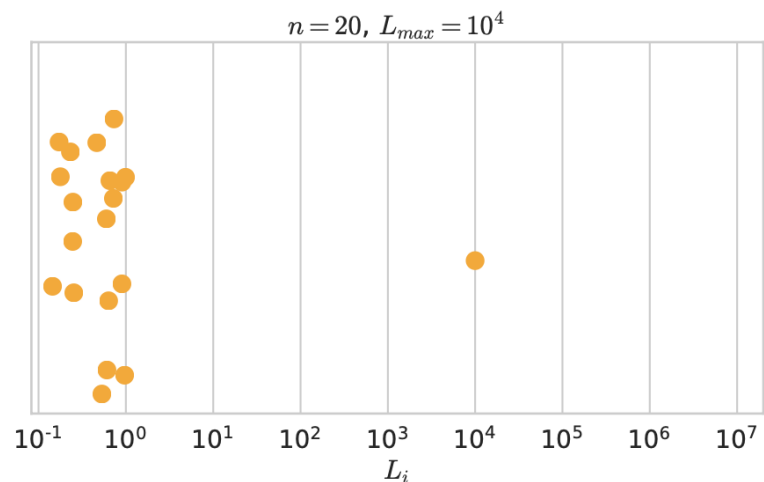


Artavazd Maranjyan, Mher Safaryan and P.R.

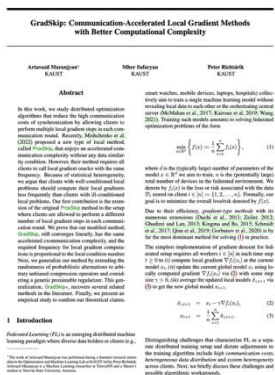
GradSkip: Communication-Accelerated Local Gradient Methods with Better Computational Complexity

TMLR 2025 (arXiv 2210.16402)

GradSkip



arXiv:2210.16407v1 [cs.LG] 28 Oct 2022



Algorithm 2 GradSkip+

- 1: **Parameters:** stepsize $\gamma > 0$, compressors $\mathcal{C}_\omega \in \mathbb{B}^d(\omega)$ and $\mathcal{C}_\Omega \in \mathbb{B}^d(\Omega)$.
- 2: **Input:** initial iterate $x_0 \in \mathbb{R}^d$, initial control variate $h_0 \in \mathbb{R}^d$, number of iterations $T \geq 1$.
- 3: **for** $t = 0, 1, \dots, T - 1$ **do**
- 4: $\hat{h}_{t+1} = \nabla f(x_t) - (\mathbf{I} + \Omega)^{-1} \mathcal{C}_\Omega (\nabla f(x_t) - h_t)$ ◊ Update the shift $\hat{h}_{i,t}$ via shifted compression
- 5: $\hat{x}_{t+1} = x_t - \gamma (\nabla f(x_t) - \hat{h}_{t+1})$ ◊ Update the iterate $\hat{x}_{i,t}$ via shifted gradient step
- 6: $\hat{g}_t = \frac{1}{\gamma(1+\omega)} \mathcal{C}_\omega \left(\hat{x}_{t+1} - \text{prox}_{\gamma(1+\omega)\psi} \left(\hat{x}_{t+1} - \gamma(1+\omega) \hat{h}_{t+1} \right) \right)$ ◊ Estimate the proximal gradient
- 7: $x_{t+1} = \hat{x}_{t+1} - \gamma \hat{g}_t$ ◊ Update the main iterate $x_{i,t}$
- 8: $h_{t+1} = \hat{h}_{t+1} + \frac{1}{\gamma(1+\omega)} (x_{t+1} - \hat{x}_{t+1})$ ◊ Update the main shift $h_{i,t}$
- 9: **end for**



Artavazd Maranjyan, Mher Safaryan and P.R.

GradSkip: Communication-Accelerated Local Gradient Methods with Better Computational Complexity

TMLR 2025

RandProx: Primal–Dual Optimization Algorithms with Randomized Proximal Updates

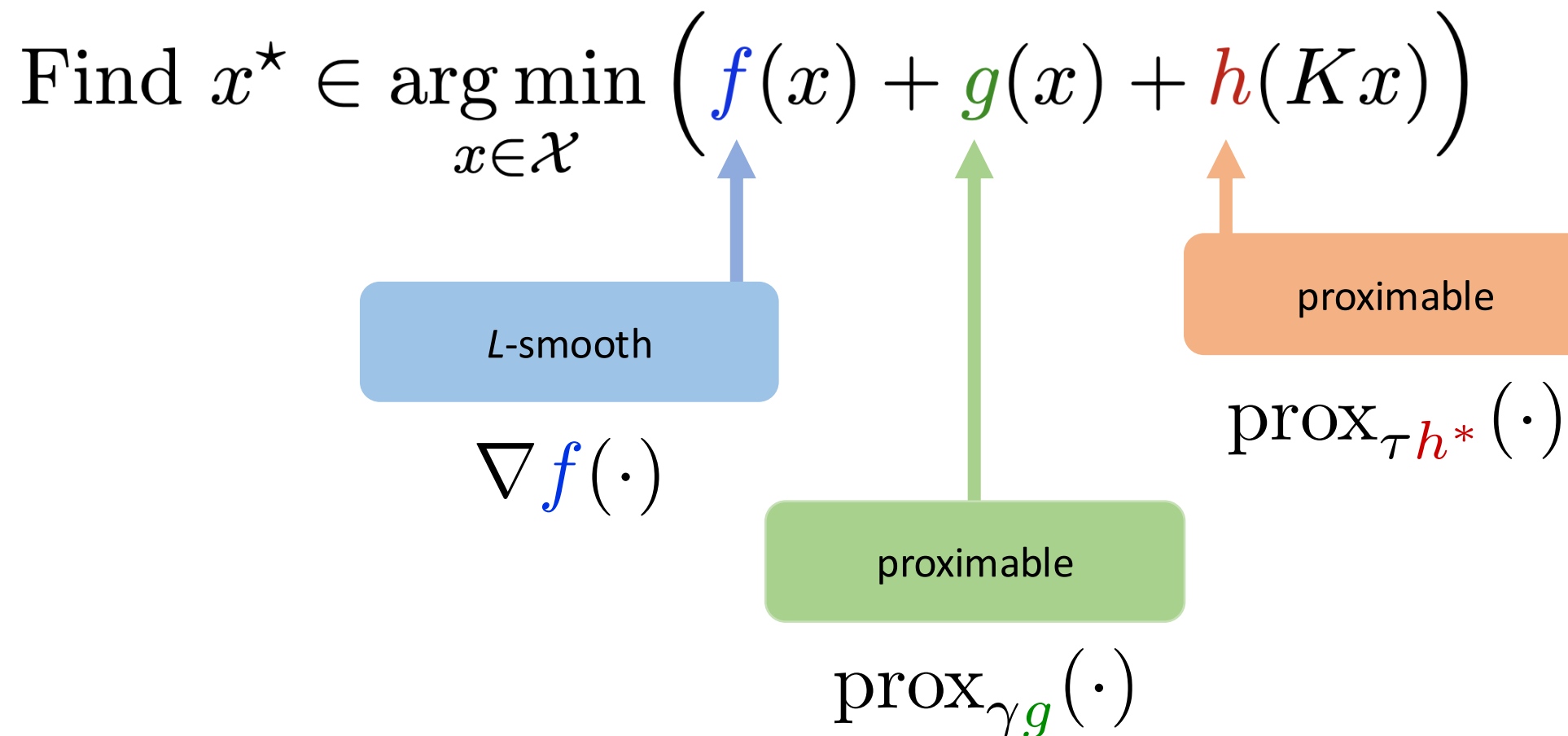


Laurent Condat and P.R.

RandProx: Primal–Dual Optimization Algorithms with Randomized Proximal Updates

ICLR 2023 (arXiv 2207.12891)

Minimization of the Sum of Three Functions



PDDY: Primal-Dual Davis Yin Splitting

$$\text{Find } x^* \in \arg \min_{x \in \mathcal{X}} \left(\textcolor{blue}{f}(x) + \textcolor{green}{g}(x) + \textcolor{red}{h}(Kx) \right)$$

Algorithm **PDDY** [Salim et al., 2022b]

input: initial points $x^0 \in \mathcal{X}$, $u^0 \in \mathcal{U}$;

stepsizes $\gamma > 0$, $\tau > 0$

for $t = 0, 1, \dots$ **do**

$$\hat{x}^t := \text{prox}_{\gamma \textcolor{green}{g}} \left(x^t - \gamma \nabla \textcolor{blue}{f}(x^t) - \gamma K^* u^t \right)$$

$$\textcolor{yellow}{u^{t+1} := \text{prox}_{\tau \textcolor{red}{h}^*} \left(u^t + \tau K \hat{x}^t \right)}$$

$$x^{t+1} := \hat{x}^t - \gamma K^* (u^{t+1} - u^t)$$

end for



Adil Salim, Laurent Condat, Konstantin Mishchenko and P.R.

Dualize, Split, Randomize: Toward Fast Nonsmooth Optimization Algorithms

JOTA 2022

Special Cases of PDDY

$$\arg \min_{x \in \mathcal{X}} \left(\underset{f}{\textcolor{blue}{f}}(x) + \underset{g}{\textcolor{green}{g}}(x) + \underset{h}{\textcolor{red}{h}}(Kx) \right)$$

f	g	h	K	Deterministic algorithm
any	any	any	any	PDDY
any	0	any	any	PAPC
any	0	any	Id	forward-backward (FB)
any	0	$\iota_{\{b\}}$	any	PAPC
0	any	any	any	Chambolle–Pock (CP)
0	any	any	Id	ADMM
any	any	any	Id	Davis–Yin (DY)

[Salim, Condat, Mishchenko & R 2022]

[Drori, Sabach & Teboulle 2015]

[Drori, Sabach & Teboulle 2015]

[Chambolle & Pock 2011]

[Boyd et al 2011]

[Davis & Yin 2017]

From PDDY to RandProx: Compress the Prox!

Algorithm **PDDY** [Salim et al., 2022b]

input: initial points $x^0 \in \mathcal{X}$, $u^0 \in \mathcal{U}$;
 stepsizes $\gamma > 0$, $\tau > 0$
for $t = 0, 1, \dots$ **do**
 $\hat{x}^t := \text{prox}_{\gamma g}(x^t - \gamma \nabla f(x^t) - \gamma K^* u^t)$
 $u^{t+1} := \text{prox}_{\tau h^*}(u^t + \tau K \hat{x}^t)$
 $x^{t+1} := \hat{x}^t - \gamma K^*(u^{t+1} - u^t)$
end for

Algorithm **RandProx** [new]

input: initial points $x^0 \in \mathcal{X}$, $u^0 \in \mathcal{U}$;
 stepsizes $\gamma > 0$, $\tau > 0$; $\omega \geq 0$
for $t = 0, 1, \dots$ **do**
 $\hat{x}^t := \text{prox}_{\gamma g}(x^t - \gamma \nabla f(x^t) - \gamma K^* u^t)$
 $u^{t+1} := u^t + \frac{1}{1+\omega} \mathcal{R}^t(\text{prox}_{\tau h^*}(u^t + \tau K \hat{x}^t) - u^t)$
 $x^{t+1} := \hat{x}^t - \gamma (1 + \omega) K^*(u^{t+1} - u^t)$
end for

Variance reduction technique known as EF21:



P. R., Igor Sokolov and Ilyas Fatkhullin

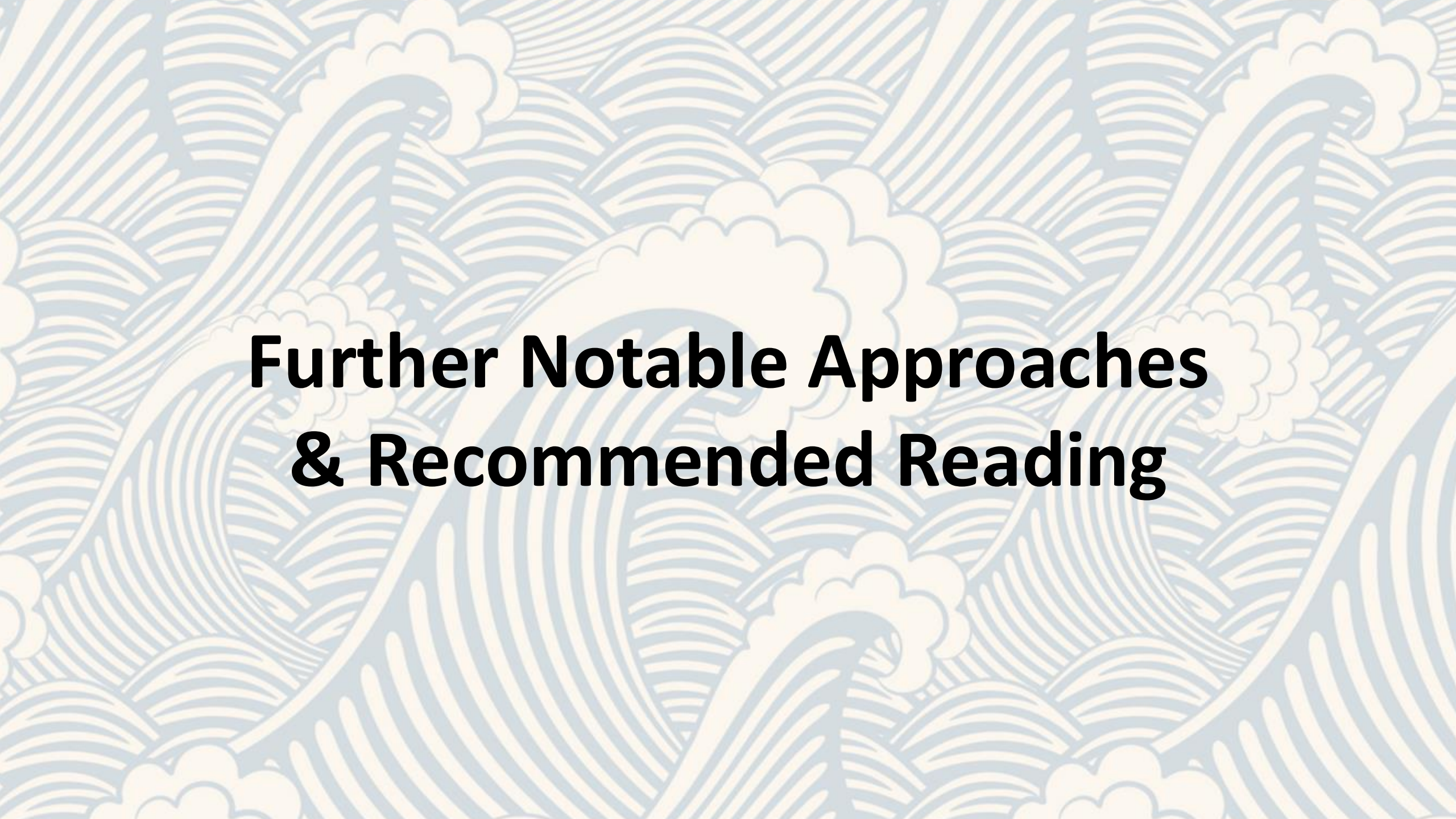
EF21: A New, Simpler, Theoretically Better, and Practically Faster Error Feedback

NeurIPS 2021

Compression operator:

$$\mathbb{E} [\mathcal{R}^t(x)] = x$$

$$\mathbb{E} [\|\mathcal{R}^t(x) - x\|^2] \leq \omega \|x\|^2$$



Further Notable Approaches & Recommended Reading

Distributed Proximal Point Method: (= FedProx with Full Client Participation)

All machines do local training:

$$x_i^{k+1} \approx \text{prox}_{\lambda f_i}(x^k) := \arg \min_x \left\{ f_i(x) + \frac{\lambda}{2} \|x - x^k\|^2 \right\}$$

Approximate proxies get averaged:

$$x^{k+1} = \frac{1}{n} \sum_{i=1}^n x_i^{k+1}$$

This is an **algorithm-agnostic (in terms of local training) interpolation** of

$$x^k \quad \text{and} \quad x_i^* := \arg \min_x f_i(x)$$

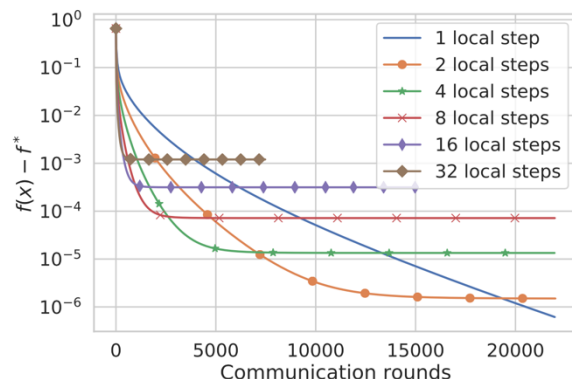
-  Generalizes LocalGD (for $\lambda = 0$) & K steps of GD used to solve prox approximately
-  One can use arbitrary solver for local training
-  Has theoretical guarantees
-  Empirically more robust than FedAvg
-  Li et al (2020) theory does not improve on distributed SGD
-  No theoretical benefit from LT



FedProx

T. Li, A K Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, V. Smith
Federated Optimization in Heterogeneous Networks
MLSys 2020

Distributed Local Fixed-Point Method



Fixed-point formulation:

Find a fixed-point of the operator $\mathcal{T} := \frac{1}{n} \sum_{i=1}^n \mathcal{T}_i$

$$\mathcal{T}_i : \mathbb{R}^d \rightarrow \mathbb{R}^d$$

examples: $\mathcal{T}_i(x) = x - \gamma \nabla f_i(x)$

$$\mathcal{T}_i(x) = \text{prox}_{\lambda f_i}(x)$$

Relaxation parameter

$$x_i^{k+1} = ((1 - \lambda) \text{Id} + \lambda \mathcal{T}_i)^K (x^k)$$

Number of local FPM iterations

- ✓ Generalizes & improves:
 - LocalGD $\mathcal{T}_i(x) = x - \gamma \nabla f_i(x)$ $\lambda = 1$
 - FedProx $\mathcal{T}_i(x) = \text{prox}_{\lambda f_i}(x)$ $\lambda = 1$ $K = 1$

- ✓ ✓ Theoretical benefit from LT!

$$x^{k+1} = \frac{1}{n} \sum_{i=1}^n x_i^{k+1}$$

Local models get averaged



Grigory Malinovsky, Dmitry Kovalev, Elnur Gasanov, Laurent Condat, P.R.
From Local SGD to Local Fixed-Point Methods for Federated Learning
ICML 2020 (arXiv 2004.01442)

Further Recommended Reading (Prox, Local Training, 2nd-order Similarity & Acceleration)

12/2013
DANE



Ohad Shamir, Nati Srebro, and Tong Zhang
Communication-efficient distributed optimization using an approximate Newton-type method
ICML 2014

Seminal work introducing
2nd order similarity

8/2016
AIDE



Sashank J Reddi, Jakub Konecny, P.R., Barnabás Póczós, and Alex Smola
AIDE: Fast and communication efficient distributed optimization
arXiv 1608.06879

Accelerated DANE

10/2021
MARINA + Perm-K



Rafał Szlendak, Alexander Tyurin and P.R.
Permutation compressors for provably faster distributed nonconvex optimization
ICLR 2022

2022
Inex Acc SONATA



Ye Tian, Gesualdo Scutari, Tianyu Cao, and Alexander Gasnikov
Acceleration in distributed optimization under similarity
AISTATS 2022

MARINA (SOTA nonconvex
variance-reduced DCGD)
benefits from **similarity**

2022
Acc Extragrad



Dmitry Kovalev, Aleksandr Beznosikov, Ekaterina Borodich, Alexander Gasnikov, and Gesualdo Scutari
Optimal gradient sliding and its application to optimal distributed optimization under similarity
NeurIPS 2022

Further Recommended Reading (Prox, Local Training, 2nd-order Similarity & Acceleration)

9/2022

SPPM
SVRP
Catalyst SVRP



Ahmed Khaled and Chi Jin
Faster federated optimization under second-order similarity
ICLR 2023

Cool convergence proof of
SPPM under individual strong
convexity
(+ variance reduction +
acceleration)

2024

Acc SVRS



Dachao Lin, Yuze Han, Haishan Ye, and Zhihua Zhang
Stochastic distributed optimization under average second-order similarity: Algorithms and analysis
NeurIPS 2024

4/2024

FedRED



Xiaowen Jiang, Anton Rodomanov, and Sebastian U Stich
Federated optimization with doubly regularized drift correction

Latest & best results!

2024
S-DANE, Acc-S-DANE



Xiaowen Jiang, Anton Rodomanov, Sebastian U. Stich
Stabilized Proximal-Point Methods for Federated Optimization
NeurIPS 2024

Further Recommended Reading (Prox, Local Training, 2nd-order Similarity & Acceleration)

5/2024

SPPM, SPPM-LC, SPPM-NS,
SPPM-AS, SPPM*, ...



P.R., Abdurakhmon Sadiev, and Yury Demidovich
A unified theory of stochastic proximal point methods without smoothness
arXiv 2405.15941

Analysis of an array SPPM methods, including variance-reduced ones

5/2024



Abdurakhmon Sadiev, Laurent Condat, and P.R.
Stochastic proximal point methods for monotone inclusions under expected similarity
OPT 2024

SPPM for monotone inclusions

2/2025

SPPM



Zhirayr Tovmasyan, Grigory Malinovsky, Laurent Condat, and P.R.
Revisiting stochastic proximal point methods: generalized smoothness and similarity
arXiv 2502.03401

Selected Notable Works on Local Training I Won't Have Time to Talk About

5/2018

Local SGD



Sebastian U. Stich

Local SGD Converges Fast and Communicates Little

8/2018

Post-Local SGD



Tao Lin, Sebastian U. Stich, K K Patel, Martin Jaggi

Don't Use Large Minibatches, Use Local SGD

ICLR 2020

2/2020



Blake Woodworth, Kumar Kshitij Patel, Sebastian U. Stich, Zhen Dai, Brian Bullins, H. Brendan

McMahan, Ohad Shamir, Nathan Srebro

Is Local SGD Better than Minibatch SGD?

5/2020

FedSplit



Reese Pathak and Martin J. Wainwright

FedSplit: An algorithmic framework for fast federated optimization

arXiv 2005.05238

6/2020



Blake Woodworth, Kumar Kshitij Patel, Nathan Srebro

Minibatch vs Local SGD for Heterogeneous Distributed Learning

Quadratics: Local SGD > Minibatch SGD

Convex Case: Local SGD sometimes better



An approach to FL based on
operator splitting



Selected Notable Works on Local Training I Won't Have Time to Talk About

9/2021



Blake Woodworth
The Minimax Complexity of Distributed Optimization
PhD Thesis, 2021

Nice PhD theses

7/2025



Kumar Kshitij Patel
What Makes Local Updates Effective: The Role of Data Heterogeneity and Smoothness
PhD Thesis, 2025

1/2025

Local GD



Michael Crawshaw, Blake Woodworth, Mingrui Liu
Local Steps Speed up Local GD for Heterogeneous Distributed Logistic Regression

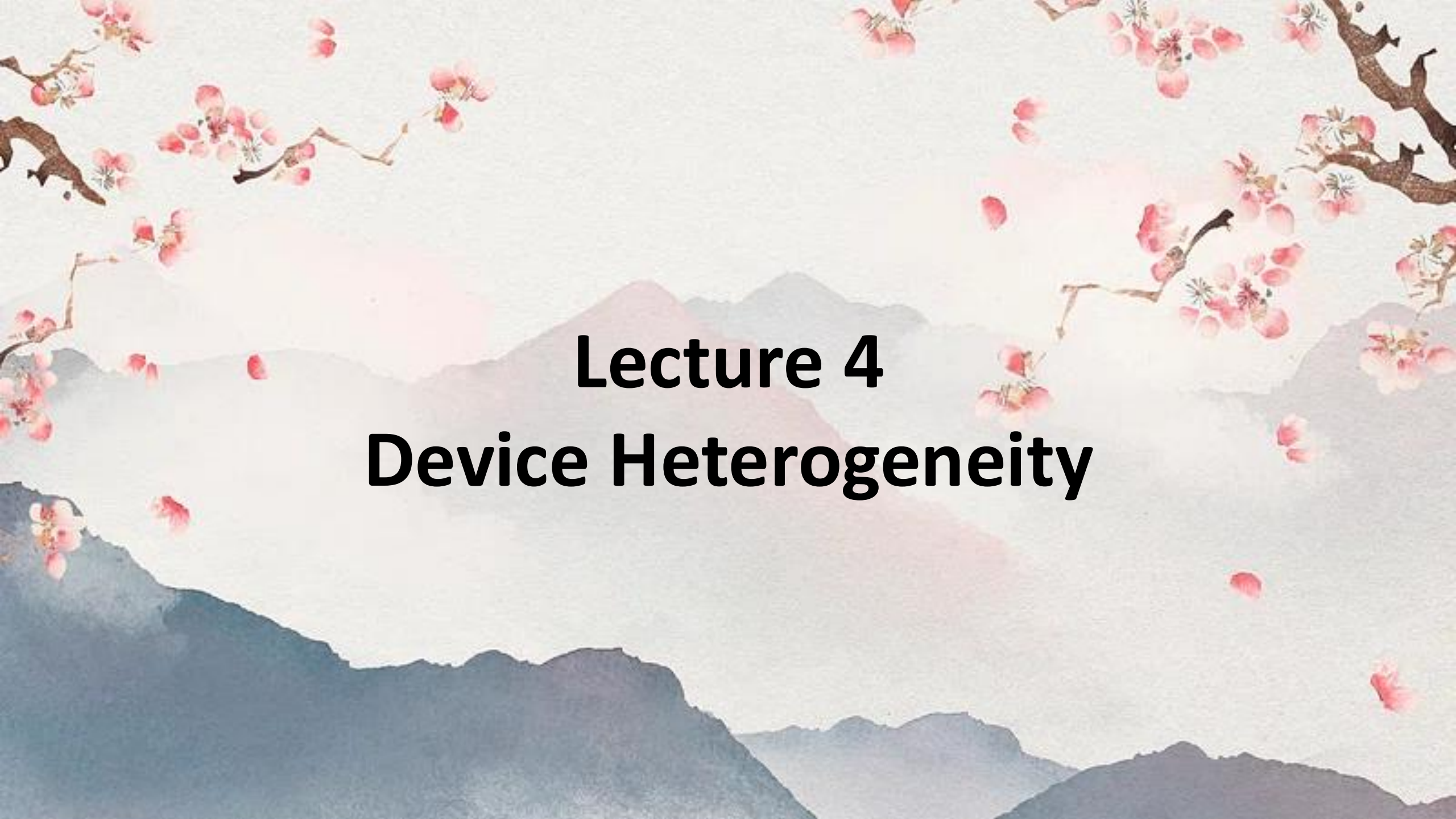
9/2025



Adrien Fradin, P.R., and Alexander Tyurin
Local SGD and federated averaging through the lens of time complexity
arXiv 2509.23207

Dual Local SGD
Decaying Local SGD
Decaying Local ASGD

What can recent
research on
asynchronous SGD
say about Local
SGD?

The background is a traditional Chinese ink wash painting. It features misty, layered mountains in shades of blue and grey. Delicate cherry blossom branches with pink flowers and falling petals are scattered across the scene, particularly in the upper corners. The overall style is soft and artistic.

Lecture 4

Device Heterogeneity



Introduction

Optimization Problem

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

parallel machines

model parameters / features

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 τ_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 θ_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 $\mathbb{E} [\|\nabla f(\hat{x})\|^2] \leq \varepsilon$

Typical Assumptions

- 1 $\inf f \in \mathbb{R}$
- 2 $f_i(x) := \mathbb{E}_{\xi \sim \mathcal{D}_i} [f_i(x, \xi)]$

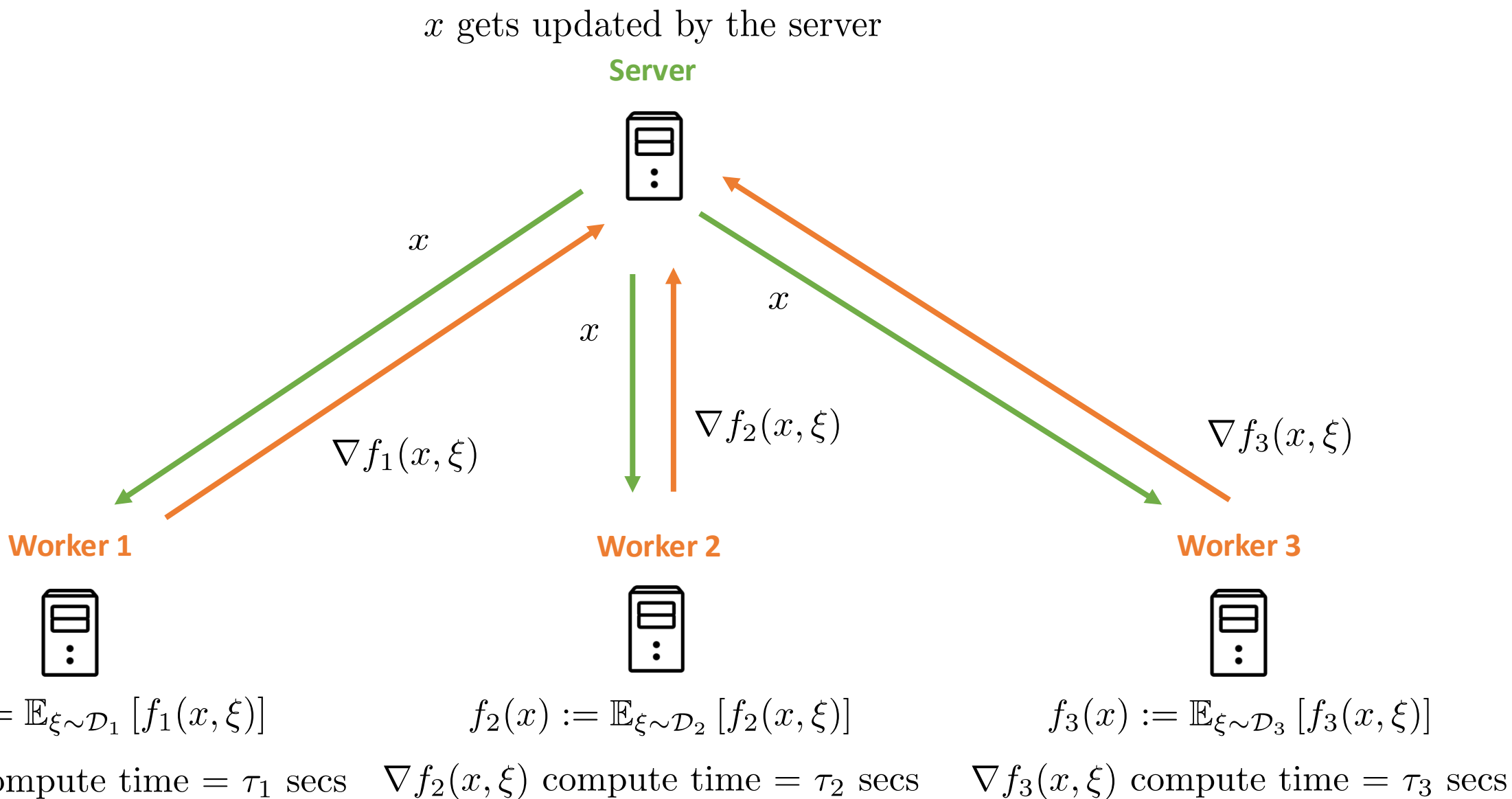
Gradient of local functions is Lipschitz:

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

Stochastic gradients have bounded variance:

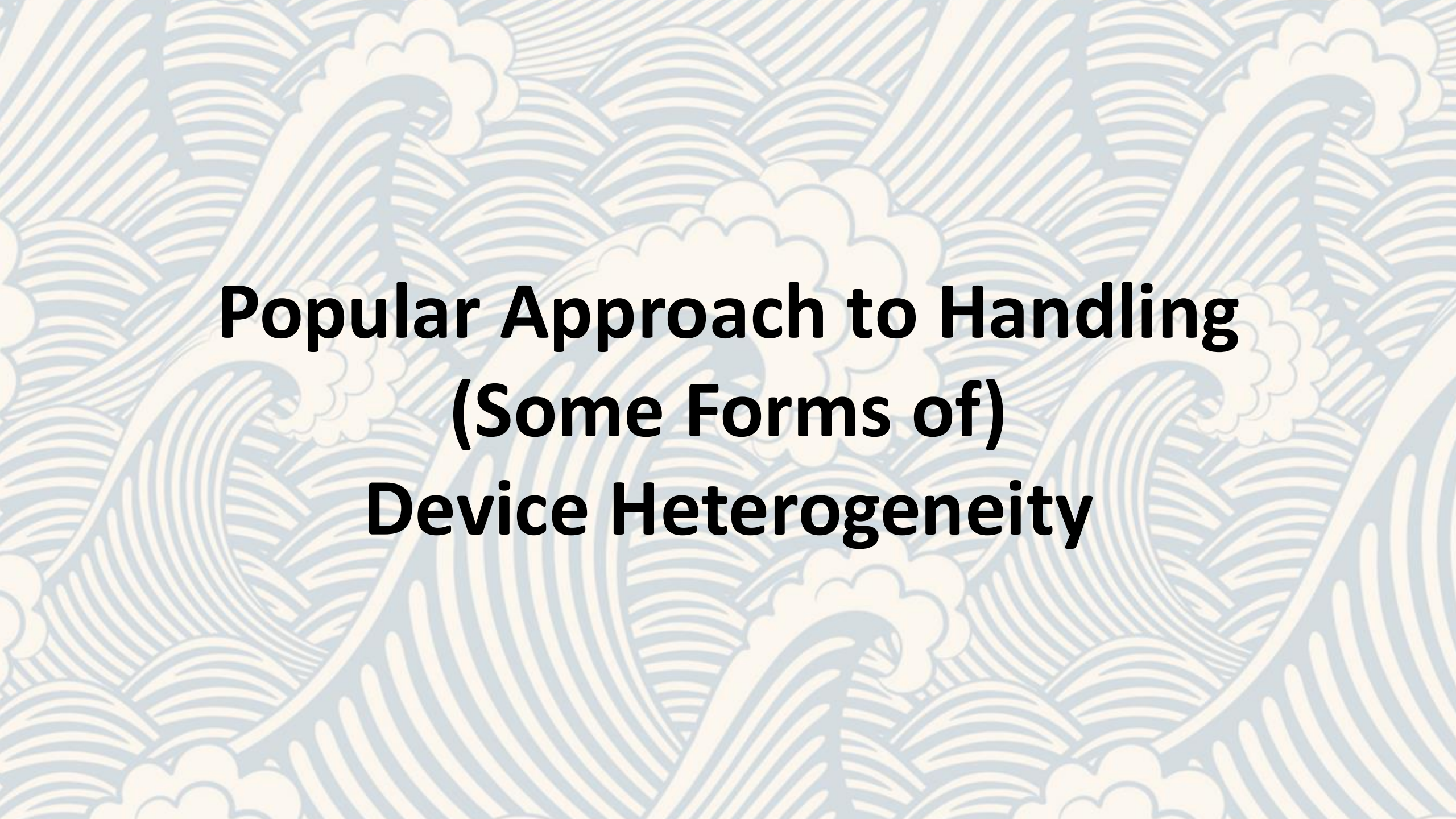
$$\max_{i \in \{1, \dots, n\}} \sup_{x \in \mathbb{R}^d} \mathbb{E}_{\xi \sim \mathcal{D}_i} [\|\nabla f_i(x, \xi) - \mathbb{E}_{\xi \sim \mathcal{D}_i} [\nabla f_i(x, \xi)]\|^2] \leq \sigma^2$$

Parallel Computing Architecture



Three Types of Heterogeneity

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



Popular Approach to Handling (Some Forms of) Device Heterogeneity

Async FL in Practice

6/2021

FedBuff



John Nguyen, Kshitiz Malik, Hongyuan Zhan, Ashkan Yousefpour,
Michael Rabbat, Mani Malek, Dzmitry Huba

Federated Learning with Buffered Asynchronous Aggregation

AISTATS 2022

11/2021

Papaya



D. Huba, J. Nguyen, K. Malik, R. Zhu, M. Rabbat, A. Yousefpour, C. -
J. Wu, H. Zhan, P. Ustinov, H. Srinivas, K. Wang, A. Shoumikhin, J.
Min, and M. Malek

Papaya: Practical, private, and scalable federated learning

Proceedings of Machine Learning and Systems 2022

Async FL systems in practice

Federated Learning with Buffered Asynchronous Aggregation

John Nguyen
Mike Rabbat

Kshitiz Malik
Mani Malek
Meta AI

Hongyuan Zhan

Ashkan Yousefpour
Dzmitry Huba

PAPAYA: PRACTICAL, PRIVATE, AND SCALABLE FEDERATED LEARNING

Dzmitry Huba¹ John Nguyen¹ Kshitiz Malik¹ Ruiyu Zhu¹ Mike Rabbat¹ Ashkan Yousefpour¹
Carole-Jean Wu¹ Hongyuan Zhan¹ Pavel Ustinov¹ Harish Srinivas¹ Kaikai Wang¹ Anthony Shoumikhin¹
Jesik Min¹ Mani Malek¹

ABSTRACT

Cross-device Federated Learning (FL) is a distributed learning paradigm with several challenges that differentiate it from traditional distributed learning, variability in the system characteristics on each device, and millions of clients coordinating with a central server being primary ones. Most FL systems described in the literature are synchronous – they perform a synchronized aggregation of model updates from individual clients. Scaling synchronous FL is challenging since increasing the number of clients training in parallel leads to diminishing returns in training speed, analogous to large-batch training. Moreover, stragglers hinder synchronous FL training. In this work, we outline a production asynchronous FL system design. Our work tackles the aforementioned issues, sketches of some of the system design challenges and their solutions, and touches upon principles that emerged from building a production FL system for millions of clients. Empirically, we demonstrate that asynchronous FL converges faster than synchronous FL when training across nearly one hundred million devices. In particular, in high concurrency settings, asynchronous FL is 5× faster and has nearly 8× less communication overhead than synchronous FL.

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

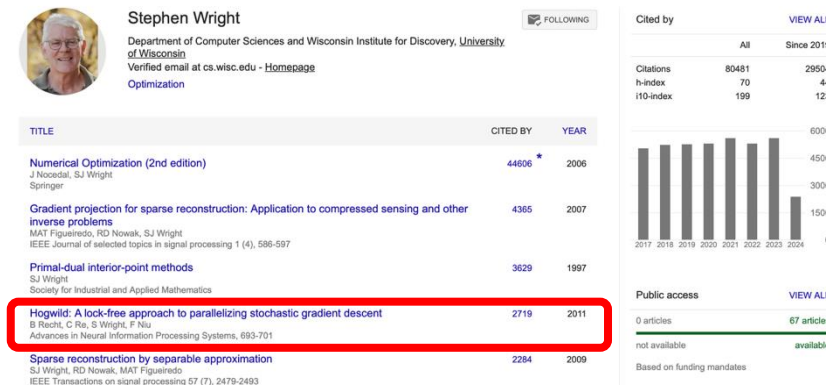
Stochastic Gradient Descent (SGD) is a popular algorithm that can achieve state-of-the-art performance on a variety of machine learning tasks. Several researchers have recently proposed schemes to parallelize SGD, but all require performance-destroying memory locking and synchronization. This work aims 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 (a thousand-dollar RAID

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

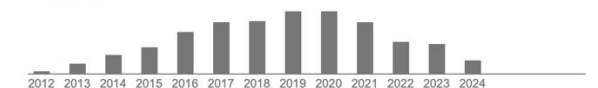
Publication date 2011

Conference Advances in Neural Information Processing Systems

Pages 693-701

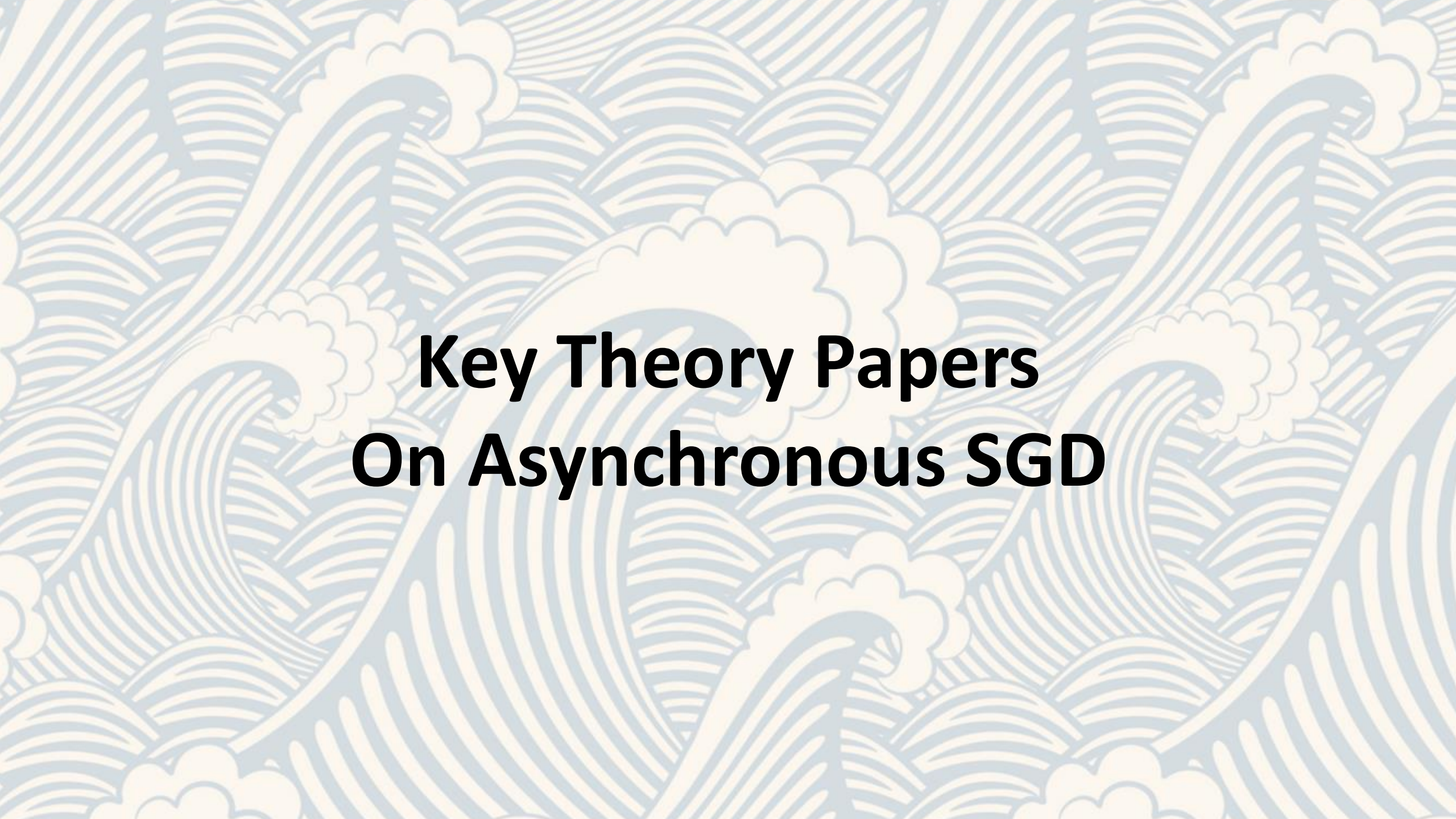
Description Stochastic Gradient Descent (SGD) is a popular algorithm that can achieve state-of-the-art performance on a variety of machine learning tasks. Several researchers have recently proposed schemes to parallelize SGD, but all require performance-destroying memory locking and synchronization. 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



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 *
RB NiuF - ... Systems, Granada, Spain, 2011
Cited by 2 Related articles



Key Theory Papers On Asynchronous SGD

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 ξ . 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 ∇F to update the parameters as $\mathbf{x}_k = \mathbf{x}_{k-1} - \gamma_k \nabla f(\mathbf{x}_{k-1}; \xi_{k-1})$ 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 a 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).

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 ε -stationary point is reached after $\mathcal{O}(\sigma^2 \varepsilon^{-2} + \tau_{\max} \varepsilon^{-1})$ iterations, where σ denotes the variance of stochastic gradients.

In this work we obtain (i) a tighter convergence rate of $\mathcal{O}(\sigma^2 \varepsilon^{-2} + \sqrt{\tau_{\max} \tau_{\text{avg}}} \varepsilon^{-1})$ *without any change in the algorithm*, where τ_{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 $\mathcal{O}(\sigma^2 \varepsilon^{-2} + \tau_{\text{avg}} \varepsilon^{-1})$, 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-

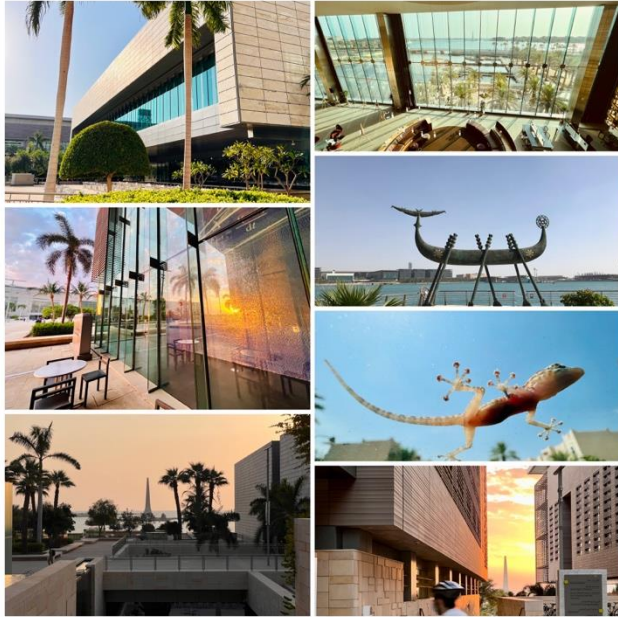
*CISPA Helmholtz Center for Information Security

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

arXiv: June 15, 2022

arXiv: June 16, 2022

Optimization & Machine Learning Lab @ KAUST



Artavazd Maranjyan

Ivan Ilin

Kaja Gruntkowska

Alexander Tyurin

Omar Shaikh Omar



Key Papers 1

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

First optimal parallel SGD under...

... 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
NeurIPS 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
NeurIPS 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
NeurIPS 2024

... computation and communication heterogeneity in the decentralized setup

Key Papers 2

8/2024

Rennala SGD
Malenia SGD



Alexander Tyurin

Tight Time Complexities in Parallel Stochastic Optimization with Arbitrary Computation Dynamics

ICLR 2025

... universal computation model

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

UAI 2025

*... random compute times
(when server can stop workers)*

1/2025

Ringmaster SGD



Artavazd Maranjyan, Alexander Tyurin and P.R.

Ringmaster ASGD: The first asynchronous SGD with optimal time complexity

ICML 2025

*1st optimal asynchronous SGD
(for homogeneous data)*

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

ICML 2025

*Adresses resource wastefulness
(for homogeneous data)*

Key Papers 3

5/2025

Birch SGD
(6 new methods)



Alexander Tyurin and Danil Sivtsov
Birch SGD: A tree graph framework for local and asynchronous SGD methods
arXiv 2505.09218

General framework

6 new methods!



9/2025

Ringleader
ASGD



Artavazd Maranjyan and P.R.
Ringleader ASGD: The first asynchronous SGD with optimal time complexity under data heterogeneity
ICLR 2026

***1st optimal asynchronous SGD
(for heterogeneous data)***

9/2025

Dual Local SGD
Decaying Local SGD
Hero SGD



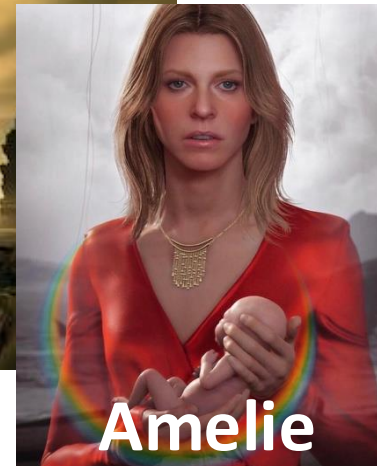
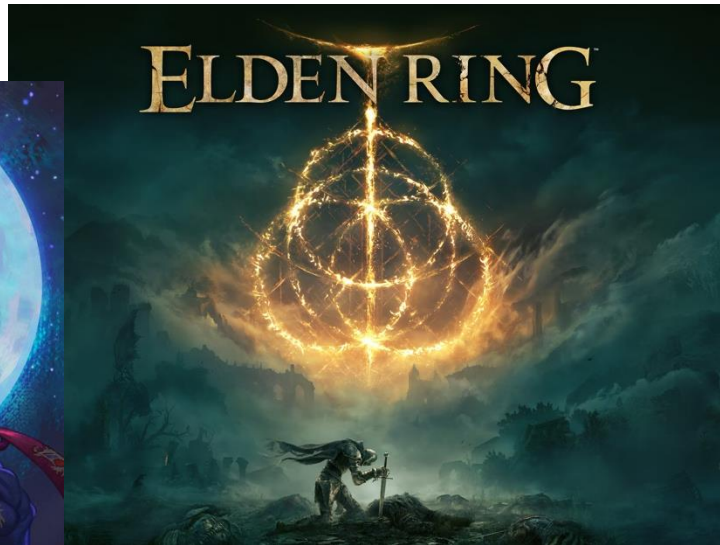
Adrien Fradin, P.R. and Alexander Tyurin
Local SGD and Federated Averaging Through the Lens of Time Complexity
arXiv 2509.23207

***1st proof that (new) variants of
Local SGD are optimal from the
viewpoint of time complexity
(for homogeneous data)***

Peter, What About the Weird Algorithm Names?



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.



Optimal Parallel Stochastic Gradient Methods

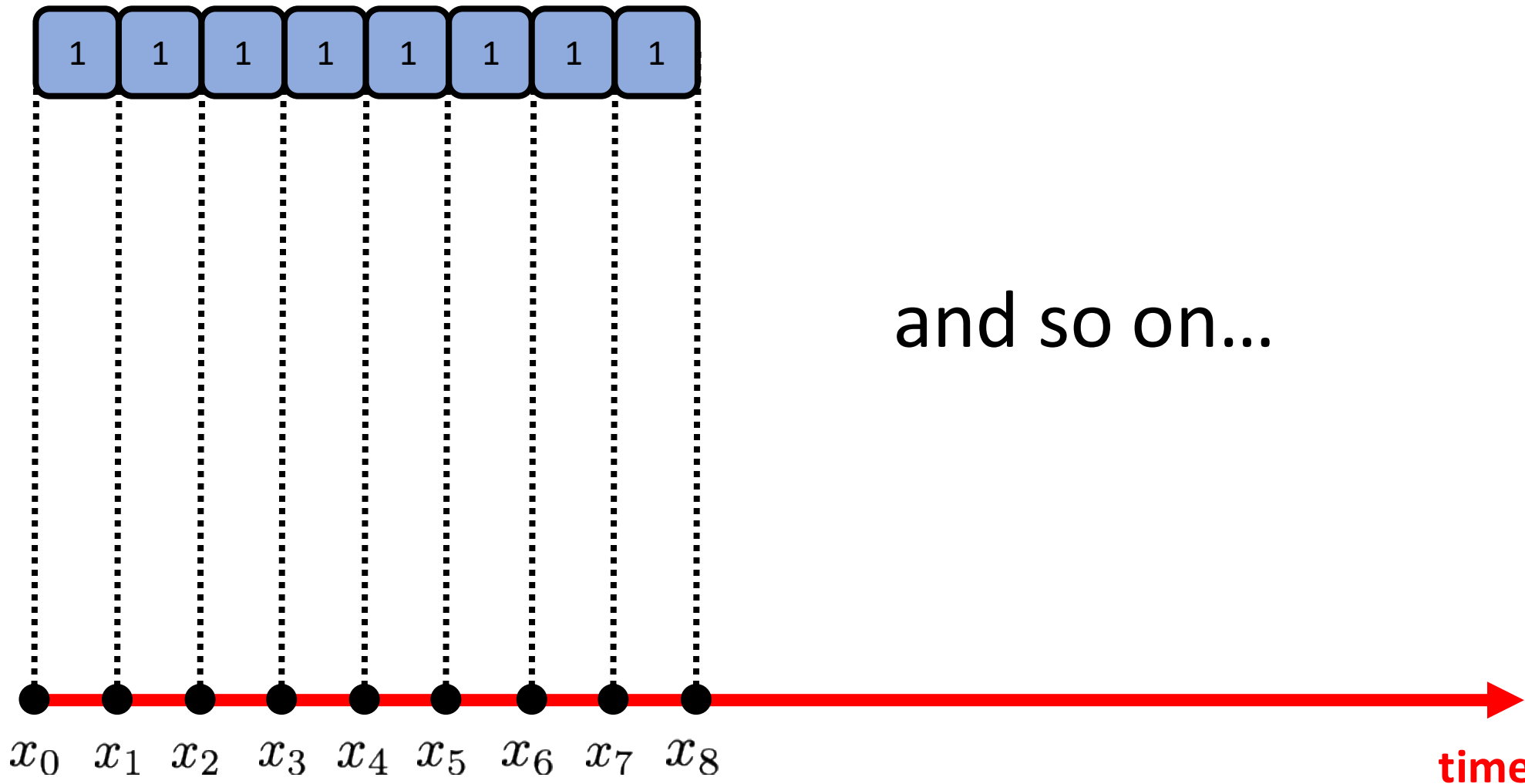
	Data Heterogeneity ($\mathcal{D}_i$ different)	Compute Heterogeneity (τ_i different)	Communication Heterogeneity (θ_i different)	Smooth Nonconvex	Smooth Convex	Infinite / Finite Sum?	Supports Decentralized Setup?	Optimal Time Complexity?
Rennala SGD Tyurin & R (NeurIPS '23)	✗	✓	0	✓		Inf	✗	✓
Malenia SGD 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	✓	✓



Previous Approaches to Parallelizing SGD

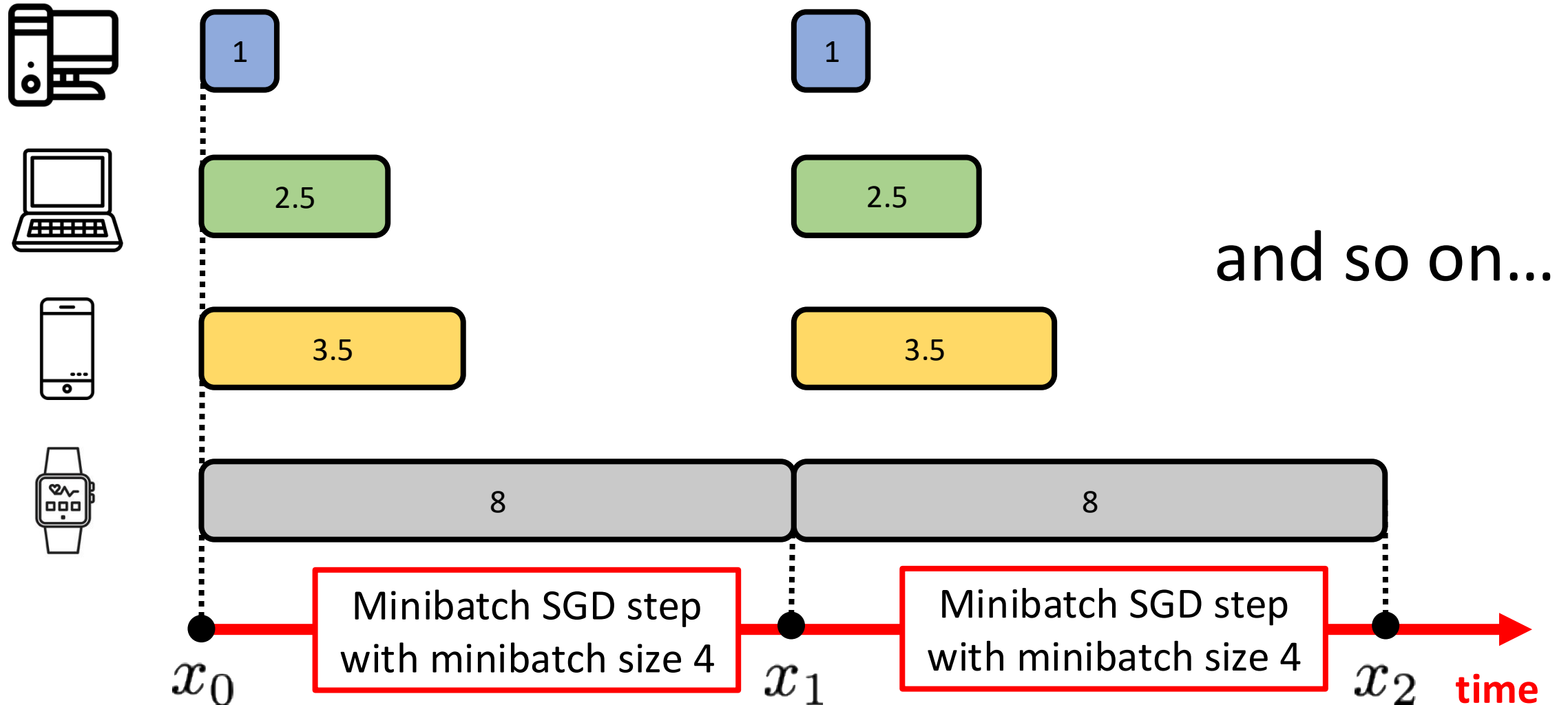
Hero SGD

Algorithmic idea: The fastest worker does it all



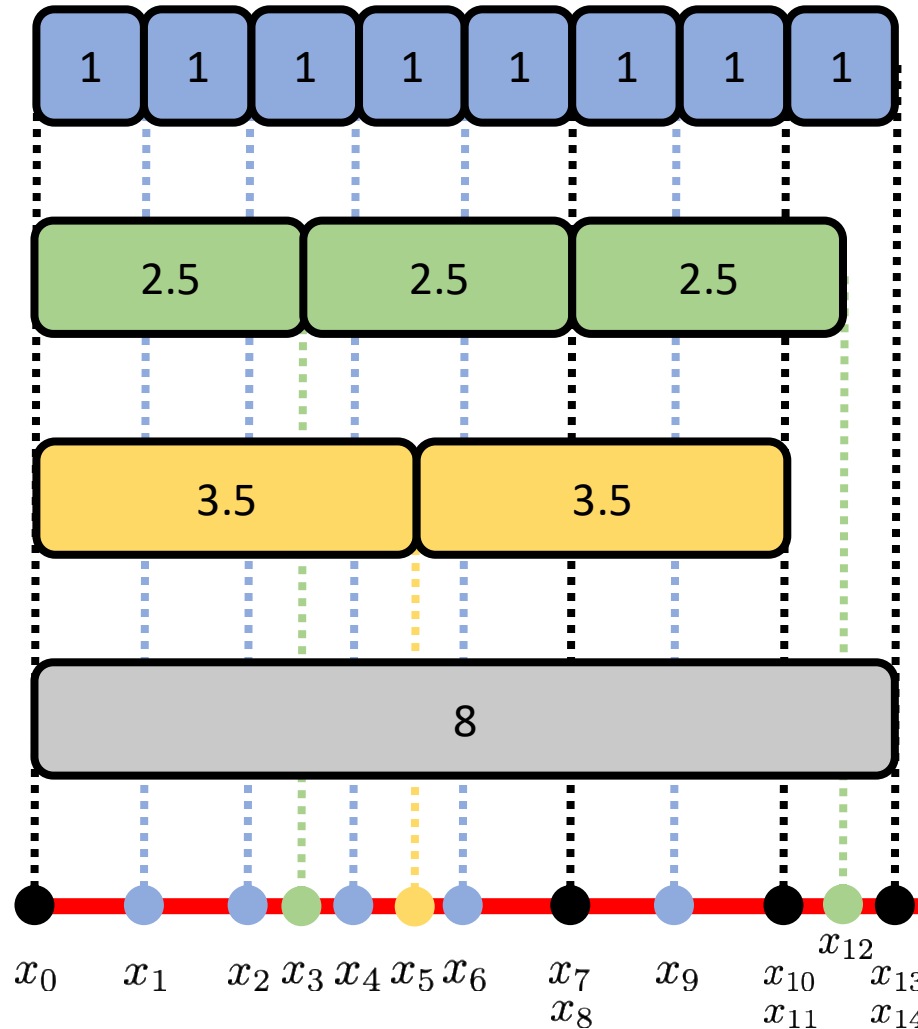
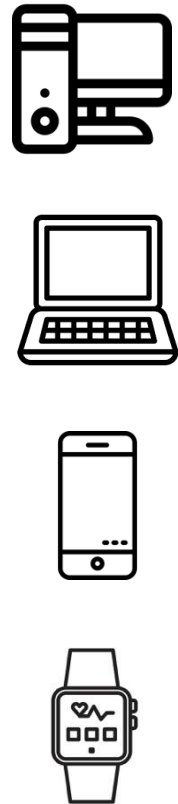
Naïve Minibatch SGD

Algorithmic idea: Each worker does one job only



Asynchronous SGD

Algorithmic idea: All workers are slaves and useful



and so on...

time

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 ξ . 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 ∇F to update the parameters as $\mathbf{x}_k = \mathbf{x}_{k-1} - \gamma_k \nabla f(\mathbf{x}_{k-1}; \xi_{k-1})$ 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 a 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).

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 ε -stationary point is reached after $\mathcal{O}(\sigma^2 \varepsilon^{-2} + \tau_{\max} \varepsilon^{-1})$ iterations, where σ denotes the variance of stochastic gradients.

In this work we obtain (i) a tighter convergence rate of $\mathcal{O}(\sigma^2 \varepsilon^{-2} + \sqrt{\tau_{\max} \tau_{\text{avg}}} \varepsilon^{-1})$ *without any change in the algorithm*, where τ_{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 $\mathcal{O}(\sigma^2 \varepsilon^{-2} + \tau_{\text{avg}} \varepsilon^{-1})$, 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 15, 2022

arXiv: June 16, 2022

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

Alexander Tyurin
KAUST
Saudi Arabia
alexandertyurin@gmail.com

Peter Richtárik
KAUST
Saudi Arabia
richtarik@gmail.com

Abstract

Parallelization is a popular strategy for improving the performance of iterative algorithms. Optimization methods are no exception: design of efficient parallel optimization methods and tight analysis of their theoretical properties are important research endeavors. While the minimum complexities are well known for sequential optimization methods, the theory of parallel optimization methods is less explored. In this paper, we propose a new protocol that generalizes the classical oracle framework approach. Using this protocol, we establish minimum complexities for parallel optimization methods that have access to an unbiased stochastic gradient oracle with bounded variance. We consider a fixed computation model characterized by each worker requiring a fixed but worker-dependent time to calculate stochastic gradient. 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

$$\min_{x \in Q} \left\{ f(x) := \mathbb{E}_{\xi} [L(x; \xi)] \right\}, \quad (1)$$

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

- (i) n workers are available to work in parallel,
- (ii) the i^{th} worker requires τ_i seconds¹ 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 σ^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 (∇f) calls (Lan, 2020; Nemirovski, 2018; Carmon 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

¹Or any other unit of time.

Rennala SGD



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

Setup

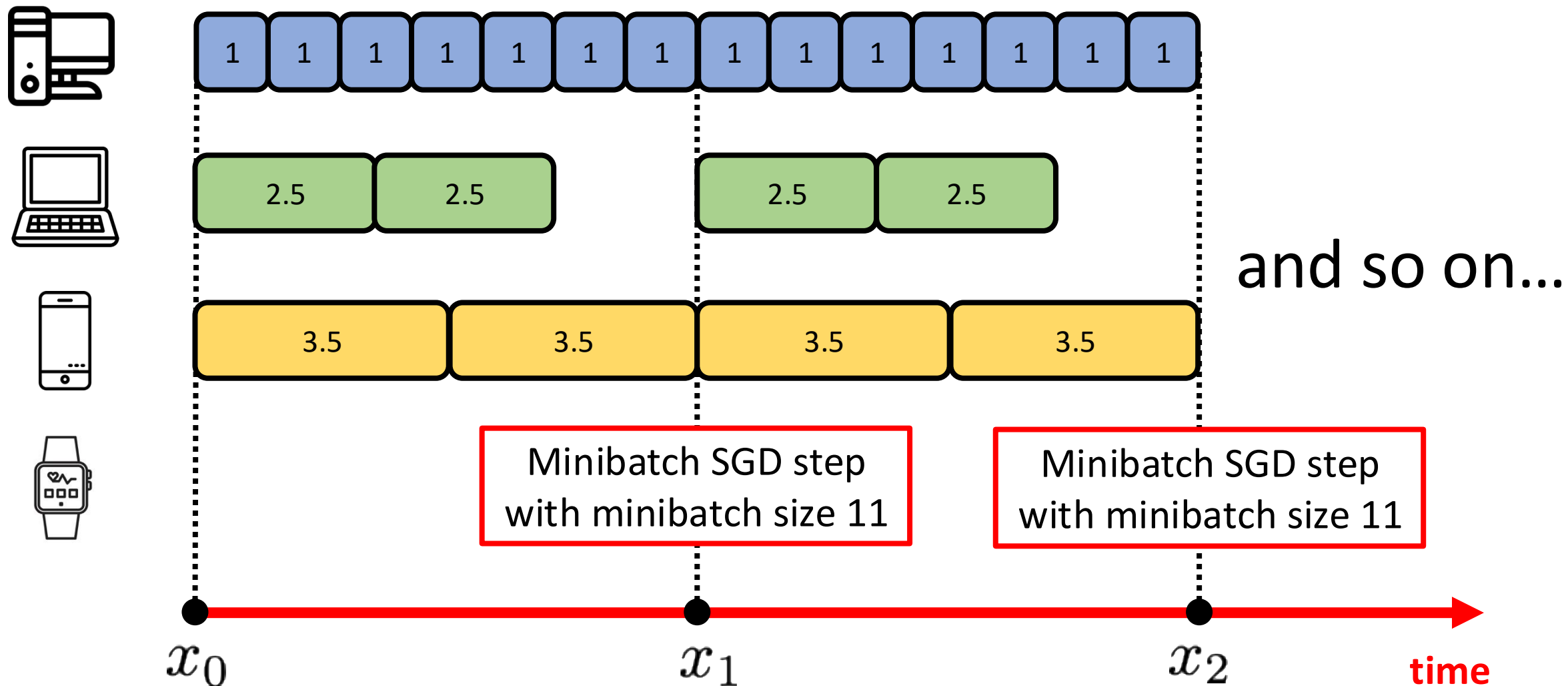
Optimal Parallel Stochastic Gradient Methods



	Data Heterogeneity ($\mathcal{D}_i$ different)	Compute Heterogeneity (τ_i different)	Communication Heterogeneity (θ_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	✓	✓

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

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

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

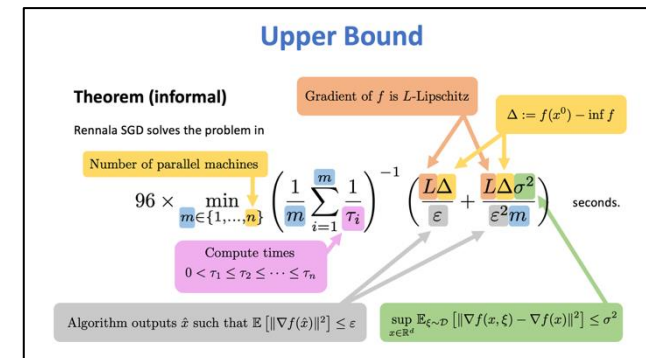
Matching Lower Bound

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

Function class

Distribution governing noise

Oracle class

Algorithm class

Protocol 1 Classical Oracle Protocol

- 1: **Input:** function $f \in \mathcal{F}$ oracle and
- 2: **for** $k = 0, \dots, \infty$ **do**
- 3: $x^k = A^k(a^1, \dots, a^k)$
- 4: $g^{k+1} = O(x^k, \xi^{k+1})$
- 5: **end for**

$\triangleright x^0 = A^0$ for $k = 0$.

Stochastic gradient:
 $g^{k+1} = \nabla f(x^k, \xi^{k+1})$

Iteration complexity (measure):

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

[Nemirovsky and Yudin, 1983] [Nesterov, 2018]

[Carmon et al, 2020] [Arjevani et al, 2022]

Natural for sequential methods, where a single worker does all the work!

New Oracle: Keeps Track of Time

Protocol 2 Time Oracle Protocol

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

Iteration complexity

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

Time complexity (complexity measure):

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

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

Natural for parallel methods!

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)

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

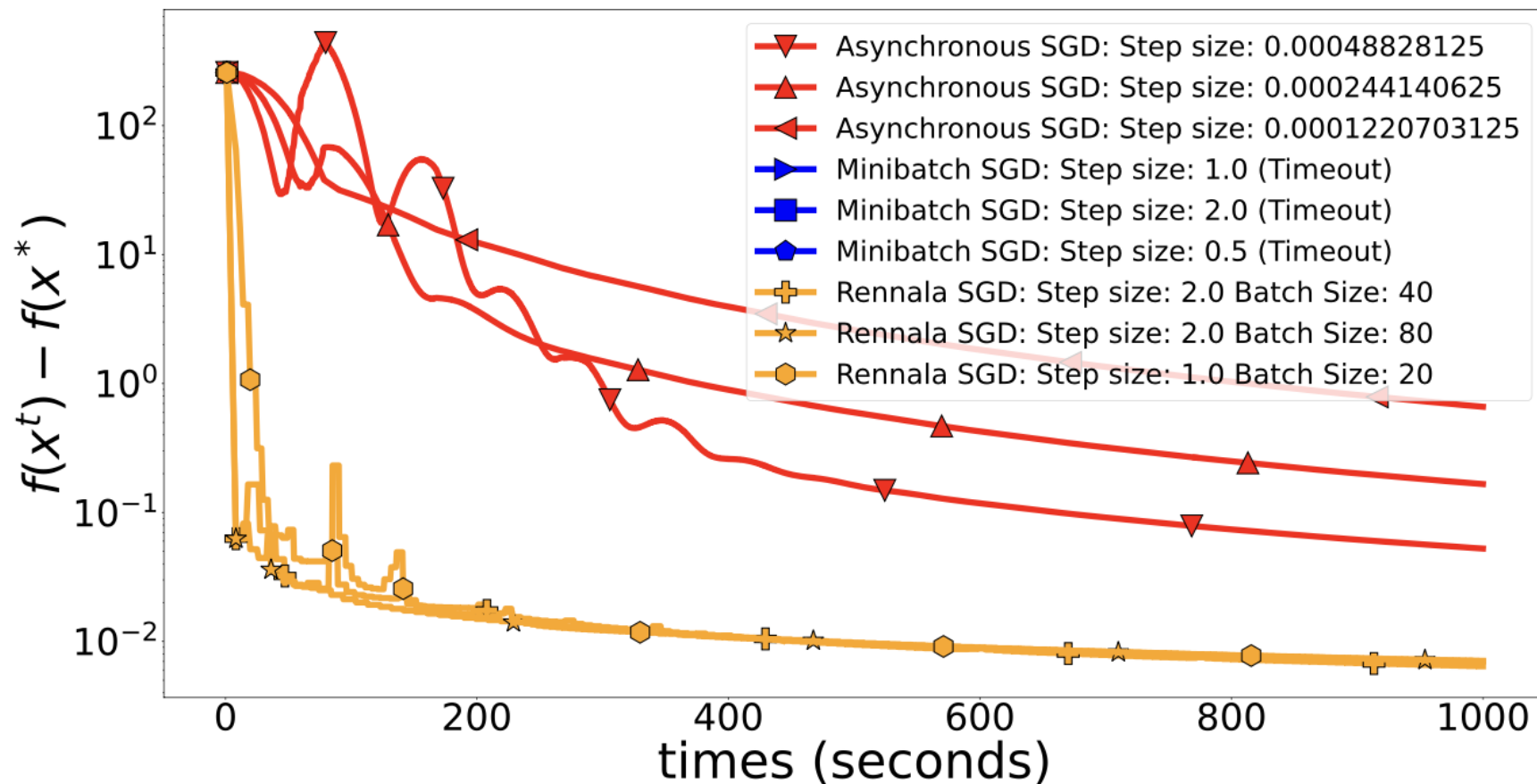


Figure 3: # of workers $n = 10000$.

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

Alexander Tyurin
KAUST
Saudi Arabia
alexandertyurin@gmail.com

Peter Richtárik
KAUST
Saudi Arabia
richtarik@gmail.com

Abstract

Parallelization is a popular strategy for improving the performance of iterative algorithms. Optimization methods are no exception: design of efficient parallel optimization methods and tight analysis of their theoretical properties are important research endeavors. While the minimum complexities are well known for sequential optimization methods, the theory of parallel optimization methods is less explored. In this paper, we propose a new protocol that generalizes the classical oracle framework approach. Using this protocol, we establish minimum complexities for parallel optimization methods that have access to an unbiased stochastic gradient oracle with bounded variance. We consider a fixed computation model characterized by each worker requiring a fixed but worker-dependent time to calculate stochastic gradient. 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

$$\min_{x \in Q} \left\{ f(x) := \mathbb{E}_{\xi} [L(x; \xi)] \right\}, \quad (1)$$

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

- (i) n workers are available to work in parallel,
- (ii) the i^{th} worker requires τ_i seconds¹ 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 σ^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 (∇f) calls (Lan, 2020; Nemirovski, 2010; Carmon 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

¹Or any other unit of time.

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 (τ_i different)	Communication Heterogeneity (θ_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:    $g = \widehat{\nabla} f_i(x^k; \xi)$ ,  $\xi \sim \mathcal{D}$ 
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 (τ_i different)	Communication Heterogeneity (θ_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	✓	✓

Convex (Data Homogeneous) Regime

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

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



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
NeurIPS 2024



Shadowheart SGD: Setup

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

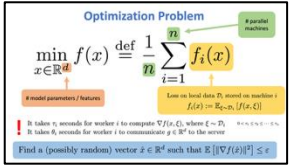
$$f_i(x) := \mathbb{E}_{\xi \sim \mathcal{D}_i} [f_i(x, \xi)]$$

Optimal Parallel Stochastic Gradient Methods

	Data Heterogeneity ($\mathcal{D}_i$ different)	Compute Heterogeneity (τ_i different)	Communication Heterogeneity (θ_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	✓	✓

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

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

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

$$x^{k+1} = x^k - \gamma \cdot$$

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

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

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

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

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

Shadowheart SGD

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 τ_i seconds to send *one coordinate/float* to server. Abbreviations: L = smoothness constant, ε = 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.

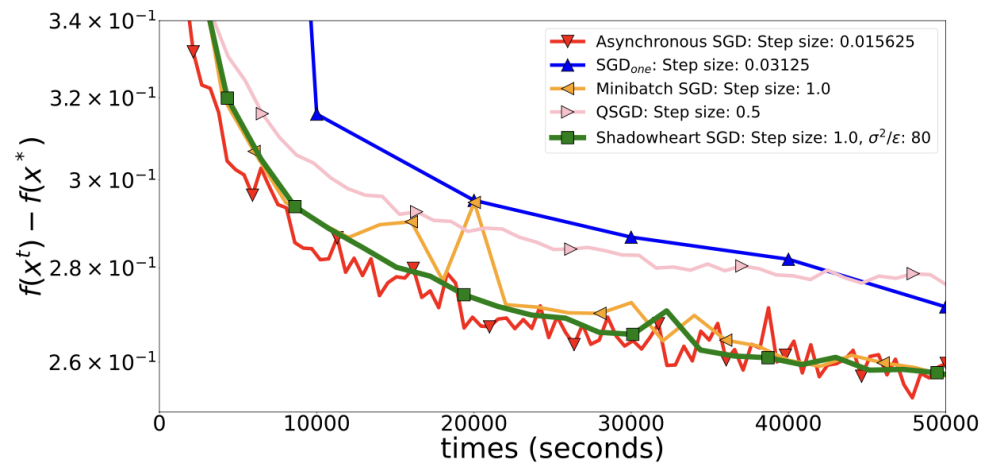
Method	Time Complexity	$\max\{h_n, \tau_n\} \rightarrow \infty$, $\max\{h_i, \tau_i\} < \infty \forall i < n$ (the last worker is slow)	Time Complexities in Some Regimes $h_i = h, \tau_i = \tau \forall i \in [n]$ (equal performance)	Numerical Comparison ^(b) $\sigma^2/\varepsilon =$ 1 10^3 10^6		
Minibatch SGD (see (3))	$\max_{i \in [n]} \max\{h_i, d\tau_i\} \left(\frac{L\Delta}{\varepsilon} + \frac{\sigma^2 L\Delta}{n\varepsilon^2} \right)$	∞ (non-robust)	$\max\{h, d\tau, \frac{d\tau\sigma^2}{n\varepsilon}, \frac{h\sigma^2}{n\varepsilon}\} \frac{L\Delta}{\varepsilon}$ (worse, e.g., when τ , d or n large)	$\times 10^3$	$\times 10^3$	$\times 10^4$
QSGD (see (7)) (Alistarh et al., 2017) (Khaled & Richtárik, 2020)	$\max_{i \in [n]} \max\{h_i, \tau_i\} \left(\left(\frac{d}{n} + 1 \right) \frac{L\Delta}{\varepsilon} + \frac{d\sigma^2 L\Delta}{n\varepsilon^2} \right)$	∞ (non-robust)	$\geq \frac{dh\sigma^2}{n\varepsilon} \frac{L\Delta}{\varepsilon}$ (worse, e.g., when ε small)	$\times 3$	$\times 10^2$	$\times 10^4$
Rennala SGD (Tyurin & Richtárik, 2023c), Asynchronous SGD (e.g., (Mishchenko et al., 2022))	$\geq \min_{j \in [n]} \max \left\{ h_{\bar{\pi}_j}, d\tau_{\bar{\pi}_j}, \frac{\sigma^2}{\varepsilon} \left(\sum_{i=1}^j \frac{1}{h_{\bar{\pi}_i}} \right)^{-1} \right\} \frac{L\Delta}{\varepsilon}$ ^(a)	$< \infty$ (robust)	$\geq \max \left\{ h, d\tau, \frac{h\sigma^2}{n\varepsilon} \right\} \frac{L\Delta}{\varepsilon}$ (worse, e.g., when τ , d or n large)	$\times 10^2$	$\times 10$	$\times 1.5$
Shadowheart SGD (see (9) and Alg. 1) (Corollary 4.4)	$t^*(d-1, \sigma^2/\varepsilon, [h_i, \tau_i]_1^n) \frac{L\Delta}{\varepsilon}$ ^(c)	$< \infty$ (robust)	$\max \left\{ h, \tau, \frac{d\tau}{n}, \sqrt{\frac{d\tau h\sigma^2}{n\varepsilon}}, \frac{h\sigma^2}{n\varepsilon} \right\} \frac{L\Delta}{\varepsilon}$	$\times 1$	$\times 1$	$\times 1$

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).

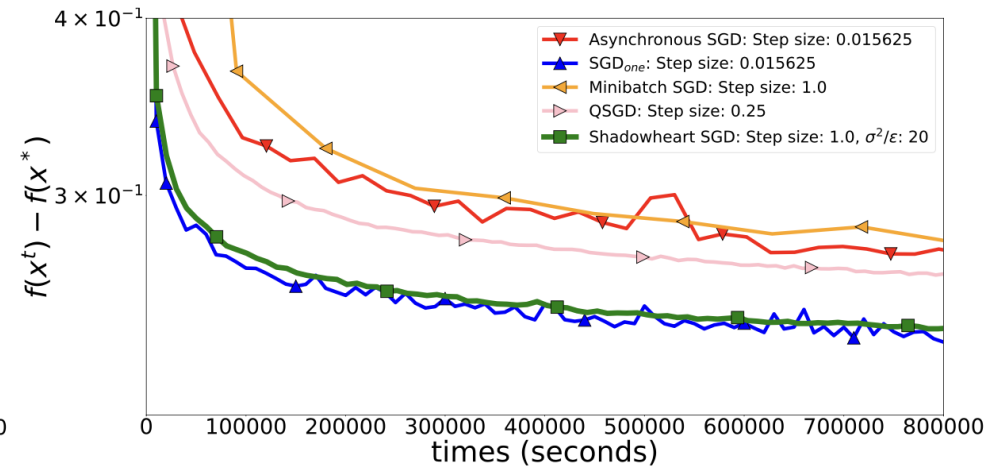
^(a) Upper bound time complexities are not derived for Rennala SGD and Asynchronous SGD. However, we can derive the lower bound using Theorem N.5 with $\omega = 0$. One should take $d\tau_i$ instead of τ_i when apply Theorem N.5 because these methods send d coordinates. $\bar{\pi}$ is a permutation that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

^(b) We numerically compute time complexities for $d = 10^6$, $n = 10^3$, $h_i \sim U(0.1, 1)$, $\tau_i \sim U(0.1, 1)$ (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.

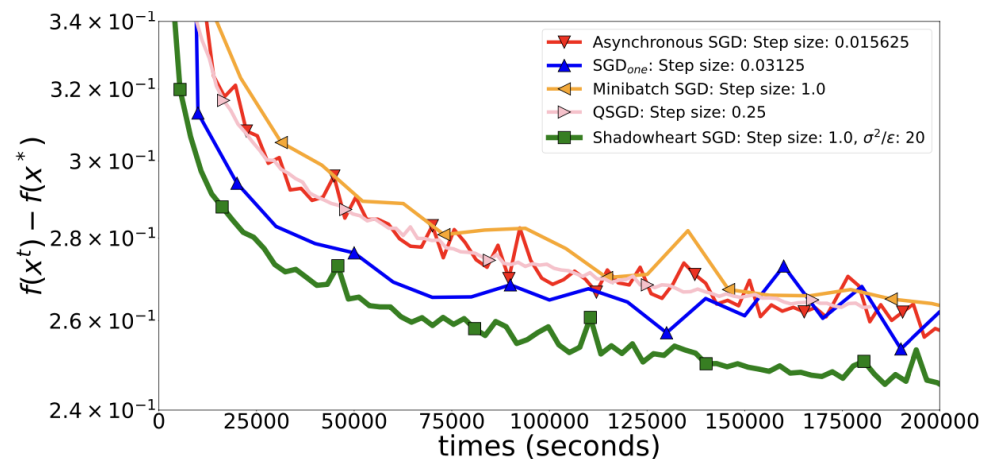
^(c) The mapping t^* is defined in Def. 4.2.



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



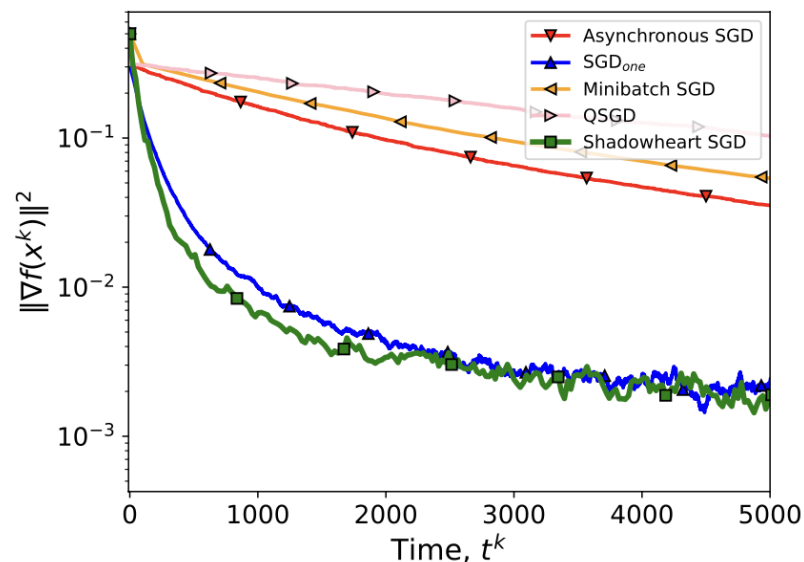
Slow communication: $\dot{\theta}_i = \frac{\sqrt{i}}{d^{1/2}}$



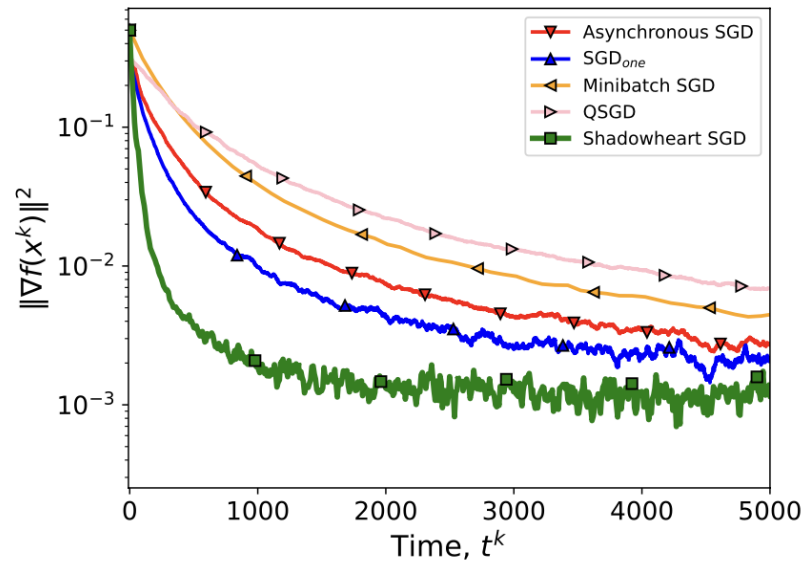
Medium-speed communication: $\dot{\theta}_i = \frac{\sqrt{i}}{d^{3/4}}$

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

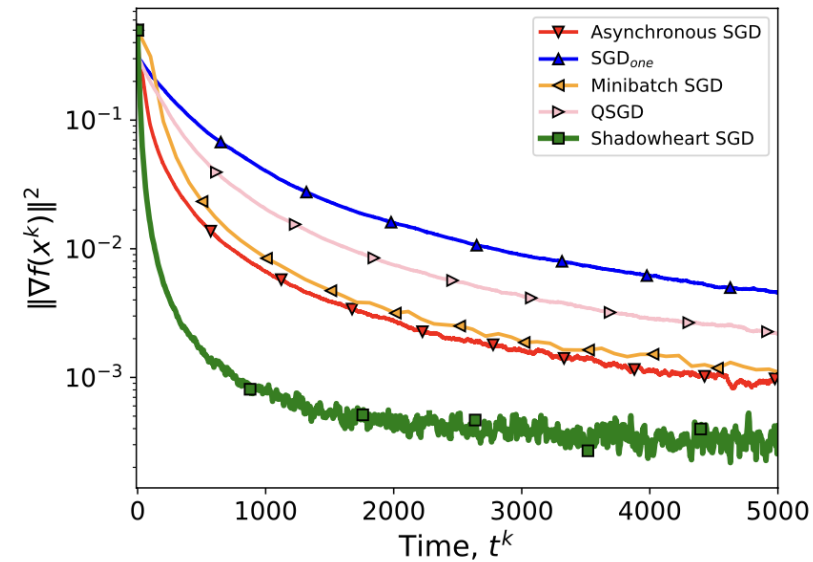
Shadowheart SGD: Adding More Workers...



(a) $n = 10$



(b) $n = 10^2$



(c) $n = 10^3$

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

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

Alexander Tyurin Kaja Gruntkowska Peter Richtárik

Abstract

In practical distributed systems, workers are typically not homogeneous, and due to differences in hardware configurations and network conditions, can have highly varying processing times. We consider smooth nonconvex finite-sum empirical risk minimization problems in this setting and consider a new stochastic optimization, Freya PAGE, designed to handle arbitrarily heterogeneous and asynchronous computations. By being robust to “stragglers” and applying sparsity-time computations, Freya PAGE offers significantly improved time complexity guarantees compared to all previous methods, including Accelerating SGD, Resona SGD, SPQR, and PHOTON, while requiring modest assumptions. We establish new worst-case and stochastic gradient reflection strategies with theoretical guarantees that can be of interest on their own, and may be used in the design of future optimization methods. Furthermore, we establish a lower bound for smooth nonconvex finite-sum problems in the asynchronous setting, providing a fundamental time complexity barrier. This lower bound is tight and demonstrates the optimality of Freya PAGE in the large-scale regime, i.e., when $n \gg \frac{1}{\epsilon}$, where n is # of workers, and ϵ is # of data samples.

1 Introduction

In real-world distributed systems used for large-scale machine learning tasks, it is common to encounter device heterogeneity and variations in processing times among different computational units. These can arise from CPU, network, or other factors in hardware configurations, network conditions, and other factors, resulting in different computational capabilities and delays across workers. (Chen et al., 2018; Tyurin and Richtárik, 2021). As a result, some slower than others computations take, while others experience delays or even fail to participate in the training altogether. Due to the above reasons, we aim to address the challenges posed by device heterogeneity in the context of solving finite-sum nonconvex optimization problems of the form

$$\min_{\theta} \left\{ F(\theta) = \frac{1}{n} \sum_{i=1}^n f_i(\theta) \right\}, \quad (1)$$

where $f_i: \mathbb{R}^d \rightarrow \mathbb{R}$ can be viewed as the loss of a machine learning model θ on the i^{th} example in a training dataset with n samples. Our goal is to find an ϵ -stationary point, i.e., a (possibly random) point θ such that $\mathbb{E} \|\nabla F(\theta)\|^2 \leq \epsilon$. We focus on the homogeneous distributed setting

- there are n workers/clients that are able to work in parallel;
- each worker has access to stochastic gradients $\nabla f_i(\theta)$, $i \in [n]$, $\theta \in \mathbb{R}^d$;
- worker i calculates $\nabla f_i(\theta)$ in time or speed $\tau_i \in [0, \infty)$, rounds the all $\theta \in \mathbb{R}^d$, $i \in [n]$.

Proposed. 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

NeurIPS 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 (τ_i different)	Communication Heterogeneity (θ_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	✓	✓



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

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

PAGE: Optimal Serial SGD for Finite-Sum Nonconvex Optimization

PAGE: A Simple and Optimal Probabilistic Gradient Estimator for Nonconvex Optimization

Zhize Li¹, Hongyan Bao¹, Xiangliang Zhang¹, Peter Richtárik¹

Abstract

In this paper, we propose a novel stochastic gradient estimator—Probabilistic Gradient Estimator (PAGE)—for nonconvex optimization. PAGE is easy to implement as it is designed via a small adjustment to vanilla SGD: in each iteration, PAGE uses the vanilla minibatch SGD update with probability p_t or reuses the previous gradient with a small adjustment, at a much lower computational cost, with probability $1 - p_t$. We give a simple formula for the optimal choice of p_t . Moreover, we prove the first tight lower bound $\Omega(n + \frac{1}{p^2})$ for nonconvex finite-sum problems, which also leads to a tight lower bound $\Omega(b + \frac{1}{p^2})$ for nonconvex online problems, where $b := \min(\frac{1}{p^2}, n)$. Then, we show that PAGE obtains the optimal convergence results $O(n + \frac{1}{p^2})$ (finite-sum) and $O(b + \frac{1}{p^2})$ (online) matching our lower bounds for both nonconvex finite-sum and online problems. Besides, we also show that for nonconvex functions satisfying the Polyak-Lejasiewicz (PL) condition, PAGE can automatically switch to a faster linear convergence rate $O(\log \frac{1}{\epsilon})$. Finally, we conduct several deep learning experiments (e.g., LeNet, VGG, ResNet) on real datasets in PyTorch showing that PAGE not only converges much faster than SGD in training but also achieves the higher test accuracy, validating the optimal theoretical results and confirming the practical superiority of PAGE.

1. Introduction

Nonconvex optimization is ubiquitous across many domains of machine learning, including robust regression, low rank matrix recovery, sparse recovery and supervised learning

¹King Abdullah University of Science and Technology, Thuwal, Kingdom of Saudi Arabia. Correspondence to: Zhize Li <zhize.li@kaust.edu.sa>.

Proceedings of the 38th International Conference on Machine Learning, PMLR 139, 2021. Copyright 2021 by the author(s).

(Jain & Kar, 2017). Driven by the applied success of deep neural networks (LeCun et al., 2015), and the critical place nonconvex optimization plays in training them, research in nonconvex optimization has been undergoing a renaissance (Ghadimi & Lan, 2013; Ghadimi et al., 2016; Zhou et al., 2018; Fang et al., 2018; Li, 2019; Li & Richtárik, 2020).

1.1. The problem

Motivated by this development, we consider the general optimization problem

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

where $f: \mathbb{R}^d \rightarrow \mathbb{R}$ is a differentiable and possibly nonconvex function. We are interested in functions having the finite-sum form

$$f(x) := \frac{1}{n} \sum_{i=1}^n f_i(x), \quad (2)$$

where the functions f_i are also differentiable and possibly nonconvex. Form (2) captures the standard empirical risk minimization problems in machine learning (Shalev-Shwartz & Ben-David, 2014). Moreover, if the number of data samples n is very large or even infinite, e.g., in the online/streaming case, then $f(x)$ usually is modeled via the online form

$$f(x) := \mathbb{E}_{\zeta \sim \mathcal{D}} [F(x, \zeta)], \quad (3)$$

which we also consider in this work. For notational convenience, we adopt the notation of the finite-sum form (2) in the descriptions and algorithms in the rest of this paper. However, our results apply to the online form (3) as well by letting $f_i(x) := F(x, \zeta_i)$ and treating n as a very large value or even infinite.

1.2. Gradient complexity

To measure the efficiency of algorithms for solving the nonconvex optimization problem (1), it is standard to bound the number of stochastic gradient computations needed to find a solution of suitable characteristics. In this paper we

$$\min_{x \in \mathbb{R}^d} \left\{ f(x) = \frac{1}{n} \sum_{i=1}^n f_i(x) \right\}$$
$$f_i(x) := \mathbb{E}_{\xi \sim \mathcal{D}_i} [f_i(x, \xi)]$$

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

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



Zhize Li, Hongyan Bao, Xiangliang Zhang, and P.R.
**PAGE: A simple and optimal probabilistic
gradient estimator for nonconvex optimization**
ICML 2021

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 ∇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, $\varepsilon =$ 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 ^(a) , suboptimal. ^(b)
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: σ^2 -bounded variance assumption, suboptimal when ε 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: σ^2 -bounded variance assumption, suboptimal when ε is small.
Freya PAGE (Theorems 7 and 8)	$\min_{j \in [n]} \left(\left(\sum_{i=1}^j \frac{1}{\tau_i} \right)^{-1} (m + j) \right) + \frac{\delta^0}{\varepsilon} \min_{j \in [n]} \left(\left(\sum_{i=1}^j \frac{1}{\tau_i} \right)^{-1} (\sqrt{m} + j) \right)^{(c)}$	Optimal in the large-scale regime, i.e., $\sqrt{m} \geq n$ (see Section 5)
Lower bound (Theorem 10)	$\min_{j \in [n]} \left(\left(\sum_{i=1}^j \frac{1}{\tau_i} \right)^{-1} (m + j) \right) + \frac{\delta^0}{\sqrt{m\varepsilon}} \min_{j \in [n]} \left(\left(\sum_{i=1}^j \frac{1}{\tau_i} \right)^{-1} (m + j) \right)$	—

Freya PAGE has *universally* better guarantees than all previous methods: the dependence on ε 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 $\sqrt{m} \geq n$.

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

^(b) Their convergence rates in Theorems 1 and 3 depend on a bound on the delays Δ , 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.

^(c) 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)

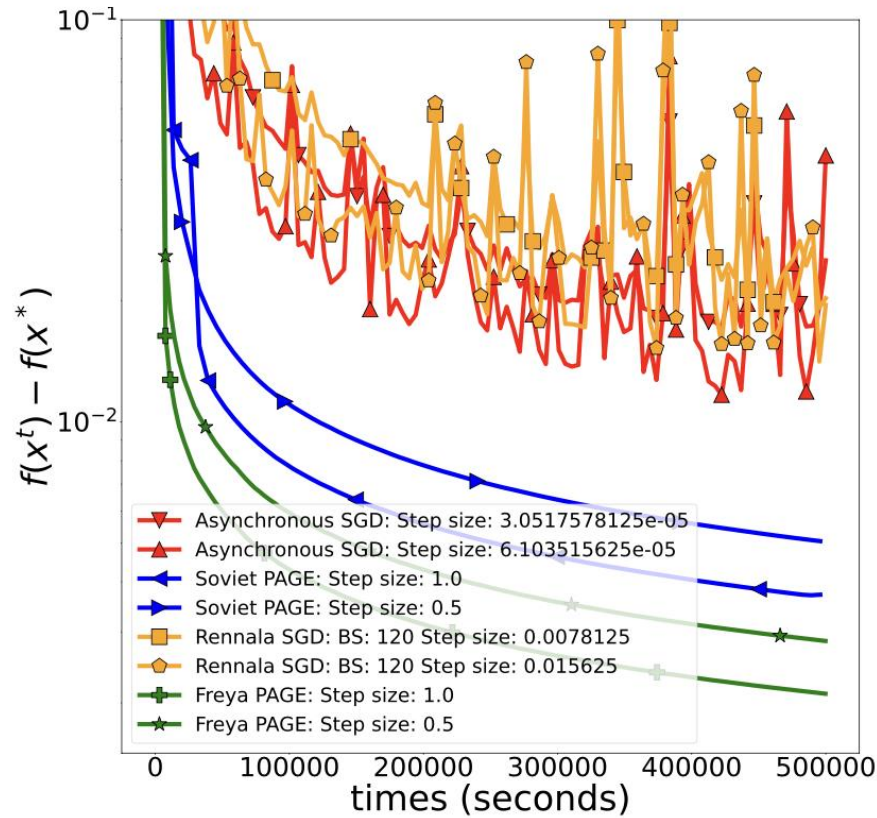
```
1: Input: point  $x \in \mathbb{R}^d$ 
2: Init  $g = 0 \in \mathbb{R}^d$ , set  $\mathcal{M} = \emptyset$ 
3: Broadcast  $x$  to all workers
4: For each worker  $i \in [n]$ , sample  $j$  from  $[m]$ 
   uniformly and ask it to calculate  $\nabla f_j(x)$ 
5: while  $\mathcal{M} \neq [m]$  do
6:   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)

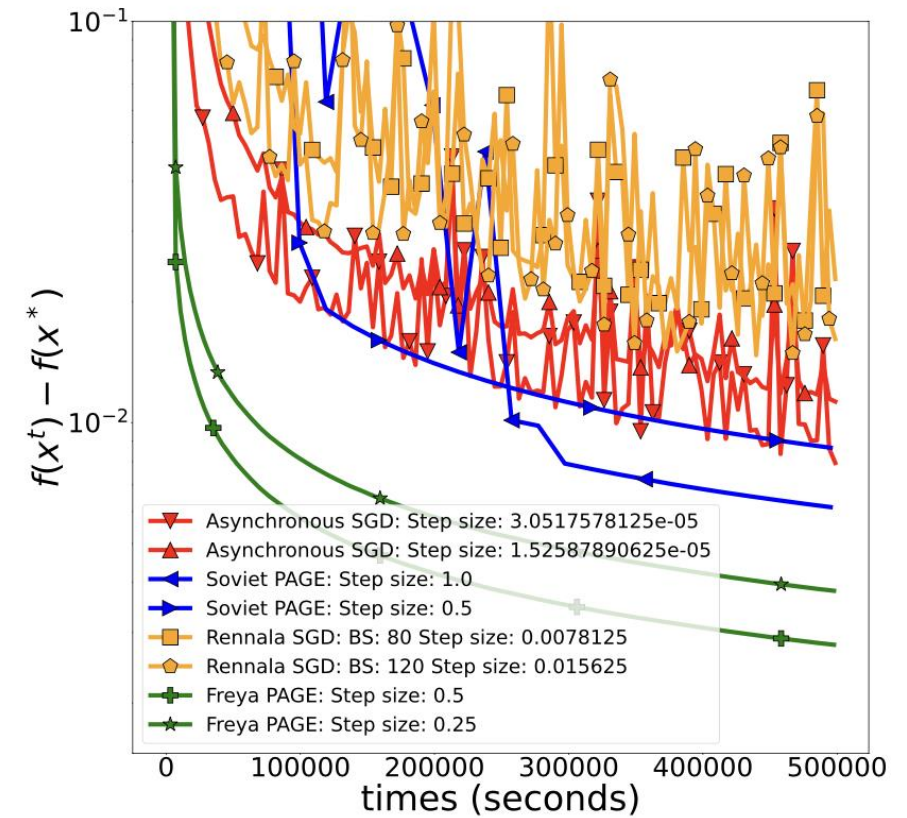
```
1: Input: batch size  $S \in \mathbb{N}$ , points  $x, y \in \mathbb{R}^d$ 
2: Init  $g = 0 \in \mathbb{R}^d$ 
3: Broadcast  $x, y$  to all workers
4: For each worker, sample  $j$  from  $[m]$  uniformly
   and ask it to calculate  $\nabla f_j(x) - \nabla f_j(y)$ 
5: for  $i = 1, 2, \dots, S$  do
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 ∇f_p locally, and the algorithm can call AllReduce once to collect all calculated gradients. ii) By splitting $[m]$ into blocks, instead of one ∇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



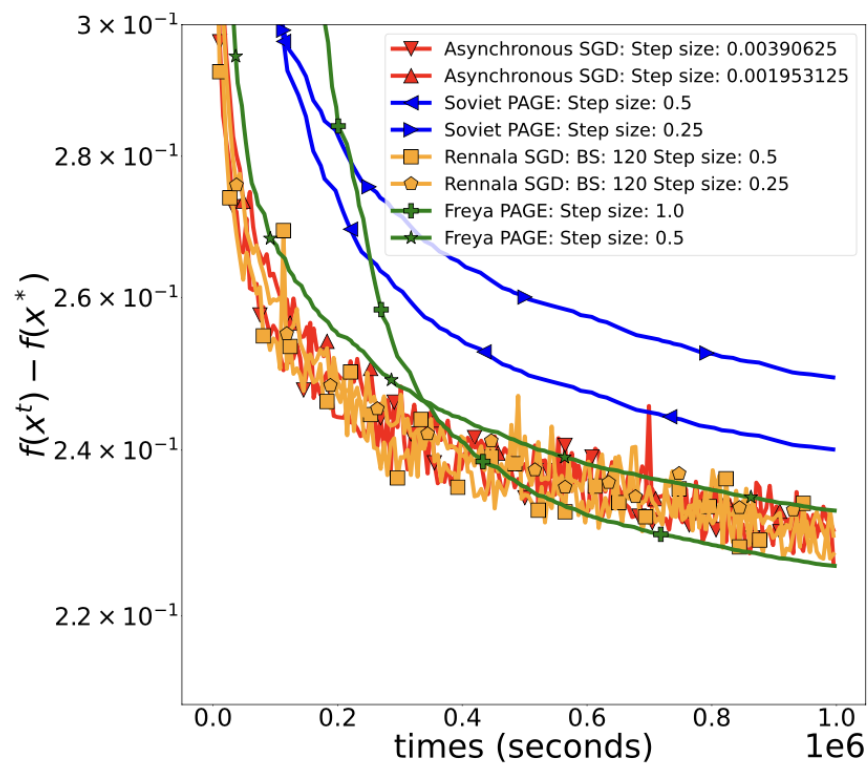
(a) $n = 1000$



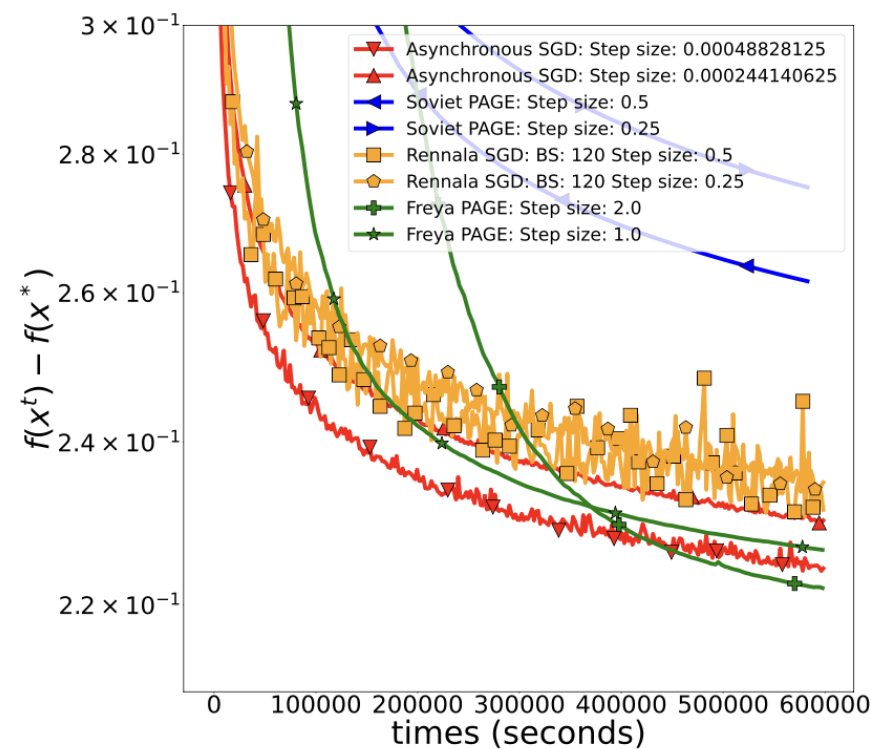
(b) $n = 10000$

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
Freya PAGE	92.66	1.01e-07

On the Optimal Time Complexities in Decentralized Stochastic Asynchronous Optimization

Alexander Tyurin Peter Rakhovskiy
King Abdullah University of Science and Technology (KAUST)
Saudi Arabia South Korea
alexander.tyurin@kaust.edu.sa prakhovskiy@kaust.edu.sa

Abstract

We consider the decentralized stochastic asynchronous optimization setup, where many workers asynchronously calculate stochastic gradients and asynchronously communicate with each other using edges in a network. For both homogeneous and heterogeneous setups, we prove near time complexity lower bounds under the assumption that computation and communication speeds are bounded. We propose a new nearly optimal method, Fragile SGD, and a new optimal method, Amie SGD. Our results extend existing homogeneous computation and communication speed and delay-free lower bounds to heterogeneous setups in the homogeneous setting. Our time complexities are new, nearly optimal, and possibly represent all previous asynchronous-stochastic methods in the decentralized setup.

1 Introduction

We consider the smooth recurrent optimization problem

$$\min_{x \in \mathbb{R}^n} \{f(x) = \mathbb{E}_{x \sim P_t} [f(x|t)]\}, \quad (1)$$

where $f: \mathbb{R}^n \rightarrow \mathbb{R}$, and P_t is a distribution on an empty set $\mathbb{R}^n$. For a given $t > 0$, we want to find a possibly random point $\hat{x}$, called an ϵ -stationary point, such that $\mathbb{E}[\|\nabla f(\hat{x})\|^2] \leq \epsilon$. We analyze the heterogeneous setup and the current setup with smooth and non-smooth functions in Sec. 1.1.

1.1 Decentralized setup with time

We investigate the following decentralized asynchronous setup. Assume that we have a network with the associated computer time $\{t_i\}$ and communication time $\{c_{ij}\}$. It takes time t_i to compute a stochastic gradient at node i , and c_{ij} to send it to node j . We assume that $t_i \in [0, \infty)$ and $c_{ij} \in [0, \infty)$. All computations and communications are done asynchronously and in parallel. We assume that the network is connected, i.e., there is a path between any two nodes i and j . We assume that the network is connected, i.e., there is a path between any two nodes i and j . We assume that the network is connected, i.e., there is a path between any two nodes i and j .

We consider now weighted directed undigraph represented by a matrix $A \in \mathbb{R}^{n \times n}$ such that $A_{ij} \in [0, 1]$, and matrix of distances $L \in \mathbb{R}^{n \times n}$ such that $L_{ij} \in [0, \infty)$ for all $i, j \in [n]$ and $L_{ii} = 0$ for all $i \in [n]$. Every vertex i is connected to one other vertex j with no edges $i \rightarrow j$ and $j \rightarrow i$. For this setup, it would be convenient to define the distance of the shortest path from

Amelie SGD

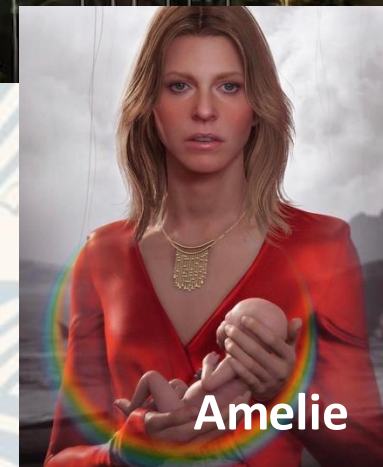
Optimal Decentralized SGD under Computation & Communication Heterogeneity



Alexander Tyurin and P.R.

On the optimal time complexities in decentralized stochastic asynchronous optimization

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

The background is a traditional Chinese ink wash painting. It features a misty, layered mountain landscape. In the foreground, dark, jagged mountain peaks are rendered in shades of blue and grey. The middle ground shows more distant, softer mountain peaks in light grey and pinkish tones. The sky is a pale, hazy white. Scattered throughout the scene are delicate cherry blossom branches with small, pinkish-red flowers and petals, some of which appear to be falling. The overall style is elegant and serene, typical of classical Chinese art.

THE END

The background is a traditional Chinese ink wash painting. It features a misty, layered mountain landscape with soft, hazy peaks. In the foreground, dark, craggy mountain slopes are visible. Delicate, dark brown branches with clusters of pink cherry blossoms and falling petals frame the top and sides of the image. The overall style is elegant and serene, typical of classical Chinese art.

Lecture 2

Selected Fundamentals

refer to my lecture notes

SGD: AC Theory

(for Smooth Strongly Convex Functions)

SGD with Nonuniform & Importance Sampling

SGD with Nice Sampling

SGD with Independent Sampling

SGD with Multi-Sampling

SGD with Arbitrary Sampling

SGD: Adaptive Stepsizes Theory

SGD with Random Reshuffling

Variance Reduction Theory for SGD (for Smooth Strongly Convex Functions)

Loopless SVRG: Variance Reduction for Subsampling

SGD with Nonuniform & Importance Sampling

SGD: AC Theory

(for Smooth Convex Functions)

SPPM Theory: From Smoothness to Similarity

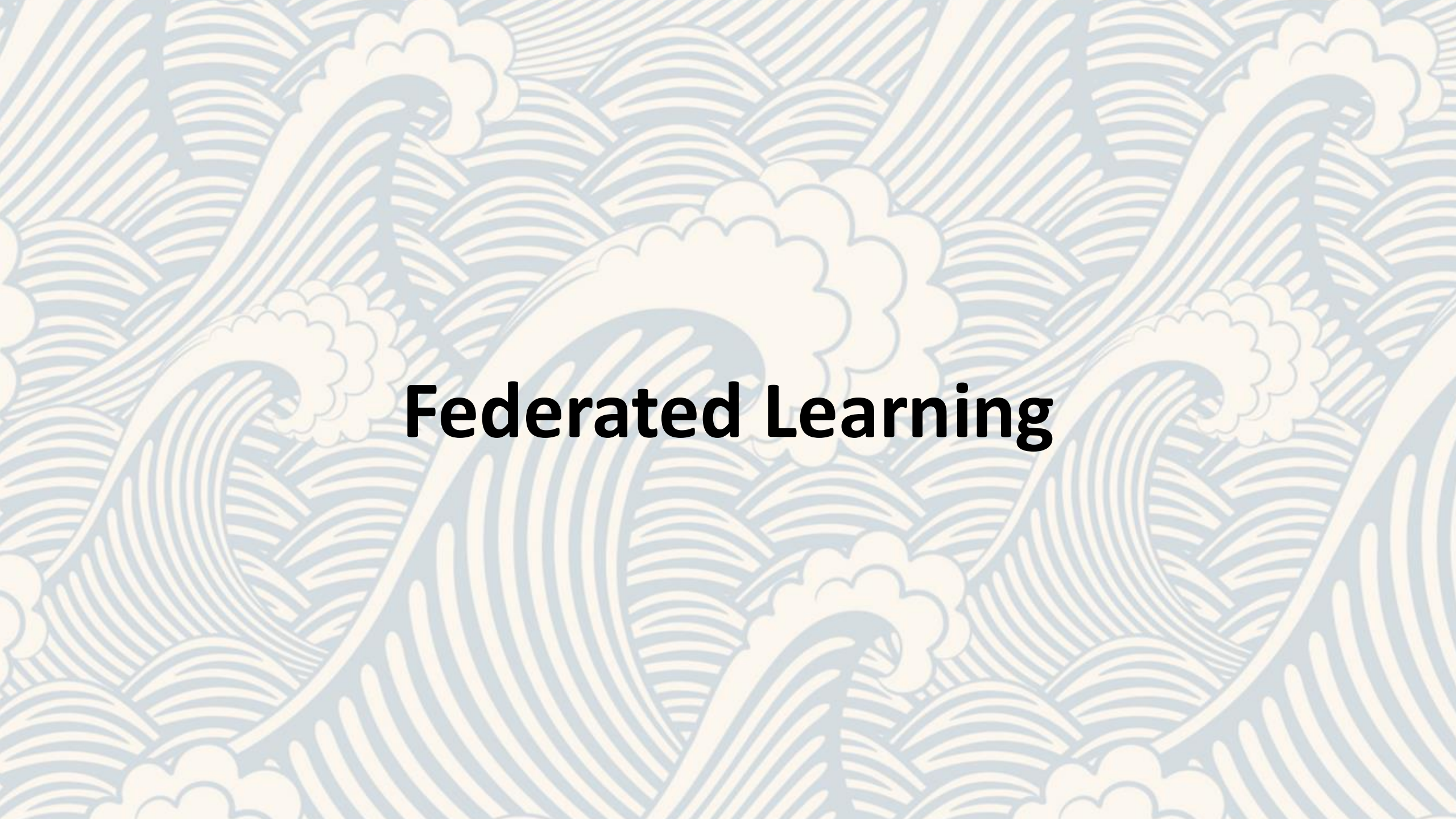
SGD: ABC Theory

(for Smooth Non-Convex Functions)

PAGE: Optimal SGD **(in the Smooth Nonconvex Regime)**

Optimal Client Sampling

General Sampling Patterns

The background of the slide is a repeating pattern of stylized waves. The waves are rendered in a light blue color against a white background. Each wave is composed of multiple curved lines, giving it a sense of movement and depth. The pattern is dense and covers the entire area of the slide.

Federated Learning



Jakub Konečný



H Brendan McMahan



THE UNIVERSITY
of EDINBURGH

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

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

Google AI Blog
The latest from Google Research

Federated Learning: Collaborative Machine Learning without Centralized Training Data

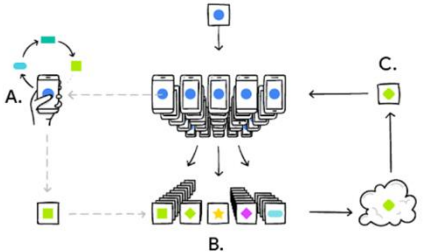
Thursday, April 6, 2017

Posted by Brendan McMahan and Daniel Ramage, Research Scientists

Standard machine learning approaches require centralizing the training data on one machine or in 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.

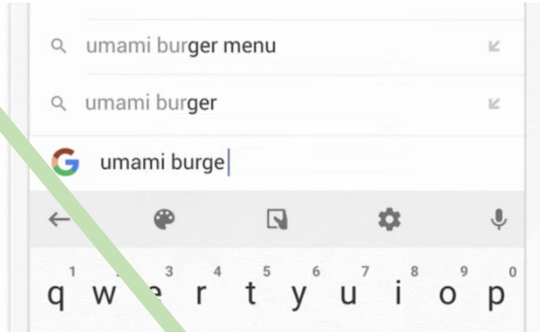


Your phone personalizes the model locally, based on your usage (A). Many users' updates are aggregated (B) to form a consensus change (C) to the shared model, after which the procedure is repeated.

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

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.



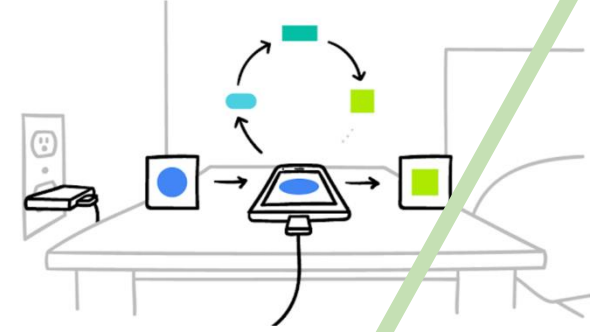
To make Federated Learning possible, we had to overcome many algorithmic and technical challenges. In a typical machine learning system, an optimization algorithm like [Stochastic Gradient Descent](#) (SGD) runs on a large dataset partitioned homogeneously across servers in the cloud. Such highly iterative 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 matrix 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.

Jakub Konečný, H. Brendan McMahan, Daniel Ramage, Peter Richtárik
Federated Optimization: Distributed Machine Learning for On-Device Intelligence
arXiv:1610.02527, 2016

Keith Bonawitz et al
Practical Secure Aggregation for Federated Learning on User-Held Data
NIPS Private Multi-Party Machine Learning Workshop, 2016



Your phone participates in Federated Learning only when it won't negatively impact your experience.

The system then needs to communicate and aggregate 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.

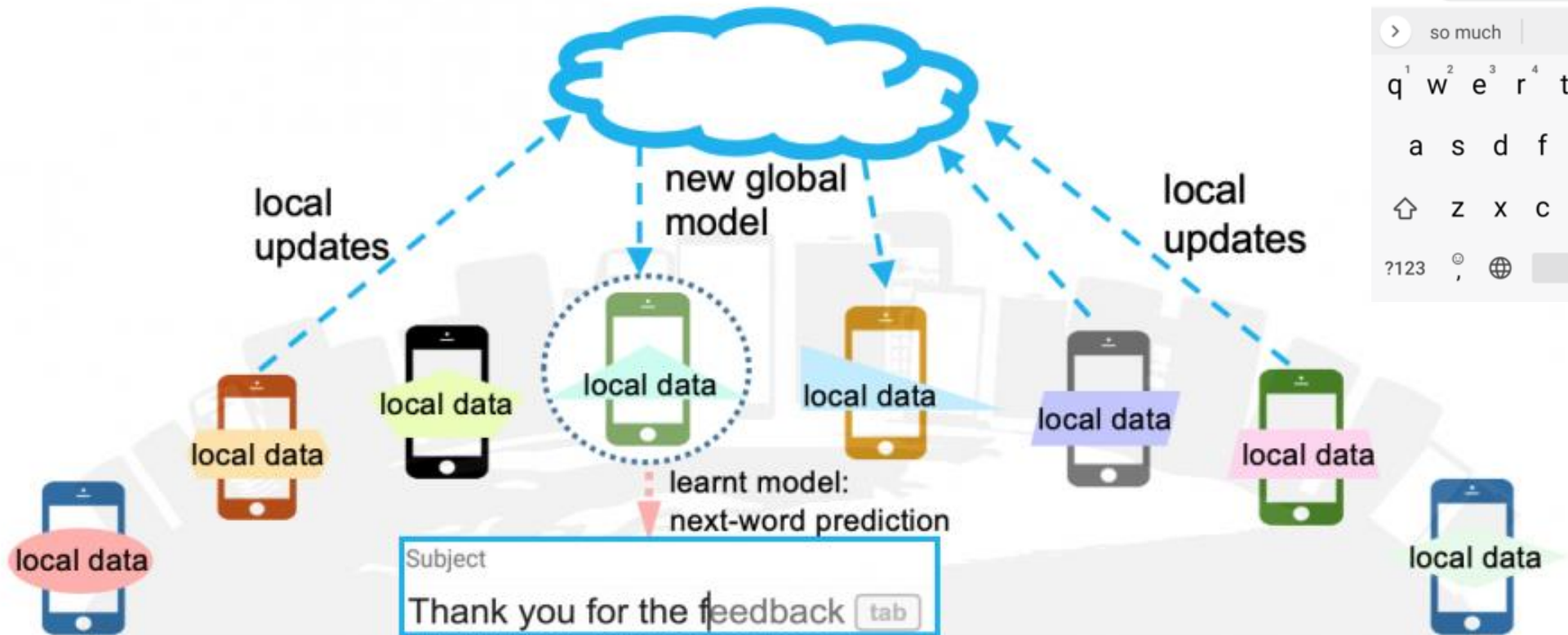
Federated learning works without the need to store user data in the cloud, but we're not stopping there. We've developed a [Secure Aggregation](#) protocol that uses cryptographic techniques so a coordinating server can only decrypt the average update if 100s or 1000s of users have participated — no 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.

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 Il'in, 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]

AI

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>



Meta



FedML



NVIDIA®

WeBank
微众银行



SAMSUNG





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

Executive Summary	vii
Introduction to the <i>National AI R&D Strategic Plan: 2023 Update</i>	1
AI as a National Priority	1
Strategy 1: Make Long-Term Investments in Fundamental and Responsible AI Research	3
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