KAUST
King Abdullah University of Science and Technology

Technical University of Munich

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

Peter Richtárik

FL-ICML'21 Workshop
July 24, 2021

Peter Richtárik, Igor Sokolov and Ilyas Fatkhullin
EF21: A New, Simpler, Theoretically Better, and Practically Faster Error Feedback
ICML 2021 Federated Learning Workshop (FL-ICML'21)
arXiv:2106.05203, 2021

Part 1
Introduction

Training a Federated Learning Model

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

model parameters / features

devices

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

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

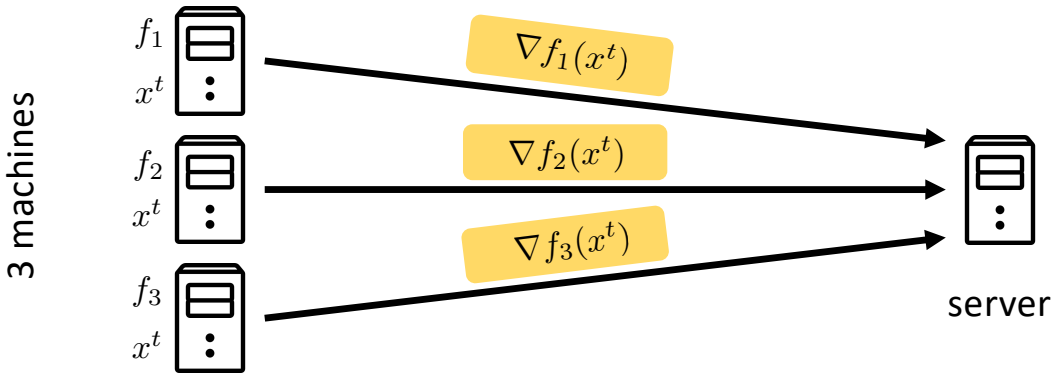
Heterogeneous data regime:

The datasets $\mathcal{D}_1, \mathcal{D}_2, \cdots, \mathcal{D}_n$ are allowed to be **different**

DGD: Distributed Gradient Descent

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

d-dimensional gradient computed by machine *i*



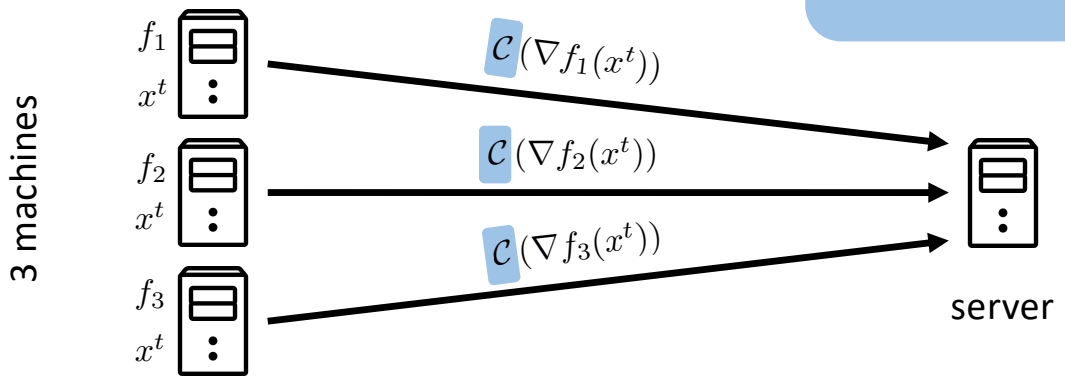
$$x^{t+1} = x^t - \gamma \frac{1}{3} \sum_{i=1}^{3} \nabla f_i(x^t)$$

DCGD: Distributed Compressed Gradient Descent

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

Contractive compression operator

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



$$x^{t+1} = x^t - \gamma \frac{1}{3} \sum_{i=1}^{3} \mathcal{C}\left(\nabla f_i(x^t)\right)$$

However, DCGD Does Not Work in General!

For Contractive Compressors such as Top-K, DCGD Can Diverge Exponentially!!!

arXiv:2002.12410v1 [cs.LG] 27 Feb 2020

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

*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.

1

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

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

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

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

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
arXiv:2002.12410 , 2020

Part 2
Fixing Divergence via Error Feedback

Error Feedback:
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 Seide[1], *Hao Fu*[1,2], *Jasha Droppo*[3], *Gang Li*[1], *and Dong Yu*[3]

[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-

proving 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].

Frank Seide

1-bit stochastic gradient descent and its application to data-parallel distributed training of speech dnns

[PDF] from psu.edu

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

Publication date 2014

Conference Fifteenth Annual Conference of the International Speech Communication Association

Description 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 parallelism. Unexpectedly, quantization benefits AdaGrad, giving a small accuracy gain.

For a typical Switchboard DNN with 46M parameters, we reach computation speeds of 27k frames per second (kfps) when using 2880 samples per minibatch, and 51kfps with 16k, on a server with 8 K20X GPUs. This corresponds to speed-ups over a single GPU of 3.6 and 6.3, respectively. 7 …

Error Feedback

Algorithm 4 EF (Original error feedback)

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

Error Feedback Theory is Very Limited!

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
- Weak $\mathcal{O}(1/T^{2/3})$ 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 Ajalloeian 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 $\nabla f_i(x^\star) = 0\ \forall i$
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

Part 3
Solving All Issues via Error Feedback 2021 (EF21)

Towards Error Feedback 2021

Contractive compressor

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

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

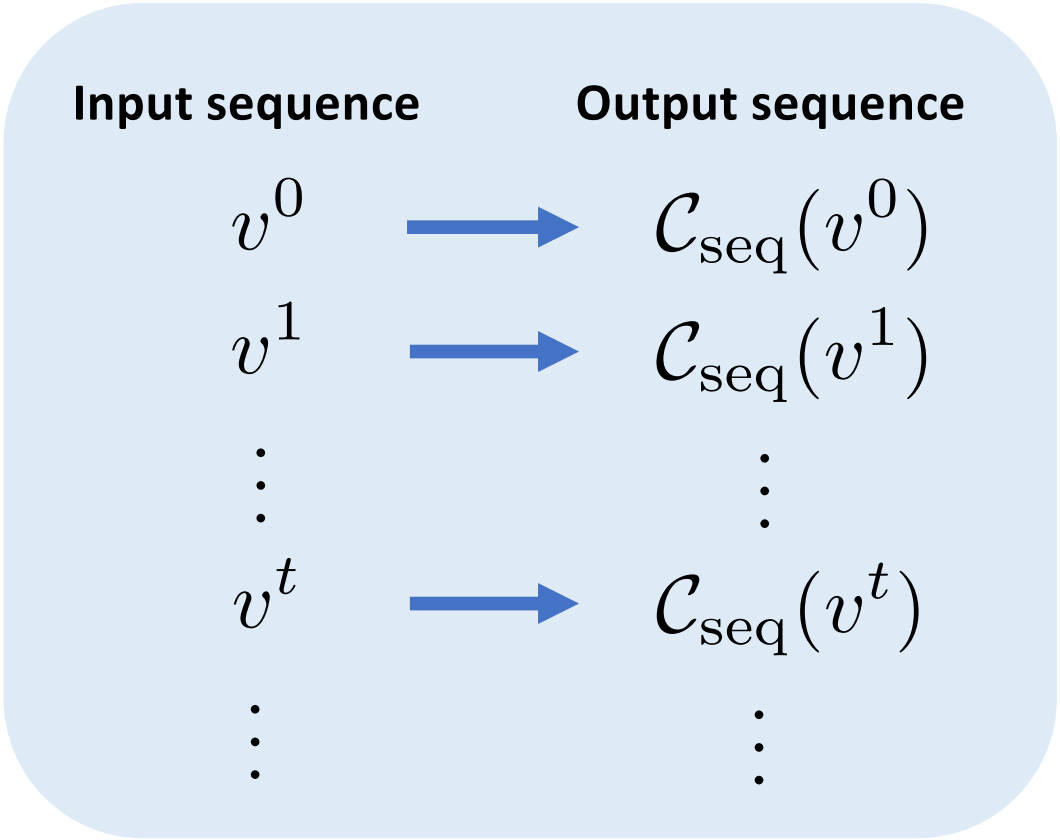
Distributed CGD can diverge exponentially when used with a contractive compressor!

Example [Beznosikov et al, 2020]:
n=*d*=3, *C* = Top-1

Wishful thinking?
A new compressor which handles the errors automatically so that EF is not needed!

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

Compressing a Sequence of Vectors



What we Want:

1. To define $\mathcal{C}_{\text{seq}}$, use $\mathcal{C} \in \mathbb{B}(\alpha)$ only

2. Vanishing compression error

$$\lim_{t\to\infty} \mathbb{E}\left[\left\|\mathcal{C}_{\text{seq}}(v^t) - v^t\right\|^2 \mid v^t\right] \to 0$$

Attempt 1: Naïve Idea

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

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

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

Attempt 2: A Better Idea, but with a Problem

Assumptions:

$$v^\star = \lim_{t\to\infty} v^t$$

$v^\star$ is known by the server

$$\mathcal{C}_{\text{seq}}(v^t) \equiv v^\star + \mathcal{C}(v^t - v^\star)$$

Compression error vanishes in the limit!

$$\mathbb{E}\left[\left\|\mathcal{C}_{\text{seq}}\left(v^t\right) - v^t\right\|^2 \mid v^t\right] = \mathbb{E}\left[\left\|v^\star + \mathcal{C}\left(v^t - v^\star\right) - v^t\right\|^2 \mid v^t\right] \leq (1-\alpha)\left\|v^t - v^\star\right\|^2 \to 0$$

The limit vector is not known!

Contraction property:

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

Attempt 3: Solving the Problem

Assumptions:

$$v^\star = \lim_{t\to\infty} v^t$$

$v^\star$ is known by the server

$$\mathcal{C}_{\text{seq}}(v^t) \equiv \cancel{v^\star} + \mathcal{C}(v^t - \cancel{v^\star})$$

$$v^\star \approx \mathcal{C}_{\text{seq}}\left(v^{t-1}\right)$$

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

Attempt 3: Solving the Problem

Markov property:

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

Definition (Markov compressor)

$$\mathcal{C}_{\text{seq}}\left(v^0\right) \overset{\text{def}}{=} \mathcal{C}\left(v^0\right)$$

$$\mathcal{C}_{\text{seq}}\left(v^t\right) \overset{\text{def}}{=} \mathcal{C}_{\text{seq}}\left(v^{t-1}\right) + \mathcal{C}\left(v^t - \mathcal{C}_{\text{seq}}\left(v^{t-1}\right)\right), \quad t \geq 1$$

Compression error vanishes in the limit!

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

Algorithm 2 EF21 (Multiple nodes)

Input: starting point $x^0 \in \mathbb{R}^d$; $g_i^0 = \mathcal{C}(\nabla f_i(x^0))$ for $i = 1, \ldots, n$ (known by nodes and the master); learning rate $\gamma > 0$; $g^0 = \frac{1}{n}\sum_{i=1}^n g_i^0$ (known by master)
for $t = 0, 1, 2, \ldots, T-1$ **do**
  Master computes $x^{t+1} = x^t - \gamma g^t$ and broadcasts x^{t+1} to all nodes
  for all nodes $i = 1, \ldots, n$ **in parallel do**
    Compress $c_i^t = \mathcal{C}(\nabla f_i(x^{t+1}) - g_i^t)$ and send c_i^t to the master
    Update local state $g_i^{t+1} = g_i^t + \mathcal{C}(\nabla f_i(x^{t+1}) - g_i^t)$
  end for
  Master computes $g^{t+1} = \frac{1}{n}\sum_{i=1}^n g_i^{t+1}$ via $g^{t+1} = g^t + \frac{1}{n}\sum_{i=1}^n c_i^t$
end for

Markov compressor:

$$\mathcal{C}_{\text{seq}}(v^0) \overset{\text{def}}{=} \mathcal{C}(v^0)$$
$$\mathcal{C}_{\text{seq}}(v^t) \overset{\text{def}}{=} \mathcal{C}_{\text{seq}}(v^{t-1}) + \mathcal{C}(v^t - \mathcal{C}_{\text{seq}}(v^{t-1})), \quad t \geq 1$$
$$v^t = \nabla f_i(x^t)$$

Equivalent formulation:

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

EF21 in the Single Node Setting

Algorithm 1 EF21 (Single node)

Input: starting point $x^0 \in \mathbb{R}^d$, learning rate $\gamma > 0$, $g^0 = \mathcal{C}(\nabla f(x^0))$
for $t = 0,1,2,\ldots,T-1$ **do**
    $x^{t+1} = x^t - \gamma g^t$
    $g^{t+1} = g^t + \mathcal{C}(\nabla f(x^{t+1}) - g^t)$
end for

Extensions

1. **EF21+: A More Aggressive Variant of EF21**
2. **Handling Stochastic Gradients**

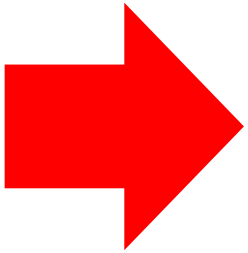
Part 4
Theory of EF21

Restricted Equivalence of EF21 and EF

Theorem

$\mathcal{C}$ is

- deterministic,
- positively homogeneous, and
- additive.



EF21 = EF

Convergence of EF21

Standard assumptions!

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

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

Part 5
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 EF

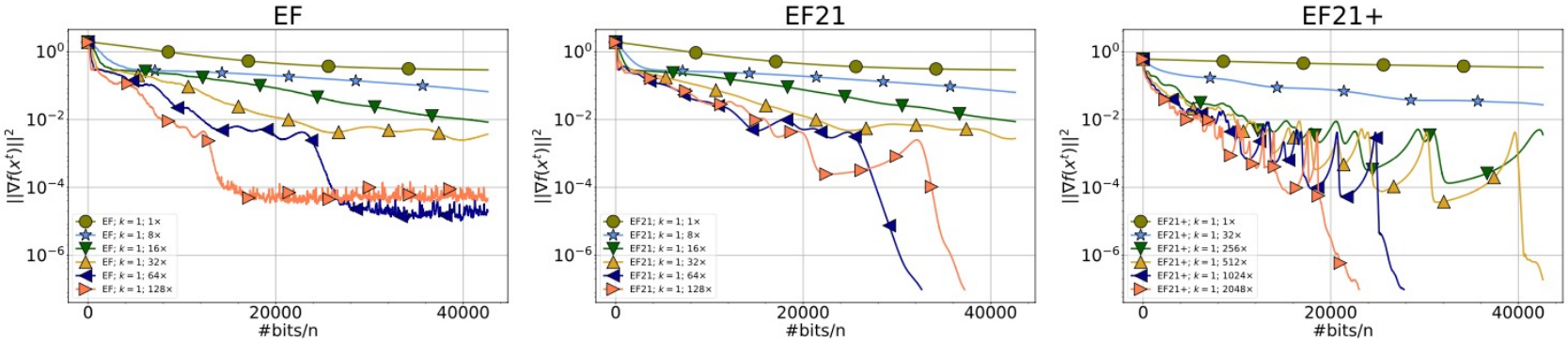


Figure 1: The performance of EF, EF21, and EF21+ with Top-1 compressor, and for increasing stepsizes. Representative dataset used: `a9a`. By $1\times, 2\times, 4\times$ (and so on) we indicate that the stepsize was set to a multiple of the largest stepsize predicted by our theory.

EF, EF21 and EF21+ with Best Fixed Stepsizes

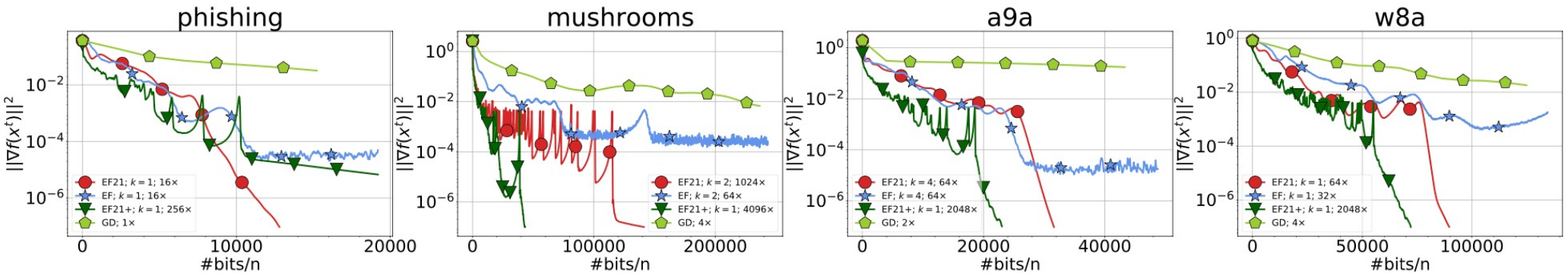


Figure 2: Comparison of EF21, EF21+ to EF with Top-k for individually fine-tuned k and fine-tuned stepsizes for all methods.

The END