Lazily Aggregated Quantized Gradient Innovation for Communication-Efficient Federated Learning

Jun Sun, *Student Member, IEEE*, Tianyi Chen, *Member, IEEE*, Georgios B. Giannakis, *Fellow, IEEE*, Qinmin Yang, *Senior Member, IEEE*, and Zaiyue Yang, *Member, IEEE*

Abstract—This paper focuses on communication-efficient federated learning problem, and develops a novel distributed quantized gradient approach, which is characterized by adaptive communications of the quantized gradients. Specifically, the federated learning builds upon the server-worker infrastructure, where the workers calculate local gradients and upload them to the server; then the server obtain the global gradient by aggregating all the local gradients and utilizes it to update the model parameter. The key idea to save communications from the worker to the server is to *quantize* gradients as well as *skip* less informative quantized gradient communications by reusing previous gradients. Quantizing and skipping result in 'lazy' worker-server communications, which justifies the term **L**azily **A**ggregated **Q**uantized (LAQ) gradient. Theoretically, the LAQ algorithm achieves the same linear convergence as the gradient descent in the strongly convex case, while effecting major savings in the communication in terms of transmitted *bits* and communication *rounds*. Empirically, extensive experiments using realistic data corroborate a significant communication reduction compared with state-of-the-art gradient- and stochastic gradient-based algorithms.

Index Terms—Federated learning, communication-efficient, gradient innovation, quantization

1 Introduction

Training today's machine learning functions (models) relies on an enormous amount of data collected by a massive number of mobile devices. This comes with substantial computational cost, and raises serious privacy concerns when the training is centralized. In addition to cloud computing, these considerations drive the vision that future machine learning tasks must be performed to the extent possible in a distributed fashion at the network edge, namely devices [2].

When distributed learning is carried out in a server-worker setup with possibly heterogeneous devices and datasets as well as privacy considerations, it is referred to as *federated learning* [3], [4]. The server updates the learning parameters utilizing the information (usually gradients) collected from local workers, and then broadcasts the parameters to workers. In this setup, the server obtains the aggregate information without requesting the raw data–what also respects privacy and mitigates the computation burden at the server. Such a learning paradigm however, incurs communication overhead that does not scale with the number of workers. This is aggravated in deep learning, which involves high-dimensional learning parameters. In fact, communication delay has become a bottleneck for fully exploiting the distributed computing resources to speed up the training of machine learning models [5], [6].

In this context, communication-efficient federated learning methods have gained popularity recently [3]. Most methods build on simple gradient updates, and are centered around gradient compression to save communication, including gradient *quantization* and *sparsification*, as outlined in the following overview of the prior art in this area.

1.1 Prior Art

*Quantization.*Today's computers usually utilize 32 or 64 bits to quantize the floating point number, which is assumed to be accurate enough in most algorithms. By quantization in this paper, we mean fewer bits are employed. Toward the goal of reducing communications, quantization compresses transmitted information (e.g., the gradient) by limiting the number of bits that represent floating point numbers, and has been successfully applied to several engineering tasks employing wireless sensor networks [7]. In the context of distributed machine learning, a 1-bit binary quantization scheme has been proposed in [8], [9]. Multi-bit quantization methods have been developed in [10], [11], where an adjustable quantization level endows flexibility to balance the communication

- *Jun Sun and Qinmin Yang are with the College of Control Science and Engineering, the State Key Laboratory of Industrial Control Technology, Zhejiang University, Hangzhou 310027, China. E-mail: sunjun16sj@gmail.com, qmyang@zju.edu.cn.*
- *Tianyi Chen is with the Department of Electrical, Computer, and Systems Engineering, Rensselaer Polytechnic Institute, Troy, NY 12180 USA. E-mail: chent18@rpi.edu.*
- *Georgios B. Giannakis is with the Department of Electrical and Computer Engineering and the Digital Technology Center, University of Minnesota, Minneapolis, MN 55455 USA. E-mail: georgios@umn.edu.*
- *Zaiyue Yang is with the Department of Mechanical and Energy Engineering, Southern University of Science and Technology, Shenzhen 518055, China. E-mail: yangzy3@sustech.edu.cn.*

Manuscript received 17 Dec. 2019; revised 11 July 2020; accepted 8 Oct. 2020. Date of publication 23 Oct. 2020; date of current version 4 Mar. 2022. (Corresponding authors: Zaiyue Yang and Qinmin Yang.) Recommended for acceptance by T. Zhang. Digital Object Identifier no. 10.1109/TPAMI.2020.3033286

cost and the convergence rate. Other variants of quantized gradient schemes include ternary quantization [12], variance-reduced quantization [13], error compensation [14], and gradient difference quantizaiton [11], [15]; and it is shown in [11] that the linear convergence rate can be maintained with gradient difference quantization.

Sparsification. Sparsification amounts to discarding some entries of the gradient and the most straightforward scheme is to transmit only gradient components with large enough magnitudes [16]. Surprisingly, the desired accuracy can be attained even with 99 percent of the gradients being dropped in some cases [17]. To reduce information losses, gradient components with small values are accumulated and then applied when they exceed a certain threshold [18]. The accumulated gradient offers variance reduction for the sparsified stochastic gradient descent (SGD) iterates [19]. With its impressive empirical performance granted, apart from recent efforts [20], deterministic sparsification schemes fall short in performance guarantees. Their randomized counterparts though come with the so-termed unbiased sparsification, which provably offers convergence guarantees [21], [22].

Quantization and sparsification can also be employed jointly [3], [23], [24]. Nevertheless, they both introduce noise to (S)GD updates, which deteriorates convergence in general. For problems with strongly convex losses, gradient compression algorithms either converge linearly to the neighborhood of the optimal solution, or, converge at sublinear rate. The exception is [11], which only focuses on reducing the required bits per communication, but not the total number of rounds. However, for message exchanging, e.g., the p-dimensional model parameter or the gradient, other latencies, such as, initiating communication links, queueing, and propagating the message, can be comparable to the message size-dependent transmission latency [25]. This motivates saving the number of communication rounds, sometimes even more than the bits per round.

Apart from the aforementioned gradient compression approaches, communication-efficient schemes aiming to reduce communication rounds have been developed by leveraging higher-order information [26], [27], periodic aggregation [4], [28], [29], and recently by adaptive aggregation [30], [31], [32]; see also [33] for a lower bound on the number of communication rounds. However, simultaneously saving communication bits and rounds without sacrificing the desired convergence guarantees has not been addressed, and constitutes the goal of this paper. Note that asynchronous algorithms, such as DC-ASGD [34], which primarily aim to save run time can also result in communication reduction. Our method and asynchronous schemes are complementary to each other, and our algorithms to be presented can also be extended to the asynchronous version.

1.2 Context and Contributions in a Nutshell

We first review the standard distributed server-worker learning architecture that typically aims at solving an optimization problem of the form

$$\min_{\boldsymbol{\theta}} \sum_{m\in\mathcal{M}} f_m(\boldsymbol{\theta}) \quad \text{with} \quad f_m(\boldsymbol{\theta}) := \sum_{n=1}^{N_m} \ell(\mathbf{x}_{m,n};\boldsymbol{\theta}), \tag{1}$$

where $\boldsymbol{\theta}\in\mathbb{R}^p$ denotes the parameter to be learned; $\mathcal{M}$ with $|\mathcal{M}|=M$ is the set of workers; $\mathbf{x}_{m,n}$ represents the nth data vector at worker m (e.g., feature vector and its label); N_m is the number of data samples at worker m; while $\ell(\mathbf{x};\boldsymbol{\theta})$ denotes the loss associated with $\boldsymbol{\theta}$ and $\mathbf{x}$; and $f_m(\boldsymbol{\theta})$ stands for the local loss corresponding to $\boldsymbol{\theta}$ and all data at worker m. For ease in exposition, we further let $f(\boldsymbol{\theta}) := \sum_{m\in\mathcal{M}} f_m(\boldsymbol{\theta})$ denote the overall loss function.

Throughout this paper, we consider implementing distributed gradient descent (GD) in the commonly employed worker-server setup. Since the data samples are distributed across the workers, in each iteration the workers need to download the model parameter from the server, calculate the local gradient using local data and upload the gradients to the server; upon receiving all the local gradients, the server then updates the parameter vector following the GD iteration

$$\boldsymbol{GD}: \qquad \boldsymbol{\theta}^{k+1} = \boldsymbol{\theta}^k - \alpha \sum_{m\in\mathcal{M}} \nabla f_m\big(\boldsymbol{\theta}^k\big), \tag{2}$$

where superscript k indexes the iteration, α is the stepsize, and $\nabla f(\boldsymbol{\theta}^k) = \sum_{m\in\mathcal{M}} \nabla f_m(\boldsymbol{\theta}^k)$ is the aggregated (or, global) gradient. It is clear that to implement (2), the server has to communicate with *all* workers to obtain 'fresh' gradients $\{\nabla f_m\big(\boldsymbol{\theta}^k\big)\}_{m=1}^M$. In several settings though, communication is much slower than computation [5]. Thus, as the number of workers grows, worker-server communications become the bottleneck [35]. This becomes more challenging when adopting popular deep learning models with high-dimensional parameters, and correspondingly large gradients. This clearly prompts the need for communication-eficient learning.

Before introducing our communication-efficient learning approach, we revisit the canonical form of popular quantized (Q) GD methods [8], [9], [10], [11], [12], [13], [14], [15] in the simple setup of (1) with one server and M workers

$$\boldsymbol{QGD}: \qquad \boldsymbol{\theta}^{k+1} = \boldsymbol{\theta}^k - \alpha \sum_{m\in\mathcal{M}} Q_m\big(\boldsymbol{\theta}^k\big), \tag{3}$$

where $Q_m\big(\boldsymbol{\theta}^k\big)$ is the quantized gradient that coarsely approximates the local gradient $\nabla f_m(\boldsymbol{\theta}^k)$. While the exact quantization scheme varies across algorithms, transmitting $Q_m\big(\boldsymbol{\theta}^k\big)$ generally requires fewer bits than transmitting its accurate counterpart $\nabla f_m(\boldsymbol{\theta}^k)$. Similar to GD however, only when all the local quantized gradients $\{Q_m\big(\boldsymbol{\theta}^k\big)\}$ are collected, the server can update $\boldsymbol{\theta}$.

In this context, the present paper puts forth a quantized gradient innovation method (as simple as QGD) that also *skips* communication rounds. Different from the downlink server-to-worker communications that can be performed simultaneously (e.g., by broadcasting $\boldsymbol{\theta}^k$), the server in the uplink receives the workers' gradients in the presence of interference, whose mitigation costs resources, e.g., extra latency or bandwidth. For this reason, our focus here is on reducing the number of worker-to-server uplink communications, which we will also refer to as uploads. Our **L**azily **A**ggregated **Q**uantized (**LAQ**) GD update is given by (cf. (3))

$$\boldsymbol{LAQ}: \ \boldsymbol{\theta}^{k+1} = \boldsymbol{\theta}^k - \alpha\nabla^k \ with \ \nabla^k = \nabla^{k-1} + \sum_{m\in\mathcal{M}^k} \delta Q_m^k, \tag{4}$$

where ∇^k is an approximate aggregated gradient that summarizes the parameter change at iteration k, and

$\delta Q_m^k := Q_m(\theta^k) - Q_m(\hat{\theta}_m^{k-1})$ is the difference between two quantized gradients of f_m at the current iterate θ^k and the previous copy $\hat{\theta}_m^{k-1}$. With a judiciously selected criterion that will be elaborated later, $\mathcal{M}^k$ denotes the subset of workers whose local δQ_m^k is uploaded in iteration k, while the parameter vector iterates are given by $\hat{\theta}_m^k := \theta^k$, $\forall m \in \mathcal{M}^k$, and $\hat{\theta}_m^k := \hat{\theta}_m^{k-1}$, $\forall m \notin \mathcal{M}^k$. For worker m, the copy $\hat{\theta}_m^{k-1}$ is employed to remember the model parameter when last time it is selected to communicate with the server.

In comparison to QGD as (3) where 'fresh' quantized gradient is required from each and every worker, the key idea of LAQ is to obtain ∇^k by refining the previous aggregated gradient ∇^{k-1} with the selected gradient differences $\{\delta Q_m^k\}_{m\in\mathcal{M}^k}$; that is, using only the new gradients from the *selected* workers in $\mathcal{M}^k$, while reusing the outdated gradients from the rest of the workers. With ∇^{k-1} stored in the server, this simple modification scales down the per-iteration communication rounds from QGD's M to LAQ's $|\mathcal{M}^k|$. Note that one round of communication through out this paper means one worker's upload.

Compared with alternative quantization schemes, we have that i) LAQ quantizes the gradient innovation—the difference of the current gradient relative to the previous quantized gradient; and ii) LAQ skips the gradient communication—if the gradient innovation of a worker is not significant enough, the communication of this worker is skipped. We will rigorously establish that LAQ achieves the same linear convergence as GD under the strongly convex assumption on the loss function. Numerical tests will demonstrate that our approach outperforms competing methods in terms of both communication bits and rounds.

Notation. Bold lowercase fonts will be used to denote column vectors; $\|\mathbf{x}\|_2$ and $\|\mathbf{x}\|_\infty$ the ℓ_2-norm and ℓ_∞-norm of $\mathbf{x}$, respectively; and $[\mathbf{x}]_i$ the i-th entry of $\mathbf{x}$; while $\lfloor a \rfloor$ will stand for the floor of a; and $|\cdot|$ for the cardinality of a set or vector.

2 LAQ: A Lazily Aggregated Quantized Gradient Approach

With the goal of reducing the communication overhead, two complementary techniques are incorporated in our algorithm design: i) gradient innovation-based quantization; and ii) gradient innovation-based uploading or aggregation—giving the name **L**azily **A**ggregated **Q**uantized gradient. The former reduces the number of bits per upload, while the latter cuts down the number of uploads, and jointly they effect parsimony in communications. The remainder of this section elaborates further on LAQ.

2.1 Gradient Innovation-Based Quantization

Quantization limits the number of bits to encode a vector during communication. Suppose we use b bits to quantize each coordinate of the gradient in contrast to 32 or 64 bits used by most computers. With $\mathcal{Q}$ denoting the quantization operator, the quantized gradient per worker m at iteration k is $Q_m(\theta^k) = \mathcal{Q}(\nabla f_m(\theta^k), Q_m(\hat{\theta}_m^{k-1}))$, which depends on the gradient $\nabla f_m(\theta^k)$ and its previous quantization $Q_m(\hat{\theta}_m^{k-1})$. The gradient is element-wise quantized by projecting to the closest point in a uniformly discretized grid. The grid is a p-dimensional hypercube with center at $Q_m(\hat{\theta}_m^{k-1})$ and radius $R_m^k = \|\nabla f_m(\theta^k) - Q_m(\hat{\theta}_m^{k-1})\|_\infty$. With $\tau := 1/(2^b - 1)$ defining the quantization granularity, the gradient innovation $[f_m(\theta^k)]_i - [Q_m(\hat{\theta}_m^{k-1})]_i$ at worker m is mapped to an integer as

$$[q_m(\theta^k)]_i = \left\lfloor \frac{[\nabla f_m(\theta^k)]_i - [Q_m(\hat{\theta}_m^{k-1})]_i + R_m^k}{2\tau R_m^k} + \frac{1}{2} \right\rfloor, \tag{5}$$

which falls in $\{0, 2, \ldots, 2^b - 1\}$, and thus can be encoded by b bits. Note that adding R_m^k in the numerator ensures the non-negativity of $[q_m(\theta^k)]_i$, and adding $1/2$ in (5) guarantees rounding to the closest point. Hence, the *quantized* gradient innovation at worker m is (with $\mathbf{1} := [1 \cdots 1]^\top$)

$$\delta Q_m^k = Q_m(\theta^k) - Q_m(\hat{\theta}_m^{k-1}) = 2\tau R_m^k q_m(\theta^k) - R_m^k \mathbf{1}, \tag{6}$$

which can be transmitted by $32 + bp$ bits (32 bits for R_m^k and bp bits for $q_m(\theta^k)$) instead of the original $32p$ bits. With the outdated gradients $Q_m(\hat{\theta}_m^{k-1})$ stored in the memory and τ known a priori, upon receiving δQ_m^k the server can recover the quantized gradient as

$$Q_m(\theta^k) = Q_m(\hat{\theta}_m^{k-1}) + \delta Q_m^k. \tag{7}$$

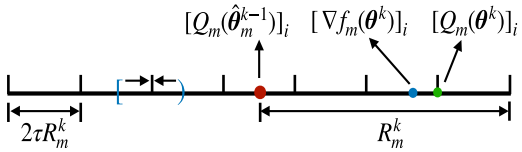


Fig. 1. Quantization example ($b = 3$).

Fig. 1 presents an example for quantizing one coordinate of the gradient with $b = 3$ bits. The original value is quantized with 3 bits and $2^3 = 8$ values, each of which covers an interval of length $2\tau R_m^k$ centered at itself. With $\varepsilon_m^k := \nabla f_m(\theta^k) - Q_m(\theta^k)$ denoting the local quantization error, it is clear that the quantization error is not larger that half of the length of the interval that each value covers, namely,

$$\|\varepsilon_m^k\|_\infty \leq \tau R_m^k. \tag{8}$$

The aggregated quantized gradient is $Q(\theta^k) = \sum_{m\in\mathcal{M}} Q_m(\theta^k)$, and the aggregated quantization error is $\varepsilon^k := \nabla f(\theta^k) - Q(\theta^k) = \sum_{m=1}^M \varepsilon_m^k$; that is, $Q(\theta^k) = \nabla f(\theta^k) - \varepsilon^k$.

2.2 Gradient Innovation-Based Aggregation

The intuition behind lazy gradient aggregation is that if the difference of two consecutive locally quantized gradients is small, it is safe to skip the redundant gradient upload, and reuse the previous one at the server. In addition, we also ensure the server has a relatively "fresh" gradient for each worker by enforcing communication if any worker has not uploaded during the last $\bar{t}$ rounds. We set a clock $t_m,\ m \in \mathcal{M}$ for worker m counting the number of iterations since last time it uploaded information. Equipped with the quantization and selection, our LAQ update takes the form we presented in (4).

Now it only remains to design the selection criterion to decide which worker to upload the quantized gradient or its innovation. We propose the following communication criterion: worker $m \in \mathcal{M}$ skips the upload at iteration k, if it satisfies

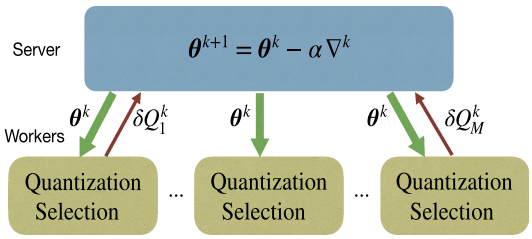


Fig. 2. Federated learning via LAQ.

$$\|Q_m(\hat{\theta}_m^{k-1}) - Q_m(\theta^k)\|_2^2 \le \frac{1}{\alpha^2 M^2}\sum_{d=1}^{D}\xi_d\|\theta^{k+1-d} - \theta^{k-d}\|_2^2 + 3\left(\|\varepsilon_m^k\|_2^2 + \|\hat{\varepsilon}_m^{k-1}\|_2^2\right); \tag{9a}$$

$$t_m \le \bar{t}, \tag{9b}$$

where $D \le \bar{t}$ and $\{\xi_d\}_{d=1}^D$ are predetermined constants, and $\hat{\varepsilon}_m^{k-1} = \nabla f_m(\hat{\theta}_m^{k-1}) - Q_m(\hat{\theta}_m^{k-1})$ is the quantization error of the local gradient when last time worker m uploads gradient innovation to the server. We will prove in the next section that the LAQ iterates in (4) converge, and are communication efficient.

With reference to Fig. 2, LAQ can be summarized as follows. At iteration k, the server broadcasts the learning parameter vector to all workers. Each worker computes the gradient, and quantizes it to decide whether to upload the quantized gradient innovation δQ_m^k. Upon receiving the gradient innovation from selected workers, the server updates the learning vector. These steps are listed in Algorithm 2.

Note that in this paper, by privacy preserving, we mean the private raw data of each worker are not released to other workers nor the server. It is true that the local information might be partially inferred by reverse engineering of the gradient. However, the gradient compression of LAQ introduces noise to the gradient, which helps promote privacy [36], [37]. To further investigate how to mitigate privacy leakage and quantify the degree of privacy remains to be our future work.

3 Convergence and Communication Analysis

Our subsequent convergence analysis of LAQ relies on the following assumptions on $f_m(\cdot)$ and $f(\cdot)$:

Assumption 1. *The local gradient $\nabla f_m(\cdot)$ is L_m-Lipschitz continuous, and the global gradient $\nabla f(\cdot)$ is L-Lipschitz continuous; i.e., there exist constants L_m and L such that*

$$\|\nabla f_m(\theta_1) - \nabla f_m(\theta_2)\|_2 \le L_m\|\theta_1 - \theta_2\|_2,\ \forall\theta_1,\ \theta_2; \tag{10a}$$

$$\|\nabla f(\theta_1) - \nabla f(\theta_2)\|_2 \le L\|\theta_1 - \theta_2\|_2,\ \forall\theta_1,\ \theta_2. \tag{10b}$$

Assumption 2. *The function $f(\cdot)$ is μ-strongly convex, meaning that there exists a constant $\mu > 0$ such that*

$$f(\theta_1) - f(\theta_2) \ge \langle\nabla f(\theta_2), \theta_1 - \theta_2\rangle + \frac{\mu}{2}\|\theta_1 - \theta_2\|_2^2, \quad \forall\theta_1,\ \theta_2.$$

Algorithm 1. QGD

Input: stepsize $\alpha > 0$, quantization bit b.
Initialize: θ^0.
for $k = 1, 2, \ldots, K$ **do**
  Server broadcasts θ^k to all workers.
  for $m = 1, 2, \ldots, M$ **do**
    Worker m computes $\nabla f_m(\theta^k)$ and $Q_m(\theta^k)$.
    Worker m uploads δQ_m^k via (6).
  end for
  Server updates θ following (4) with $\mathcal{M}^k = \mathcal{M}$.
end for

Algorithm 2. LAQ

Input: stepsize $\alpha > 0$, b, D, $\{\xi_d\}_{d=1}^D$ and $\bar{t}$.
Initialize: θ^k, and $\{Q_m(\hat{\theta}_m^0), t_m\}_{m\in\mathcal{M}}$.
for $k = 1, 2, \ldots, K$ **do**
  Server broadcasts θ^k to all workers.
  for $m = 1, 2, \ldots, M$ **do**
    Worker m computes $\nabla f_m(\theta^k)$ and $Q_m(\theta^k)$.
    if (9) holds for worker m **then**
      Worker m uploads nothing.
      Set $\hat{\theta}_m^k = \hat{\theta}_m^{k-1}$ and $t_m \leftarrow t_m + 1$.
    else
      Worker m uploads δQ_m^k via (6).
      Set $\hat{\theta}_m^k = \theta^k$, and $t_m = 0$.
    end if
  end for
  Server updates θ according to (4).
end for

Before establishing our performance analysis results, we first present salient features of our communication skipping rule. The rationale behind the selection criterion (9) is to judiciously compare the descent amount of GD versus that of LAQ.

3.1 Development of the Communication Skipping Rule

To illuminate the difference between LAQ and GD, consider re-writing (4) as

$$\begin{aligned}\theta^{k+1} &= \theta^k - \alpha[\nabla Q(\theta^k) + \sum_{m\in\mathcal{M}_c^k}(Q_m(\hat{\theta}_m^{k-1}) - Q_m(\theta^k))] \\ &= \theta^k - \alpha[\nabla f(\theta^k) - \varepsilon^k + \sum_{m\in\mathcal{M}_c^k}(Q_m(\hat{\theta}_m^{k-1}) - Q_m(\theta^k))],\end{aligned}$$

where $\mathcal{M}_c^k := \mathcal{M}\backslash\mathcal{M}^k$ denotes the subset of workers that skip communication with the server at iteration k. Compared with the GD iteration in (2), the gradient employed here degrades due to the quantization error ε^k and the missed gradient innovation $\sum_{m\in\mathcal{M}_c^k}(Q_m(\hat{\theta}^{k-1}) - Q_m(\theta^k))]$. It is clear that if a sufficiently large number of bits is used to quantize the gradient, and all $\{\xi_d\}_{d=1}^D$ are set to 0, causing $\mathcal{M}^k := \mathcal{M}$, then LAQ reduces to GD. Thus, adjusting b and $\{\xi_d\}_{d=1}^D$ directly influences the performance of LAQ.

To compare the descent amount of LAQ with that of GD, we first establish the one step descent for both algorithms. Based on Assumption 1, the next lemma holds for GD.

Lemma 1. *The GD update yields the following descent*

$$f(\boldsymbol{\theta}^{k+1}) - f(\boldsymbol{\theta}^k) \leq \Delta_{GD}^k, \tag{11}$$

where $\Delta_{GD}^k := -(1-\frac{\alpha L}{2})\alpha\|\nabla f(\boldsymbol{\theta}^k)\|_2^2$.

The descent of LAQ differs from that of GD due to the quantization and selection, as specified in the next lemma. (For readability, some proofs are deferred to Section 7)

Lemma 2. *The LAQ update yields the following descent*

$$f(\boldsymbol{\theta}^{k+1}) - f(\boldsymbol{\theta}^k) \leq \Delta_{LAQ}^k + \alpha\|\varepsilon^k\|_2^2, \tag{12}$$

where $\Delta_{LAQ}^k := -\frac{\alpha}{2}\|\nabla f(\boldsymbol{\theta}^k)\|_2^2 + \alpha\|\sum_{m\in\mathcal{M}_c^k}(Q_m(\hat{\boldsymbol{\theta}}_m^{k-1}) - Q_m(\boldsymbol{\theta}^k))\|_2^2 + (\frac{L}{2}-\frac{1}{2\alpha})\|\boldsymbol{\theta}^{k+1}-\boldsymbol{\theta}^k\|_2^2$.

At this point, it is instructive to shed more light on LAQ's gradient skipping rule. If we fix for simplicity $\alpha = 1/L$, it follows readily that

$$\Delta_{GD}^k = -\frac{\alpha}{2}||\nabla f(\boldsymbol{\theta}^k)||_2^2;$$
$$\Delta_{LAQ}^k = -\frac{\alpha}{2}||\nabla f(\boldsymbol{\theta}^k)||_2^2 + \alpha||\sum_{m\in\mathcal{M}_c^k}(Q_m(\hat{\boldsymbol{\theta}}_m^{k-1}) - Q_m(\boldsymbol{\theta}^k))||_2^2.$$

The lazy aggregation criterion selects the quantized gradient innovation by assessing its contribution to the loss function decrease. For LAQ to be more communication efficient than GD, each LAQ upload should bring more descent, that is

$$\frac{\Delta_{LAQ}^k}{|\mathcal{M}^k|} \leq \frac{\Delta_{GD}^k}{M}\ . \tag{13}$$

After simple manipulation, it can be shown that (13) boils down to

$$||\sum_{m\in\mathcal{M}_c^k}(Q_m(\hat{\boldsymbol{\theta}}_m^{k-1}) - Q_m(\boldsymbol{\theta}^k))||_2^2 \leq \frac{|\mathcal{M}_c^k|}{2M}||\nabla f(\boldsymbol{\theta}^k)||_2^2, \tag{14}$$

which implies that since

$$\begin{aligned}&||\sum_{m\in\mathcal{M}_c^k}(Q_m(\hat{\boldsymbol{\theta}}_m^{k-1}) - Q_m(\boldsymbol{\theta}^k))||_2^2\\ &\leq |\mathcal{M}_c^k|\sum_{m\in\mathcal{M}_c^k}||(Q_m(\hat{\boldsymbol{\theta}}_m^{k-1}) - Q_m(\boldsymbol{\theta}^k)||_2^2,\end{aligned} \tag{15}$$

the following condition is sufficient to guarantee (14):

$$||(Q_m(\hat{\boldsymbol{\theta}}_m^{k-1}) - Q_m(\boldsymbol{\theta}^k)||_2^2 \leq ||\nabla f(\boldsymbol{\theta}^k)||_2^2/(2M^2),\ \forall m\in\mathcal{M}_c^k. \tag{16}$$

However, it is impossible to check (16) locally per worker, because the fully aggregated gradient $\nabla f(\boldsymbol{\theta}^k)$ is required, which is exactly what we want to avoid. This motivates circumventing $||\nabla f(\boldsymbol{\theta}^k)||_2^2$ by using its approximation

$$||\nabla f(\boldsymbol{\theta}^k)||_2^2 \approx \frac{2}{\alpha^2}\sum_{k=1}^{D}\xi_d||\boldsymbol{\theta}^{k+1-d} - \boldsymbol{\theta}^{k-d}||_2^2, \tag{17}$$

where $\{\xi_d\}_{d=1}^D$ are constants. The main reason why (17) holds is that $\nabla f(\boldsymbol{\theta}^k)$ can be approximated by weighting past gradients or parameter differences since $f(\cdot)$ is L-smooth. Combining (17) and (16) leads to (9a) with the quantization error ignored.

3.2 Convergence Analysis

The rationale of the previous subsection regarding LAQ's skipping rule is not mathematically rigorous, but we will establish here that it guarantees convergent iterates. To this end, and with $\boldsymbol{\theta}^*$ denoting the optimal solution of (1), consider the Lyapunov function associated with LAQ as

$$\begin{aligned}\mathbb{V}(\boldsymbol{\theta}^k) :=\ & f(\boldsymbol{\theta}^k) - f(\boldsymbol{\theta}^*)\\ &+ \sum_{d=1}^{D}\sum_{j=d}^{D}\frac{\xi_j}{\alpha}\|\boldsymbol{\theta}^{k+1-d} - \boldsymbol{\theta}^{k-d}\|_2^2 + \gamma\sum_{m\in\mathcal{M}}\|\varepsilon_m^k\|_\infty^2.\end{aligned} \tag{18}$$

Before we quantify the process of $\mathbb{V}(\boldsymbol{\theta}^k)$ in the ensuing lemma, it is worth pointing out that the Lyapunov function associated with LAQ is a strict generalization of that used in GD or LAG [30], [31], which not only takes into account the delayed iterates but also the quantization error.

Lemma 3. *Under Assumptions 1 and 2, and by fixing parameters* $\xi_1 = \xi_2 = \cdots = \xi$, $\beta_d = \frac{(D-d+1)\xi}{\alpha}$, $\alpha = \frac{a}{L}$, *and* $\gamma\tau^2 = \frac{bL}{L_m^2}$, *with* $a, b > 0$, *the Lyapunov function obeys the inequality*

$$\mathbb{V}(\boldsymbol{\theta}^{k+1}) \leq \sigma_1\mathbb{V}(\boldsymbol{\theta}^k) + BMp^2\frac{1}{\gamma}\max_{k-t\leq t'\leq k-1}\mathbb{V}(\boldsymbol{\theta}^{t'}), \tag{19}$$

where the constant is defined as

$$B = [\frac{3a}{2L} + (\frac{3a}{2} + 3D\xi + 9ab)\frac{2a}{L} + \frac{9bL}{ML_m^2}]M,$$

and $\sigma_1 = 1 - c$ *with*

$$c = \min\left\{\frac{[\frac{1}{2} - 4(a+2D\xi+6ab)]a}{\kappa}, \frac{\frac{1}{2} - (\frac{1}{2}a + D\xi + 3ab) + \frac{3bL^2}{aL_m^2M}}{D-d+1}\right\}. \tag{20}$$

Proof. It follows from (8) that

$$\begin{aligned}\|\varepsilon_m^{k+1}\|_\infty^2 &\leq \tau^2(R_m^{k+1})^2\\ &= \tau^2\|\nabla f_m(\boldsymbol{\theta}^{k+1}) - \nabla f_m(\boldsymbol{\theta}^k) + \nabla f_m(\boldsymbol{\theta}^k)\\ &\quad - Q_m(\boldsymbol{\theta}^k) + Q_m(\boldsymbol{\theta}^k) - Q_m(\hat{\boldsymbol{\theta}}_m^k)\|_\infty^2\\ &\leq 3\tau^2L_m\|\boldsymbol{\theta}^{k+1} - \boldsymbol{\theta}^k\|_2^2 + 3\tau^2\|\varepsilon_m^k\|_\infty^2\\ &\quad + 3\tau^2\|Q_m(\boldsymbol{\theta}^k) - Q_m(\hat{\boldsymbol{\theta}}_m^k)\|_2^2.\end{aligned} \tag{21}$$

Then the one-step Lyapunov function difference is bounded as

$$
\begin{aligned}
&\mathbb{V}(\boldsymbol{\theta}^{k+1})-\mathbb{V}(\boldsymbol{\theta}^{k})\\
&\le -\alpha\big\langle\nabla f(\boldsymbol{\theta}^{k}),Q(\boldsymbol{\theta}^{k})\big\rangle+\frac{\alpha}{2}\|\nabla f(\boldsymbol{\theta}^{k})\|_2^2\\
&+\frac{\alpha}{2}\|\sum_{m\in\mathcal{M}_c^k}Q_m(\hat{\boldsymbol{\theta}}_m^{k-1})-Q_m(\boldsymbol{\theta}^{k})\|_2^2\\
&+(\frac{L}{2}+\beta_1+3\gamma\tau^2L_m^2)\|\boldsymbol{\theta}^{k+1}-\boldsymbol{\theta}^{k}\|_2^2\\
&+\sum_{d=1}^{D-1}(\beta_{d+1}-\beta_d)\|\boldsymbol{\theta}^{k+1-d}-\boldsymbol{\theta}^{k-d}\|_2^2-\beta_D\|\boldsymbol{\theta}^{k+1-D}-\boldsymbol{\theta}^{k-D}\|_2^2\\
&+\gamma(3\tau^2-1)\sum_{m\in\mathcal{M}}\|\varepsilon_m^k\|_\infty^2+3\gamma\tau^2\sum_{m\in\mathcal{M}}\|Q_m(\hat{\boldsymbol{\theta}}_m^{k-1})-Q_m(\boldsymbol{\theta}^{k})\|_2^2\\
&\le -\alpha\big\langle\nabla f(\boldsymbol{\theta}^{k}),Q(\boldsymbol{\theta}^{k})\big\rangle+\frac{\alpha}{2}\|\nabla f(\boldsymbol{\theta}^{k})\|_2^2\\
&+[\frac{\alpha}{2}+(\frac{L}{2}+\beta_1+3\gamma\tau^2L_m^2)(1+\rho_2^{-1})\alpha^2]\|\sum_{m\in\mathcal{M}_c^k}Q_m(\hat{\boldsymbol{\theta}}_m^{k-1})-Q_m(\boldsymbol{\theta}^{k})\|_2^2\\
&+(\frac{L}{2}+\beta_1+3\gamma\tau^2L_m^2)(1+\rho_2)\alpha^2\|Q(\boldsymbol{\theta}^{k})\|_2^2\\
&+\sum_{d=1}^{D-1}(\beta_{d+1}-\beta_d)\|\boldsymbol{\theta}^{k+1-d}-\boldsymbol{\theta}^{k-d}\|_2^2\\
&-\beta_D\|\boldsymbol{\theta}^{k+1-D}-\boldsymbol{\theta}^{k-D}\|_2^2+\gamma(3\tau^2-1)\sum_{m\in\mathcal{M}}\|\varepsilon_m^k\|_\infty^2\\
&+3\gamma\tau^2\sum_{m\in\mathcal{M}}\|Q_m(\hat{\boldsymbol{\theta}}_m^{k-1})-Q_m(\boldsymbol{\theta}^{k})\|_2^2,
\end{aligned}
$$

where the second inequality follows from Young's inequality, namely $\|\boldsymbol{a}+\boldsymbol{b}\|_2^2\le(1+\rho)\|\boldsymbol{a}\|_2^2+(1+\rho^{-1})\|\boldsymbol{b}\|_2^2$.

Considering the criterion (9a), we have

$$
\begin{aligned}
&\|\sum_{m\in\mathcal{M}_c^k}Q_m(\hat{\boldsymbol{\theta}}_m^{k-1})-Q_m(\boldsymbol{\theta}^{k})\|_2^2\\
&\le|\mathcal{M}_c^k|\sum_{m\in\mathcal{M}_c^k}\|Q_m(\hat{\boldsymbol{\theta}}_m^{k-1})-Q_m(\boldsymbol{\theta}^{k})\|_2^2\\
&\le\frac{|\mathcal{M}_c^k|^2}{\alpha^2|\mathcal{M}|^2}\sum_{d=1}^{D}\xi_d\|\boldsymbol{\theta}^{k+1-d}-\boldsymbol{\theta}^{k-d}\|_2^2+3|\mathcal{M}_c^k|\sum_{m\in\mathcal{M}_c^k}(\|\varepsilon_m^k\|_2^2+\|\hat{\varepsilon}_m^{k-1}\|_2^2)\\
&\le\frac{1}{\alpha^2}\sum_{d=1}^{D}\xi_d\|\boldsymbol{\theta}^{k+1-d}-\boldsymbol{\theta}^{k-d}\|_2^2+3M\sum_{m\in\mathcal{M}_c^k}(\|\varepsilon_m^k\|_2^2+\|\hat{\varepsilon}_m^{k-1}\|_2^2).
\end{aligned}
\tag{22}
$$

Thus, the one-step Lyapunov difference satisfies

$$
\begin{aligned}
&\mathbb{V}(\boldsymbol{\theta}^{k+1})-\mathbb{V}(\boldsymbol{\theta}^{k})\\
&\le-\frac{\alpha}{2}\|\nabla f(\boldsymbol{\theta}^{k})\|_2^2+\alpha\big\langle\nabla f(\boldsymbol{\theta}^{k}),\varepsilon^{k}\big\rangle\\
&+\frac{[\frac{\alpha}{2}+(\frac{L}{2}+\beta_1)(1+\rho_2^{-1})\alpha^2]M+3\gamma\tau^2}{\alpha^2M}\sum_{d=1}^{D}\xi_d\|\boldsymbol{\theta}^{k+1-d}-\boldsymbol{\theta}^{k-d}\|_2^2\\
&+(\frac{L}{2}+\beta_1+3\gamma\tau^2L_m^2)(1+\rho_2)\alpha^2\|\nabla f(\boldsymbol{\theta}^{k})-\varepsilon^{k}\|_2^2\\
&+\sum_{d=1}^{D-1}(\beta_{d+1}-\beta_d)\|\boldsymbol{\theta}^{k+1-d}-\boldsymbol{\theta}^{k-d}\|_2^2-\beta_D\|\boldsymbol{\theta}^{k+1-D}-\boldsymbol{\theta}^{k-D}\|_2^2\\
&+[\frac{3\alpha}{2}+(\frac{3L}{2}+3\beta_1+9\gamma\tau^2L_m^2)(1+\rho_2^{-1})\alpha^2+3\gamma\tau^2]M\\
&\times\sum_{m\in\mathcal{M}_c^k}(\|\varepsilon_m^k\|_2^2+\|\hat{\varepsilon}_m^{k-1}\|_2^2)+\gamma(3\tau^2-1)\sum_{m\in\mathcal{M}}\|\varepsilon_m^k\|_\infty^2.
\end{aligned}
\tag{23}
$$

Since for any $\rho_1>0$, it holds that

$$
\big\langle\nabla f(\boldsymbol{\theta}^{k}),\varepsilon^{k}\big\rangle\le\frac{1}{2}\rho_1\|\nabla f(\boldsymbol{\theta}^{k})\|_2^2+\frac{1}{2\rho_1}\|\varepsilon^{k}\|_2^2,
\tag{24}
$$

we can rewrite (23) as

$$
\begin{aligned}
&\mathbb{V}(\boldsymbol{\theta}^{k+1})-\mathbb{V}(\boldsymbol{\theta}^{k})\\
&\le[(-\frac{1}{2}+\frac{1}{2}\rho_1)\alpha+(L+2\beta_1+6\gamma\tau^2L_m^2)(1+\rho_2)\alpha^2]\|\nabla f(\boldsymbol{\theta}^{k})\|_2^2\\
&+\{[(\frac{\alpha}{2}+(\frac{L}{2}+\beta_1+3\gamma\tau^2L_m^2)(1+\rho_2^{-1})\alpha^2)M+3\gamma\tau^2]\frac{\xi_D}{\alpha^2M}\\
&-\beta_D\}\|\boldsymbol{\theta}^{k+1-D}-\boldsymbol{\theta}^{k-D}\|_2^2\\
&+\sum_{d=1}^{D-1}\{[(\frac{\alpha}{2}+(\frac{L}{2}+\beta_1+3\gamma\tau^2L_m^2)(1+\rho_2^{-1})\alpha^2)M+3\gamma\tau^2]\frac{\xi_d}{\alpha^2M}\\
&+\beta_{d+1}-\beta_d\}\|\boldsymbol{\theta}^{k+1-d}-\boldsymbol{\theta}^{k-d}\|_2^2\\
&+[\frac{3\alpha}{2}+(\frac{3L}{2}+3\beta_1+9\gamma\tau^2L_m^2)(1+\rho_2^{-1})\alpha^2+3\gamma\tau^2]M\\
&\times\sum_{m\in\mathcal{M}_c^k}(\|\varepsilon_m^k\|_2^2+\|\hat{\varepsilon}_m^{k-1}\|_2^2)\\
&+[\frac{1}{2\rho_1}\alpha+(L+2\beta_1+6\gamma\tau^2L_m^2)(1+\rho_2)\alpha^2]\|\varepsilon^{k}\|_2^2\\
&+\gamma(3\tau^2-1)\sum_{m\in\mathcal{M}}\|\varepsilon_m^k\|_\infty^2.
\end{aligned}
\tag{25}
$$

It is straightforward that the following condition guarantees the first three terms in (25) are nonpositive

$$
\begin{aligned}
&(-\frac{1}{2}+\frac{1}{2}\rho_1)\alpha+(L+2\beta_1+6\gamma\tau^2L_m^2)(1+\rho_2)\alpha^2\le0;\\
&\frac{\{[\frac{\alpha}{2}+(\frac{L}{2}+\beta_1+3\gamma\tau^2L_m^2)(1+\rho_2^{-1})\alpha^2]M+3\gamma\tau^2\}\xi_D}{\alpha^2M}\le\beta_D;\\
&\frac{\{[\frac{\alpha}{2}+(\frac{L}{2}+\beta_1+3\gamma\tau^2L_m^2)(1+\rho_2^{-1})\alpha^2]M+3\gamma\tau^2\}\xi_d}{\alpha^2M}\le\beta_d-\beta_{d+1}.
\end{aligned}
\tag{26}
$$

For ease of exposition, we define the constants c and B as

$$
\begin{aligned}
c=\min\Big\{&(1-\rho_1)\alpha-2\mu(L+2\beta_1+6\gamma\tau^2L_m^2)(1+\rho_2)\alpha^2,\\
&1-[(\frac{\alpha}{2}+(\frac{L}{2}+\beta_1+3\gamma\tau^2L_m^2)(1+\rho_2^{-1})\alpha^2)M+3\gamma\tau^2]\frac{\xi_D}{\alpha^2M\beta_D},\\
&1-\frac{\beta_{d+1}}{\beta_d}-\frac{[(\frac{\alpha}{2}+(\frac{L}{2}+\beta_1+3\gamma\tau^2L_m^2)(1+\rho_2^{-1})\alpha^2)M+3\gamma\tau^2]\xi_d}{\alpha^2M\beta_d}\Big\},
\end{aligned}
\tag{27}
$$

and,

$$
\begin{aligned}
B=\max\Big\{&\Big[\frac{3\alpha}{2}+(\frac{3L}{2}+3\beta_1+9\gamma\tau^2L_m^2)(1+\rho_2^{-1})\alpha^2+3\gamma\tau^2\Big]M,\\
&\frac{1}{2\rho_1}\alpha+(L+2\beta_1+6\gamma\tau^2L_m^2)(1+\rho_2)\alpha^2\Big\}.
\end{aligned}
$$

Assumption 2 implies that $f(\cdot)$ satisfies the PL condition

$$2\mu(f(\boldsymbol{\theta}^k)-f(\boldsymbol{\theta}^*)) \le \|\nabla f(\boldsymbol{\theta}^k)\|_2^2. \tag{28}$$

Plugging (28) into (25) gives

$$\begin{aligned}\mathbb{V}(\boldsymbol{\theta}^{k+1}) &\le \sigma_1\mathbb{V}(\boldsymbol{\theta}^k)+B[\|\boldsymbol{\varepsilon}^k\|_2^2+\sum_{m\in\mathcal{M}}(\|\boldsymbol{\varepsilon}_m^k\|_2^2+\|\hat{\boldsymbol{\varepsilon}}_m^{k-1}\|_2^2)]\\ &\quad+\gamma(3\tau^2-1)\sum_{m\in\mathcal{M}}\|\boldsymbol{\varepsilon}_m^k\|_\infty^2\\ &\le \sigma_1\mathbb{V}(\boldsymbol{\theta}^k)+[BMp^2+B+\gamma(3\tau^2-1)]\sum_{m\in\mathcal{M}}\|\boldsymbol{\varepsilon}_m^k\|_2^2\\ &\quad+Bp^2\sum_{m\in\mathcal{M}}\|\hat{\boldsymbol{\varepsilon}}_m^{k-1}\|_2^2,\end{aligned}$$

where $\sigma_1 = 1-c$.

By choosing parameter stepsize α that impose the following inequality hold

$$[BMp^2+B+\gamma(3\tau^2-1)] \le 0, \tag{29}$$

one can obtain

$$\begin{aligned}\mathbb{V}(\boldsymbol{\theta}^{k+1}) &\le \sigma_1\mathbb{V}(\boldsymbol{\theta}^k)+Bp^2\frac{1}{\gamma}\cdot\gamma\sum_{m\in\mathcal{M}}\|\hat{\boldsymbol{\varepsilon}}_m^{k-1}\|_2^2\\ &\le \sigma_1\mathbb{V}(\boldsymbol{\theta}^k)+Bp^2\frac{1}{\gamma}\sum_{m\in\mathcal{M}}\max_{k-\bar{t}\le t'\le k-1}\mathbb{V}(\boldsymbol{\theta}^{t'})\\ &\le \sigma_1\mathbb{V}(\boldsymbol{\theta}^k)+BMp^2\frac{1}{\gamma}\max_{k-\bar{t}\le t'\le k-1}\mathbb{V}(\boldsymbol{\theta}^{t'}).\end{aligned}$$

For simplicity, we fix $\rho_1=\frac{1}{2}$, $\rho_2=1$, $\beta_d=\frac{(D-d+1)\xi}{\alpha}$, $\alpha=\frac{a}{L}$, and $\gamma\tau^2=\frac{bL}{L_m^2}$, with $a,b>0$. Consequently, we obtain

$$\begin{aligned}B&=\Big[\frac{3\alpha}{2}+(\frac{3L}{2}+3\beta_1+9\gamma\tau^2L_m^2)\alpha^2+3\gamma\tau^2\Big]M\\ &=\Big[\frac{3a}{2L}+(\frac{3a}{2}+3D\xi+9ab)\frac{2a}{L}+\frac{9bL}{ML_m^2}\Big]M,\end{aligned}$$

and

$$c=\min\left\{\frac{[\frac{1}{2}-4(a+2D\xi+6ab)]a}{\kappa},\frac{\frac{1}{2}-(\frac{1}{2}a+D\xi+3ab)+\frac{3bL^2}{aL_m^2M}}{D-d+1}\right\}. \tag{30}$$

Here the proof is complete. □

Theorem 1. *Under the same assumptions and parameters of Lemma 3, the Lyapunov function converges at a linear rate; that is, there exists a constant $\sigma_2\in(0,1)$ such that*

$$\mathbb{V}(\boldsymbol{\theta}^k) \le \sigma_2^k\mathbb{V}(\boldsymbol{\theta}^0), \tag{31}$$

where $\sigma_2=(\sigma_1+BMp^2\frac{1}{\gamma})^{\frac{1}{1+t}}$.

Proof. We first present a critical lemma that will be used to prove our result. □

Lemma 4. *[38, Lemma 3.2] Let $\{\mathbb{V}^k\}$ denote a sequence of nonnegative real numbers satisfying the following inequality for some nonnegative constants p and q.*

$$\mathbb{V}^{k+1} \le p\mathbb{V}^k+q\max_{(k-d(k))_+\le l\le k}\mathbb{V}^k,\quad k\ge 0. \tag{32}$$

If $p+q<1$ and $0\le d(k)\le d_{\max}$ for some positive constant $d_{\max}$, then

$$\mathbb{V}^k \le r^k\mathbb{V}^0,\quad \forall k\ge 1, \tag{33}$$

where $r=(p+q)^{\frac{1}{1+d_{\max}}}$.

Following [38, Lemma 3.2], given that the Lyapunov function obeys (19) and if the following condition is satisfied

$$\sigma_1+BMp^2\frac{1}{\gamma}<1, \tag{34}$$

then it guarantees (31) holds with $\sigma_2=(\sigma_1+BMp^2\frac{1}{\gamma})^{\frac{1}{1+t}}$.

In the sequel, we will show that we can indeed find a set of parameters that make (34) hold. For the design parameter D, we impose $D\le\kappa$. From (20), it is obvious that the following condition

$$\Big[\frac{1}{2}-4(a+2D\xi+6ab)\Big]a \le \frac{1}{2}-\Big(\frac{1}{2}a+D\xi+3ab\Big)+\frac{3bL^2}{aL_m^2M}, \tag{35}$$

guarantees

$$c=\frac{[\frac{1}{2}-4(a+2D\xi+6ab)]a}{\kappa}. \tag{36}$$

Thus, we obtain $\sigma_1=1-c=1-\frac{[\frac{1}{2}-4(a+2D\xi+6ab)]a}{\kappa}$.

It can be verified that choosing $a=\frac{1}{20}$, $b=\frac{1}{10}$, $D\xi=\frac{1}{50}$ and $\tau^2\le\frac{1}{100\kappa}/[M^2p^2(\frac{93L_m^2}{10L^2}+\frac{9}{M})]$ is a sufficient condition for (26), (35) and (34) being satisfied. With above selected parameters, we can obtain $\sigma_1=1-\frac{1}{1000\kappa}$ and

$$\sigma_2=\left(1-\frac{1}{1000\kappa}+M^2p^2\left(\frac{93L_m^2}{100L^2}+\frac{9}{10M}\right)\tau^2\right)^{\frac{1}{1+t}}\in(0,1), \tag{37}$$

which together with (31) indicates the linear convergence of the Lyapunov function and completes the proof.

Algorithm 3. SLAQ

Input: stepsize $\alpha>0$, b, D, $\{\xi_d\}_{d=1}^D$ and $\bar{t}$, and batchsize S.
Initialize: $\boldsymbol{\theta}^k$, and $\{Q_m(\hat{\boldsymbol{\theta}}_m^0),t_m\}_{m\in\mathcal{M}}$.
for $k=1,2,\ldots,K$ **do**
  Server broadcasts $\boldsymbol{\theta}^k$ to all workers.
  for $m=1,2,\ldots,M$ **do**
    Worker m draws S samples and computes the average gradient at these S samples $\nabla f_m(\boldsymbol{\theta}^k)$, along with the quantized gradient $Q_m(\boldsymbol{\theta}^k)$.
    if (9) holds for worker m **then**
      Worker m does not upload anything.
      Set $\hat{\boldsymbol{\theta}}_m^k=\hat{\boldsymbol{\theta}}_m^{k-1}$ and $t_m\leftarrow t_m+1$.
    else
      Worker m uploads δQ_m^k via (6).
      Set $\hat{\boldsymbol{\theta}}_m^k=\boldsymbol{\theta}^k$, and $t_m=0$.
    end if
  end for
  Server updates $\boldsymbol{\theta}$ according to (4).
end for

From (37), it is obvious that if the quantization is accurate enough, i.e., $\tau^2\to 0$, and no communication is skipped, i.e.,

$\bar{t} = 1$, the dependence of convergence rate on condition number is of order $\frac{1}{\kappa}$, the same as the gradient descent.

Compared to the LAG analysis in [30], the analysis for LAQ is more involved, because it needs to deal with not only outdated but also quantized (inexact) gradients. The latter challenges the monotonicity of the Lyapunov function in (18), which is the building block of the analysis in [30]. We tackle this issue by i) considering the outdated gradient in the quantization (6); and, ii) accounting for the quantization error in the new selection criterion (9). As a result, Theorem 1 establishes that LAQ retains the linear convergence rate even when quantization error is present. This is because a controlled quantization error also converges at a linear rate. As for the improvement relative to the conference version [1], the convergence rate σ_2 is explicitly characterized by the quantization parameter τ and the maximum communication skipping interval $\bar{t}$.

Proposition 1. *If under Assumption 1, we choose $\{\xi_d\}_{d=1}^D$ to satisfy $\xi_1 \geq \xi_2 \geq \cdots \geq \xi_D$, and define d_m, $m \in \mathcal{M}$ as*

$$d_m := \max_d \big\{ d | L_m^2 \leq \xi_d/(3\alpha^2 M^2 D),\ d \in \{1, 2, \ldots, D\} \big\}. \tag{38}$$

it suffices for worker m to have at most $k/(d_m + 1)$ uploads with the server until the k-th iteration.

This proposition asserts that the communication intensity per worker is determined by the smoothness of the corresponding local loss function. Workers with smaller smoothness constant communicate with the server less frequently, which justifies the term lazily communication.

For some technical issues, the current bound on the convergence rate is relatively loose in order to account for the worst-case performance. Due to the error ε_m^k introduced by the communication skipping and gradient compression, it is not theoretically established that LAQ outperforms GD. However, our empirical studies will demonstrate that LAQ significantly outperforms GD in terms of communication. To prove that LAQ is more communication-efficient than GD is more challenging and is in our future agenda.

4 Generalizing LAQ

In this section, we broaden the scope of LAQ by developing the stochastic LAQ and the two-way communication-efficient LAQ-driven federated learning, as we elaborate next.

4.1 Stochastic LAQ

Thanks to its well-documented merits in reducing complexity, stochastic gradient descent has been widely employed by learning tasks involving large-scale training data. Here we show that LAQ can also benefit from its stochastic counterpart, namely SLAQ, that is developed with a simple modification in the criterion. Specifically, (9a) in SLAQ is replaced by

$$\begin{aligned} \|Q_m^s(\hat{\boldsymbol{\theta}}_m^{k-1}) - Q_m^s(\boldsymbol{\theta}^k)\|_2^2 \leq \frac{1}{\alpha^2 M^2} \sum_{d=1}^D \xi_d \|\boldsymbol{\theta}^{k+1-d} - \boldsymbol{\theta}^{k-d}\|_2^2 \\ + 3\Big(\|\boldsymbol{\varepsilon}_m^k\|_2^2 + \|\hat{\boldsymbol{\varepsilon}}_m^{k-1}\|_2^2 \Big) + var, \end{aligned} \tag{39}$$

where superscript s will henceforth denote the stochastic counterpart of LAQ quantities defined so far; and var is a constant. Compared with (9a), the constant added in the stochastic case is to compensate for the variance coming from the stochastic sampling. In practice, we can use the empirical variance to approximate the variance, that is, the variance computed according to the drawn samples per iteration.

Algorithm 4. TWO-LAQ

Input: stepsize $\alpha > 0$, b, D, $\{\xi_d\}_{d=1}^D$ and $\bar{t}$.
Initialize: $\boldsymbol{\theta}^1$, $\tilde{\boldsymbol{\theta}}^0 = \boldsymbol{\theta}^0$, and $\{Q_m(\hat{\boldsymbol{\theta}}_m^0), t_m\}_{m \in \mathcal{M}}$.
for $k = 1, 2, \ldots, K$ **do**
  Server calculate $\tilde{\boldsymbol{\theta}}^k$ broadcasts quantized model innovation $\delta\boldsymbol{\theta}^k$ to all workers.
  for $m = 1, 2, \ldots, M$ **do**
    Worker m computes $\tilde{\boldsymbol{\theta}}^k$ according to (41) and $\nabla f_m(\tilde{\boldsymbol{\theta}}^k)$ and $Q_m(\tilde{\boldsymbol{\theta}}^k)$.
    if (9) holds for worker m **then**
      Worker m does not upload anything.
      Set $\hat{\boldsymbol{\theta}}_m^k = \hat{\boldsymbol{\theta}}_m^{k-1}$ and $t_m \leftarrow t_m + 1$.
    else
      Worker m uploads δQ_m^k via (6).
      Set $\hat{\boldsymbol{\theta}}_m^k = \tilde{\boldsymbol{\theta}}^k$, and $t_m = 0$.
    end if
  end for
  Server updates $\boldsymbol{\theta}^{k+1}$ according to (4).
end for

Apart from the criterion, SLAQ is different from LAQ only in the local (stochastic) gradient calculation. Specifically, the worker randomly draws S samples from its training set and computes the stochastic gradient as

$$\nabla f_m^s(\theta) = \frac{1}{S} \sum_{n=1}^S \nabla_{\boldsymbol{\theta}} \ell(\mathbf{x}_{m,n}; \boldsymbol{\theta}). \tag{40}$$

The quantization and other operations are the same as before. The SLAQ is summarized in Algorithm 3. Following the convention, we also consider here that the global loss function is scaled by the total number of training samples.

4.2 Two-Way Quantization

So far, communication savings have been achieved by skipping uploads and quantizing the uploaded gradient innovation. A natural extension is to also quantize the model innovation in the downlink, which results in what we term **T**wo **W**ay **L**azily **A**ggregated **G**radient (TWO-LAQ).

Let $\tilde{\boldsymbol{\theta}}^k = \mathcal{Q}(\boldsymbol{\theta}^k, \tilde{\boldsymbol{\theta}}^{k-1})$ describe the quantization model. First, the model innovation is quantized as (5), and thus we omit the details. Then, for the server to workers communication, only the quantized model innovation $\delta\boldsymbol{\theta}^k$ is broadcast. With $\tilde{\boldsymbol{\theta}}^{k-1}$ stored in memory, both the server and the workers update $\tilde{\boldsymbol{\theta}}^k$ as

$$\tilde{\boldsymbol{\theta}}^k = \tilde{\boldsymbol{\theta}}^{k-1} + \delta\boldsymbol{\theta}^k. \tag{41}$$

Different from possible alternatives, each worker m does not have the accurate model $\boldsymbol{\theta}^k$. As a result, each worker has to compute the local gradient $\nabla f_m(\tilde{\boldsymbol{\theta}}^k)$ based on $\tilde{\boldsymbol{\theta}}^k$. The rest follows the LAG algorithm, meaning workers quantize the gradient innovation, and upload it, if it is large enough; otherwise, they skip this upload round.

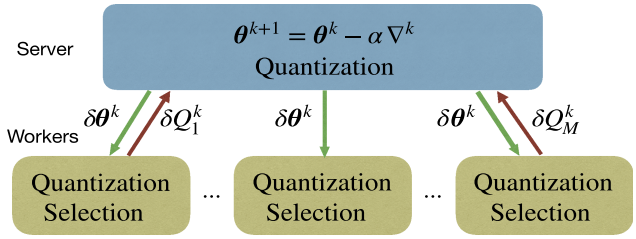


Fig. 3. Federated learning via TWO-LAQ.

The steps of TWO-LAQ are summarized in Algorithm 4, whose implementation is illustrated in Fig. 3. Comparing Fig. 3 with gradient descent, TWO-LAQ improves communication efficiency at the expense of extra memory at the server and workers. Indeed, the server needs to store θ^k, $\tilde{\theta}^k$ and ∇^k; and each worker m needs to store $\hat{\theta}^k$, $\tilde{\theta}^k$ and $Q_m(\tilde{\theta}^k)$. In contrast, GD only requires the server to store θ^k.

5 Numerical Tests

To validate our theoretical analysis and demonstrate the performance of LAQ in improving communication efficiency for practical machine learning tasks, we evaluate our algorithm on the regularized logistic regression (LR), which is strongly convex, and a neural network (NN) classifier involving a nonconvex loss. For our experiments, we use the MNIST dataset [39], which is equally distributed across $M = 10$ workers. Throughout, we set $D = 10$, $\xi_1 = \cdots = \xi_D = 0.8/D$, and $\bar{t} = 100$.

5.1 Simulation Setup

LR Classifier. Consider a multi-class classifier with say $C = 10$ classes, that relies on logistic regression trained using the MNIST dataset. Each training vector $\mathbf{x}_{m,n}$ comprises a feature-label pair $(\mathbf{x}^f_{m,n}, \mathbf{x}^l_{m,n})$, where $\mathbf{x}^f_{m,n} \in \mathbb{R}^F$ is the feature vector and $\mathbf{x}^l_{m,n} \in \mathbb{R}^C$ denotes the one-hot label vector. The model $\theta \in R^{C\times F}$ here is a matrix, which is slightly different from previous description, and it is adopted for convenience but does not change the learning problem. The estimated probability of (m,n)-th sample to belong to class i is given by

$$\hat{\mathbf{x}}^l_{m,n} = \text{softmax}(\boldsymbol{\theta}\mathbf{x}^f_{m,n}), \tag{42}$$

which can be explicitly written as

$$[\hat{\mathbf{x}}^l_{m,n}]_i = \frac{e^{[\boldsymbol{\theta}\mathbf{x}^f_{m,n}]_i}}{\sum_{j=1}^C e^{[\boldsymbol{\theta}\mathbf{x}^f_{m,n}]_j}}, \ \forall i \in \{1,2,\ldots,C\}. \tag{43}$$

The regularized logistic regression classifier relies on the following cross-entropy loss plus a regularizer

$$\ell(\mathbf{x}_{m,n}, \boldsymbol{\theta}) = -\sum_{i=1}^C [\mathbf{x}^l_{m,n}]_i \log [\hat{\mathbf{x}}^l_{m,n}]_i + \frac{\lambda}{2}\text{Tr}(\boldsymbol{\theta}^T\boldsymbol{\theta}), \tag{44}$$

where $\text{Tr}(\cdot)$ denotes trace operator, and $\boldsymbol{\theta}^T$ is the transpose of $\boldsymbol{\theta}$. Having defined $\ell(\mathbf{x}_{m,n}, \boldsymbol{\theta})$, the local loss functions are $f_m(\boldsymbol{\theta}) = \sum_{n=1}^{N_m} \ell(\mathbf{x}_{m,n}; \boldsymbol{\theta})$, and the global loss function is given by

$$f(\boldsymbol{\theta}) = \frac{1}{N}\sum_{m\in\mathcal{M}} f_m(\boldsymbol{\theta}), \tag{45}$$

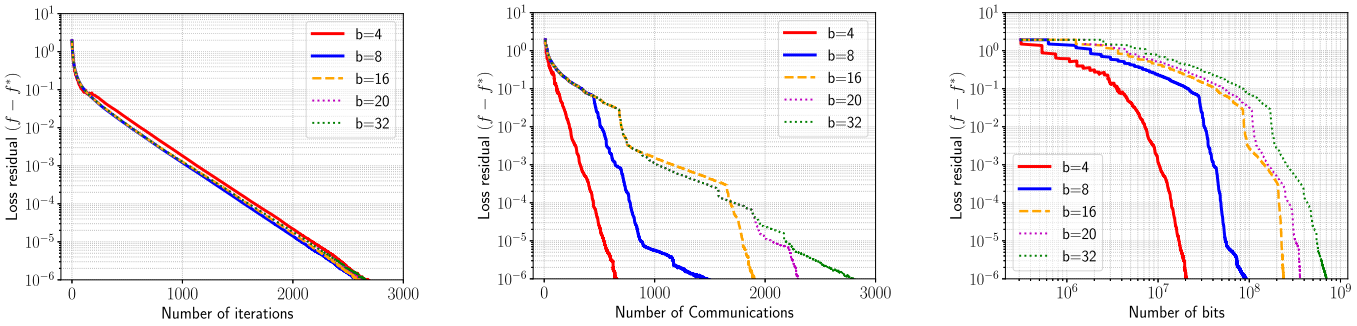
where N is the total number of data samples. In our tests, we set the regularizer coefficient to $\lambda = 0.01$.

NN Classifier. In our tests, we employ a ReLU network comprising one hidden layer having 200 nodes with dimensions of the input and output layers being $784(28 \times 28)$ and 10, respectively. The regularizer parameter is set to $\lambda = 0.01$. *CNN classifier*. For the test in CIFAR 10 dataset, we adopt the convolutional neural network (CNN) which consists of 3 VGG-type blocks [40]. Each block is constructed by stacking two convolutional layers with small 3×3 filters followed by a max pooling layer. The numbers of filters for the convolutional layers in the three blocks are 32, 64, and 128, respectively. ReLU activation function is used in each layer and padding is utilized on the convolutional layers to ensure the height and width of the output feature matches the inputs. Additionally, each block is followed by a dropout layer with the rate of 20 percent. This is followed by a fully connected layer with 128 nodes and then the softmax layer. We add an l_2 regularization with coefficient 0.001.

5.2 Numerical Tests

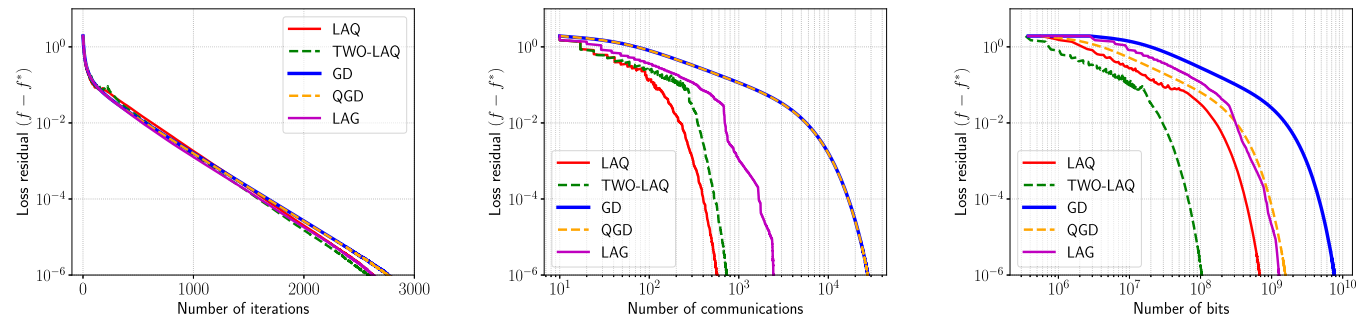
Fig. 4 illustrates the convergence with different number of quantization bits. It shows that utilizing fewer bits to quantize the gradient moderately increases the number of iterations, but markedly reduces the overall number of transmitted bits. To benchmark LAQ, we compare it with two classes of algorithms, namely GD and minibatch SGD ones, corresponding to the following two tests.

Parameters. For GD algorithms, we fix $D = 10$, $\xi_1 = \xi_2 = \cdots = \xi_D = 0.8/D$, $\bar{t} = 100$, and we set $\alpha = 0.02$, and $b = 4$ or 8 for LR and NN classifiers, respectively. For minibatch SGD algorithms, the minibath size is 500 and $\alpha = 0.008$; $b = 3$ for LR and $b = 8$ for NN.



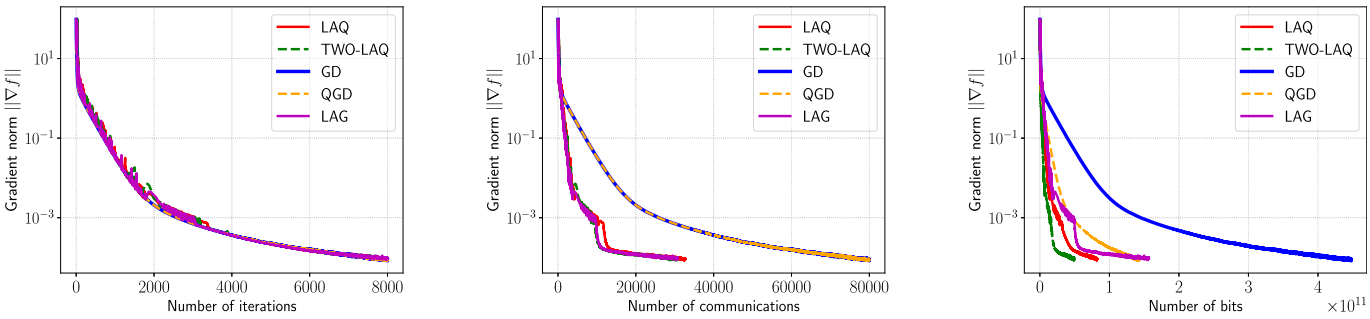
(a) Loss v.s. iteration (b) Loss v.s. communication (c) Loss v.s. bit

Fig. 4. Convergence of LAQ under different quantization bits (logistic regression).



(a) Loss v.s. iteration (b) Loss v.s. communication (c) Loss v.s. bit

Fig. 5. Convergence of the loss function (logistic regression).



(a) Gradient v.s. iteration (b) Gradient v.s. communication (c) Gradient v.s. bit

Fig. 6. Convergence of gradient norm (neural network).

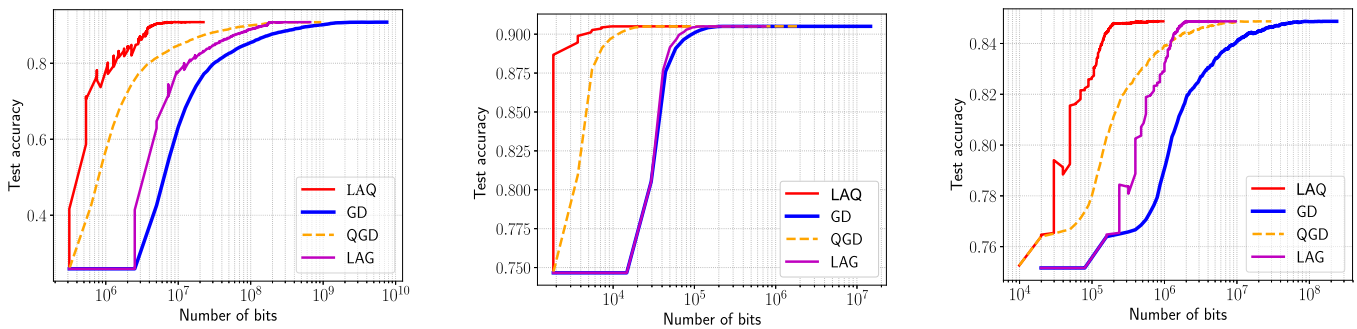
TABLE 1
Comparison of Gradient-Based Algorithms

Algorithm		Iteration #	Communication #	Bit #	Accuracy
Logistic	**LAQ**	**2626**	**572**	$\mathbf{6.78 \times 10^8}$	**0.9082**
	TWO-LAQ	**2576**	**734**	$\mathbf{1.04 \times 10^8}$	**0.9082**
	GD	2763	27630	7.63×10^9	0.9082
	QGD	2760	27600	1.56×10^9	0.9082
	LAG	2620	2431	1.27×10^9	0.9082
Neural network	**LAQ**	**8000**	**32729**	$\mathbf{8.23 \times 10^{10}}$	**0.9433**
	TWO-LAQ	**8000**	**30741**	$\mathbf{4.93 \times 10^{10}}$	**0.9433**
	GD	8000	80000	4.48×10^{11}	0.9433
	QGD	8000	80000	1.42×10^{11}	0.9433
	LAG	8000	30818	1.98×10^{11}	0.9433

For logistic regression, all algorithms terminate when loss residual reaches 10^{-6}; for neural network, all algorithms run a fixed number of iterations.

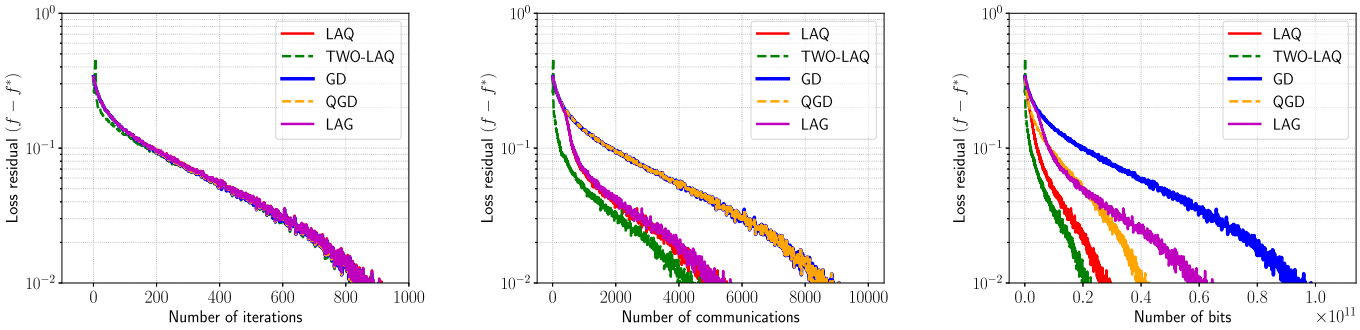
Gradient-Based Tests. The benchmark algorithms include GD, QGD [11] and lazily aggregated gradient (LAG) [30]. Fig. 5 shows the convergence of loss residual for the LR problem. Clearly, Fig. 5a corroborates Theorem 1, namely the linear convergence for the strongly convex loss function. As illustrated in Fig. 5b, LAQ incurs a smaller number of communication rounds than GD and QGD thanks to our innovation selection rule, yet more rounds than LAG due to the quantization error. Nevertheless, the total number of transmitted bits of LAQ is significantly reduced compared with that of LAG, as demonstrated in Fig. 5c. For the NN classifier, Fig. 6 reports the convergence of the gradient norm, where LAQ also shows competitive performance for nonconvex loss functions. Similar to what is observed for LR classification, LAQ outperforms the benchmark algorithms in terms of transtimitted bits. TWO-LAQ which additionally leverages model innovation quantization saves more bits than LAQ as show in Figs. 5c and 6c. Table 1 summarizes the detailed comparison of mentioned algorithms including the number of iterations, uploads and bits needed to reach a given accuracy.

Tests on More Datasets. Fig. 7 exhibits the test accuracy of the aforementioned algorithms on three commonly used datasets, namely MNIST, ijcnn1 and covtype. Applied to all these datasets, LAQ saves transmitted bits while maintaining the same accuracy. In addition, we test our algorithm on more challenging dataset—CIFAR 10, for which CNN is utilized and Adam [41] is applied. The convergence of the loss function is plotted in Fig. 8, and the validation accuracy is shown in Fig. 11. A detailed comparison with above mentioned benchmark algorithms is summarized in Table 2. These tests with different datasets and different algorithms (gradient descent, Adam and the following stochastic gradient descent) all demonstrate that the proposed communication saving scheme indeed



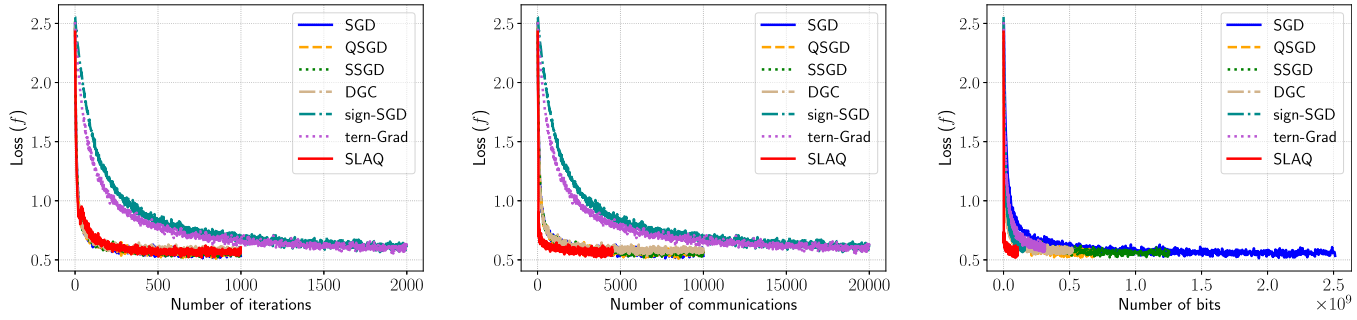
(a) MNIST (b) ijcnn1 (c) covtype

Fig. 7. Tests on different datasets.



(a) Loss v.s. iteration (b) Loss v.s. communication (c) Loss v.s. bit

Fig. 8. Convergence of loss function (CNN for CIFAR 10).



(a) Loss v.s. iteration (b) Loss v.s. communication (c) Loss v.s. bit

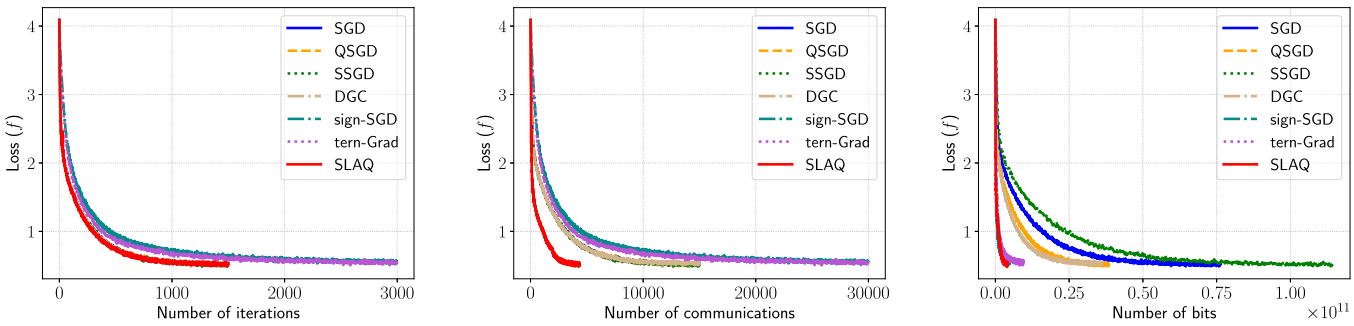
Fig. 9. Convergence of loss function (logistic regression).

provides satisfied improvement in communication efficiency and thus has promising potential for variety of distributed learning applications.

Stochastic Gradient-Based Tests. The stochastic version of LAQ abbreviated as SLAQ is tested and compared with stochastic gradient descent, quantized stochastic gradient descent (QSGD) [10], sparsified stochastic gradient descent (SSGD) [21], deep gradient compression (DGC) [18], sign-SGD, [8] and tern-Grad [12]. For the all the stochastic-based algorithms, each worker draws a bath of 500 data samples to calculate a stochastic local gradient per iteration. As demonstrated in Figs. 9 and 10, SLAQ requires the lowest number of communication rounds and bits. Albeit sign-SGD and tern-Grad need only 1 bit and 2 bits for each entry of the gradient, respectively, they have larger quantization error and require a smaller stepsize to ensure convergence. Therefore it takes more iterations for these two algorithms to reach the same loss (or accuracy), and needs to transmit more bits than that for SLAQ. In this stochastic gradient test, although the improvement of communication efficiency by SLAQ is not as evident as LAQ compared with GD-based algorithms, SLAQ still outperforms the cutting-edge schemes, namely QSGD, SSGD, sign-SGD and tern-Grad. The results are summarized in Table 3.

TABLE 2
Tests for CIFAR 10 With CNN

Algorithm	Iteration #	Communication #	Bit #	Accuracy
LAQ	1000	6359	3.34×10^{10}	87.96
TWO-LAQ	1000	5785	2.76×10^{10}	87.92
GD	1000	10000	1.09×10^{11}	87.86
QGD	1000	10000	4.69×10^{10}	87.95
LAG	1000	6542	7.44×10^{10}	87.91



(a) Loss v.s. iteration (b) Loss v.s. communication (c) Loss v.s. bit

Fig. 10. Convergence of loss function (neural network).

TABLE 3
Performance Comparison of Mini-Batch Stochastic Gradient-Based Algorithms

Algorithm		Iteration #	Communication #	Bit #	Accuracy
Logistic	**SLAQ**	**1000**	**4494**	$\mathbf{1.06 \times 10^8}$	**0.9060**
	SGD	1000	10000	2.51×10^9	0.9044
	QSGD	1000	10000	6.89×10^8	0.9062
	SSGD	1000	10000	1.26×10^8	0.9056
	DGC	1000	10000	5.21×10^9	0.9082
	sign-SGD	2000	20000	1.57×10^8	0.8905
	tern-Grad	2000	20000	3.14×10^8	0.8942
Neural network	**SLAQ**	**1500**	**4342**	$\mathbf{4.14 \times 10^9}$	**0.9360**
	SGD	1500	15000	7.63×10^{10}	0.9354
	QSGD	1500	15000	3.84×10^{10}	0.9353
	SSGD	1500	15000	1.14×10^{11}	0.9356
	DGC	1500	15000	3.60×10^{10}	0.9362
	sign-SGD	3000	30000	4.77×10^9	0.9277
	tern-Grad	3000	30000	9.54×10^9	0.9299

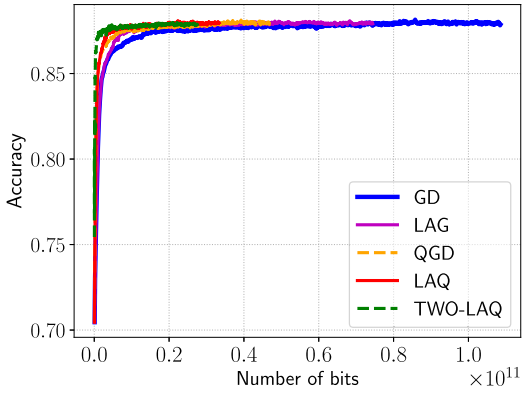


Fig. 11. Accuracy versus bit for CIFAR 10.

6 Conclusion

This paper investigated communication-efficient federated learning, and developed LAQ—an approach that integrates quantization and adaptive communication techniques based on gradient innovation. Compared with GD method, LAQ introduces errors to gradient, yet still preserves linear convergence for strongly convex problems. This is a remarkable result considering that LAQ significantly reduces both communication bits and rounds. Experiments on strongly convex and nonconvex learning problems verified our theoretical analysis and demonstrated the merits of LAQ over recent popular approaches. Furthermore, two variants of LAQ, termed TWO-LAQ and SLAQ, also exhibit promising performance and outperform prevalent compression schemes in the empirical studies.

7 Proofs

7.1 Proof of Lemma 2

For successive LAQ updates, it is not difficult to show that

$$
\begin{aligned}
&f(\boldsymbol{\theta}^{k+1}) - f(\boldsymbol{\theta}^k) \\
&\le \left\langle \nabla f(\boldsymbol{\theta}^k), -\alpha[\nabla f(\boldsymbol{\theta}^k) - \boldsymbol{\varepsilon}^k + \sum_{m\in\mathcal{M}_c^k} (Q_m(\hat{\boldsymbol{\theta}}_m^{k-1}) - Q_m(\boldsymbol{\theta}^k))] \right\rangle \\
&\quad + \frac{L}{2}\|\boldsymbol{\theta}^{k+1} - \boldsymbol{\theta}^k\|_2^2 \\
&\le -\alpha\|\nabla f(\boldsymbol{\theta}^k)\|_2^2 + \frac{L}{2}\|\boldsymbol{\theta}^{k+1} - \boldsymbol{\theta}^k\|_2^2 \\
&\quad + \alpha\left\langle \nabla f(\boldsymbol{\theta}^k), \boldsymbol{\varepsilon}^k - \sum_{m\in\mathcal{M}_c^k} (Q_m(\hat{\boldsymbol{\theta}}_m^{k-1}) - Q_m(\boldsymbol{\theta}^k)) \right\rangle \\
&= -\alpha\|\nabla f(\boldsymbol{\theta}^k)\|_2^2 - \frac{\|\boldsymbol{\theta}^{k+1} - \boldsymbol{\theta}^k\|_2^2}{2\alpha} + \frac{L}{2}\|\boldsymbol{\theta}^{k+1} - \boldsymbol{\theta}^k\|_2^2 \\
&\quad + \frac{\alpha}{2}[\|\nabla f(\boldsymbol{\theta}^k)\|_2^2 + \|\boldsymbol{\varepsilon}^k - \sum_{m\in\mathcal{M}_c^k} (Q_m(\hat{\boldsymbol{\theta}}_m^{k-1}) - Q_m(\boldsymbol{\theta}^k))\|_2^2] \\
&\le -\frac{\alpha}{2}\|\nabla f(\boldsymbol{\theta}^k)\|_2^2 + \alpha\|\sum_{m\in\mathcal{M}_c^k} (Q_m(\hat{\boldsymbol{\theta}}_m^{k-1}) - Q_m(\boldsymbol{\theta}^k))\|_2^2 \\
&\quad + \left(\frac{L}{2} - \frac{1}{2\alpha}\right)\|\boldsymbol{\theta}^{k+1} - \boldsymbol{\theta}^k\|_2^2 + \alpha\|\boldsymbol{\varepsilon}^k\|_2^2,
\end{aligned}
$$

where the second equality follows from the identity $\langle \boldsymbol{a}, \boldsymbol{b}\rangle = \frac{1}{2}(\|\boldsymbol{a}\|^2 + \|\boldsymbol{b}\|^2 - \|\boldsymbol{a}-\boldsymbol{b}\|^2)$, and the last inequality from the fact that $\|\sum_{i=1}^n \boldsymbol{a}_i\|_2^2 \leq n\sum_{i=1}^n \|\boldsymbol{a}_i\|^2$.

7.2 Proof of Proposition 1

Suppose that at current iteration k the last iteration when worker m communicated with the server is d', where $1 \leq d' \leq d_m$. Having $\boldsymbol{\theta}_m^{k-1} = \boldsymbol{\theta}^{k-d'}$, we thus deduce that

$$\begin{aligned}
&\|Q_m(\hat{\boldsymbol{\theta}}_m^{k-1}) - Q_m(\boldsymbol{\theta}^k)\|_2^2\\
=&\|Q_m(\boldsymbol{\theta}^{k-d'}) - \nabla f_m(\boldsymbol{\theta}^{k-d'}) - Q_m(\boldsymbol{\theta}^k) + \nabla f_m(\boldsymbol{\theta}^k)\\
&+ \nabla f_m(\boldsymbol{\theta}^{k-d'}) - \nabla f_m(\boldsymbol{\theta}^k)\|_2^2\\
\leq&3(\|f_m(\boldsymbol{\theta}^{k-d'}) - \nabla f_m(\boldsymbol{\theta}^k)\|_2^2 + \|\boldsymbol{\varepsilon}_m^k\|_2^2 + \|\boldsymbol{\varepsilon}_m^{k-d'}\|_2^2)\\
\leq&3L_m^2\|\boldsymbol{\theta}^{k-d'} - \boldsymbol{\theta}^k\|_2^2 + 3(\|\boldsymbol{\varepsilon}_m^k\|_2^2 + \|\boldsymbol{\varepsilon}_m^{k-d'}\|_2^2) \qquad (46)\\
=&3L_m^2\|\sum_{d=1}^{d'}\boldsymbol{\theta}^{k+1-d} - \boldsymbol{\theta}^{k-d}\|_2^2 + 3(\|\boldsymbol{\varepsilon}_m^k\|_2^2 + \|\boldsymbol{\varepsilon}_m^{k-d'}\|_2^2)\\
\leq&3L_m^2 d'\sum_{t=1}^{d'}\|\boldsymbol{\theta}^{k+1-d} - \boldsymbol{\theta}^{k-d}\|_2^2 + 3(\|\boldsymbol{\varepsilon}_m^k\|_2^2 + \|\boldsymbol{\varepsilon}_m^{k-d'}\|_2^2).
\end{aligned}$$

From the definition of d_m and since $\xi_1 \geq \xi_2 \geq \cdots \geq \xi_D$, it can be inferred that

$$L_m^2 \leq \frac{\xi_{d'}}{3\alpha^2 M^2 D},\ \textit{for all } d' \textit{ satisfying } 1 \leq d' \leq d_m. \qquad (47)$$

Substituting (47) into (46) yields

$$\begin{aligned}
&\|Q_m(\hat{\boldsymbol{\theta}}_m^{k-1}) - Q_m(\boldsymbol{\theta}^k)\|_2^2\\
&\leq \frac{\xi_{d'}}{\alpha^2 M^2}\sum_{d=1}^{d'}\xi_d\|\boldsymbol{\theta}^{k+1-d} - \boldsymbol{\theta}^{k-d}\|_2^2 + 3(\|\boldsymbol{\varepsilon}_m^k\|_2^2 + \|\hat{\boldsymbol{\varepsilon}}_m^{k-1}\|_2^2) \qquad (48)\\
&\leq \frac{1}{\alpha^2 M^2}\sum_{d=1}^{D}\xi_d\|\boldsymbol{\theta}^{k+1-d} - \boldsymbol{\theta}^{k-d}\|_2^2 + 3(\|\boldsymbol{\varepsilon}_m^k\|_2^2 + \|\hat{\boldsymbol{\varepsilon}}_m^{k-1}\|_2^2),
\end{aligned}$$

which exactly implies that (9a) is satisfied. Since $d_m \leq D \leq \bar{t}$, the criterion (9) holds, which means that worker m will not upload her/his information until at least t_m iterations after last upload. In the first k iterations, worker m will therefore have at most $k/(d_m+1)$ uploads to the server.

ACKNOWLEDGMENTS

The work of Jun Sun and Zaiyue Yang was supported in part by the NSFC Grant 61873118, in part by the Shenzhen Committee on Science and Innovations under Grant GJHZ20180411143603361, and in part by the Department of Science and Technology of Guangdong Province under Grant 2018A050506003. The work by Jun Sun was also supported by China Scholarship Council. The work of Qinmin Yang was supported in part by the Key-Area Research and Development Program of Guangdong Province (No. 2018B010107002), and in part by the National Natural Science Foundation of China (61751205). The work of Georgios Giannakis is supported partially by NSF 1901134. Part of the results in this paper has been presented in Proc. of Neural Information Processing, Vancouver, Canada, December 8-14, 2019 [1].

REFERENCES

[1] J. Sun, T. Chen, G. Giannakis, and Z. Yang, "Communication-efficient distributed learning via lazily aggregated quantized gradients," in *Proc. 33rd Int. Conf. Neural Inf. Process. Syst.*, 2019, pp. 3370–3380.

[2] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. Y. Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proc. 20th Int. Conf. Artif. Intell. Statist.*, 2017, pp. 1273–1282.

[3] J. Konečnỳ, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, "Federated learning: Strategies for improving communication efficiency," 2016, *arXiv preprint:1610.05492*.

[4] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. Y. Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proc. 20th Int. Conf. Artif. Intell. Statist.*, 2017, pp. 1273–1282.

[5] M. Li, D. G. Andersen, A. J. Smola, and K. Yu, "Communication efficient distributed machine learning with the parameter server," in *Proc. 27th Int. Conf. Neural Inf. Process. Syst.*, 2014, pp. 19–27.

[6] S. P. Karimireddy, Q. Rebjock, S. Stich, and M. Jaggi, "Error feedback fixes signSGD and other gradient compression schemes," in *Proc. 36th Int. Conf. Mach. Learn.*, 2019, pp. 3252–3261.

[7] E. J. Msechu and G. B. Giannakis, "Sensor-centric data reduction for estimation with WSNs via censoring and quantization," *IEEE Trans. Signal Process.*, vol. 60, no. 1, pp. 400–414, Jan. 2012.

[8] F. Seide, H. Fu, J. Droppo, G. Li, and D. Yu, "1-bit stochastic gradient descent and its application to data-parallel distributed training of speech DNNs," in *Proc. Conf. Int. Speech Commun. Assoc.*, 2014, pp. 1058–1062.

[9] J. Bernstein, Y.-X. Wang, K. Azizzadenesheli, and A. Anandkumar, "SignSGD: Compressed optimisation for non-convex problems," in *Proc. 35th Int. Conf. Mach. Learn.*, 2018, pp. 559–568.

[10] D. Alistarh, D. Grubic, J. Li, R. Tomioka, and M. Vojnovic, "QSGD: Communication-efficient SGD via gradient quantization and encoding," in *Proc. 31st Int. Conf. Neural Inf. Process. Syst.*, 2017, pp. 1709–1720.

[11] S. Magnússon, H. Shokri-Ghadikolaei, and N. Li, "On maintaining linear convergence of distributed learning and optimization under limited communication," 2019, *arXiv: 1902.11163*.

[12] W. Wen *et al.*, "TernGrad: Ternary gradients to reduce communication in distributed deep learning," in *Proc. 31st Int. Conf. Neural Inf. Process. Syst.*, 2017, pp. 1509–1519.

[13] H. Zhang, J. Li, K. Kara, D. Alistarh, J. Liu, and C. Zhang, "ZipML: Training linear models with end-to-end low precision, and a little bit of deep learning," in *Proc. 34th Int. Conf. Mach. Learn.*, 2017, pp. 4035–4043.

[14] J. Wu, W. Huang, J. Huang, and T. Zhang, "Error compensated quantized SGD and its applications to large-scale distributed optimization," 2018, *arXiv: 1806.08054*.

[15] K. Mishchenko, E. Gorbunov, M. Takáč, and P. Richtárik, "Distributed learning with compressed gradient differences," Jan. 2019, *arXiv: 1901.09269*.

[16] N. Strom, "Scalable distributed DNN training using commodity gpu cloud computing," in *Proc. Conf. Int. Speech Commun. Assoc.*, 2015, pp. 1488–1492.

[17] A. F. Aji and K. Heafield, "Sparse communication for distributed gradient descent," in *Proc. Conf. Empir. Methods Natural Lang. Process.*, 2017, pp. 440–445.

[18] Y. Lin, S. Han, H. Mao, Y. Wang, and W. J. Dally, "Deep gradient compression: Reducing the communication bandwidth for distributed training," in *Proc. Int. Conf. Learn. Representations*, 2018, pp. 1–12.

[19] S. U. Stich, J.-B. Cordonnier, and M. Jaggi, "Sparsified SGD with memory," in *Proc. 32nd Int. Conf. Neural Inf. Process. Syst.*, 2018, pp. 4447–4458.

[20] D. Alistarh, T. Hoefler, M. Johansson, N. Konstantinov, S. Khirirat, and C. Renggli, "The convergence of sparsified gradient methods," in *Proc. 32nd Int. Conf. Neural Inf. Process. Syst.*, 2018, pp. 5973–5983.

[21] J. Wangni, J. Wang, J. Liu, and T. Zhang, "Gradient sparsification for communication-efficient distributed optimization," in *Proc. 32nd Int. Conf. Neural Inf. Process. Syst.*, 2018, pp. 1299–1309.

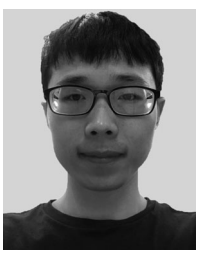
[22] H. Wang, S. Sievert, S. Liu, Z. Charles, D. Papailiopoulos, and S. Wright, "ATOMO: Communication-efficient learning via atomic sparsification," in *Proc. 32nd Int. Conf. Neural Inf. Process. Syst.*, 2018, pp. 9850–9861.

[23] P. Jiang and G. Agrawal, "A linear speedup analysis of distributed deep learning with sparse and quantized communication," in *Proc. 32nd Int. Conf. Neural Inf. Process. Syst.*, 2018, pp. 2525–2536.

[24] J. Konečnỳ and P. Richtárik, "Randomized distributed mean estimation: Accuracy vs communication," *Front. Appl. Math. Statist.*, vol. 4, Dec. 2018, Art. no. 62.

[25] L. L. Peterson and B. S. Davie, *Computer Networks: A Systems Approach*. Burlington, MA, USA: Morgan Kaufman, 2007.

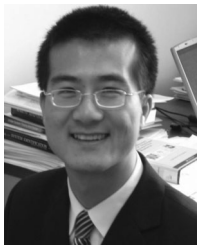
[26] O. Shamir, N. Srebro, and T. Zhang, "Communication-efficient distributed optimization using an approximate newton-type method," in *Proc. 31st Int. Conf. Mach. Learn.*, 2014, pp. 1000–1008.
[27] Y. Zhang and X. Lin, "DiSCO: Distributed optimization for self-concordant empirical loss," in *Proc. 32nd Int. Conf. Mach. Learn.*, 2015, pp. 362–370.
[28] S. Zhang, A. E. Choromanska, and Y. LeCun, "Deep learning with elastic averaging SGD," in *Proc. 28th Int. Conf. Neural Inf. Process. Syst.*, 2015, pp. 685–693.
[29] H. Yu and R. Jin, "On the computation and communication complexity of parallel SGD with dynamic batch sizes for stochastic non-convex optimization," 2019, *arXiv: 1905.04346*.
[30] T. Chen, G. Giannakis, T. Sun, and W. Yin, "LAG: Lazily aggregated gradient for communication-efficient distributed learning," in *Proc. 32nd Int. Conf. Neural Inf. Process. Syst.*, 2018, pp. 5050–5060.
[31] T. Chen, K. Zhang, G. B. Giannakis, and T. Basar, "Communication-efficient distributed reinforcement learning," 2018, *arXiv: 1812.03239*.
[32] J. Wang and G. Joshi, "Cooperative SGD: A unified framework for the design and analysis of communication-efficient SGD algorithms," 2018, *arXiv: 1808.07576*.
[33] Y. Arjevani and O. Shamir, "Communication complexity of distributed convex learning and optimization," in *Proc. 28th Int. Conf. Neural Inf. Process. Syst.*, 2015, pp. 1756–1764.
[34] S. Zheng *et al.*, "Asynchronous stochastic gradient descent with delay compensation," in *Proc. 34th Int. Conf. Mach. Learn.*, 2017, pp. 4120–4129.
[35] M. I. Jordan, J. D. Lee, and Y. Yang, "Communication-efficient distributed statistical inference," *J. Amer. Statist. Assoc.*, vol. 114, pp. 668–681, 2018.
[36] L. Zhu, Z. Liu, and S. Han, "Deep leakage from gradients," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2019, pp. 14 774–14 784.
[37] D. Yu, H. Zhang, W. Chen, T.-Y. Liu, and J. Yin, "Gradient perturbation is underrated for differentially private convex optimization," 2019, *arXiv: 1911.11363*.
[38] M. Gurbuzbalaban, A. Ozdaglar, and P. A. Parrilo, "On the convergence rate of incremental aggregated gradient algorithms," *SIAM J. Optim.*, vol. 27, no. 2, pp. 1035–1048, 2017.
[39] Y. LeCun, C. Cortes, and C. Burges, "MNIST handwritten digit database," *AT&T Labs*, 2010. [Online]. Available: http://yann. lecun. com/exdb/mnist
[40] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014, *arXiv:1409.1556*.
[41] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*.



Jun Sun (Student Member, IEEE) received the BS degree from the College of Astronautics from the Nanjing University of Aeronautics and Astronautics, China, in 2015. Currently, he is currently working toward the PhD degree from the College of Control Science and Engineering, Zhejiang University, China. He is a member of the Group of Networked Sensing and Control in the State Key Laboratory of Industrial Control Technology at Zhejiang University, China. His research interests include game theory and optimization with applications in electricity market, data centers, and machine learning.

Tianyi Chen (Member, IEEE) received the BEng degree in communication science and engineering from Fudan University, China, in 2014, the MSc and PhD degrees in electrical and computer engineering (ECE) from the University of Minnesota (UMN), Minneapolis, Minnesota, in 2016 and 2019, respectively. Since August 2019, he has been with the Department of Electrical, Computer, and Systems Engineering, Rensselaer Polytechnic Institute as an assistant professor. During 2017-2018, he has been a visiting scholar at Harvard University, Cambridge, Massachusetts, University of California, Los Angeles, Los Angeles, California, and University of Illinois Urbana-Champaign, Champaign, Illinois. He was a Best Student Paper Award finalist, in the 2017 Asilomar Conf. on Signals, Systems, and Computers. He received the National Scholarship from China, in 2013, the UMN ECE Department Fellowship, in 2014, and the UMN Doctoral Dissertation Fellowship, in 2017. His research interests include optimization and statistical signal processing with applications to distributed learning, reinforcement learning and wireless networks.

Georgios B. Giannakis (Fellow, IEEE) received the diploma in electrical engineering from the National Technical University of Athens, Greece, 1981, and the MSc degree in electrical engineering, 1983, the MSc degree in mathematics, 1986, and the PhD degree in electrical engineering, 1986 from the University of Southern California, Los Angeles, California. From 1982 to 1986 he was with the University of Southern California (USC), Los Angeles, California. He was a faculty member with the University of Virginia, Charlottesville, Virginia from 1987 to 1998, and since 1999 he has been a professor with the University of Minnesota, Minneapolis, Minnesota, where he holds an ADC Endowed chair, a University of Minnesota McKnight Presidential chair in ECE, and serves as director of the Digital Technology Center. His general interests span the areas of statistical learning, communications, and networking - subjects on which he has published more than 470 journal papers, 770 conference papers, 26 book chapters, two edited books and two research monographs. Current research focuses on Data Science, and Network Science with applications to the Internet of Things, and power networks with renewables. He is the (co-) inventor of 34 issued patents, and the (co-) recipient of 9 best journal paper awards from the IEEE Signal Processing (SP) and Communications Societies, including the G. Marconi Prize Paper Award in Wireless Communications. He also received the IEEE-SPS Norbert Wiener Society Award (2019); EURASIP's A. Papoulis Society Award (2020); Technical Achievement Awards from the IEEE-SPS (2000) and from EURASIP (2005); the IEEE ComSoc Education Award (2019); the G. W. Taylor Award for Distinguished Research from the University of Minnesota, Minneapolis, Minnesota, and the IEEE Fourier Technical Field Award (2015). He is a foreign member of the Academia Europaea, and fellow of the National Academy of Inventors, the European Academy of Sciences and EURASIP. He has served the IEEE in a number of posts, including that of a distinguished lecturer for the IEEE-SPS.



Qinmin Yang (Senior Member, IEEE) received the bachelor's degree in electrical engineering from Civil Aviation University of China, Tianjin, China, in 2001, the master of science degree in control science and engineering from the Institute of Automation, Chinese Academy of Sciences, Beijing, China, in 2004, and the PhD degree in electrical engineering from the University of Missouri-Rolla, Rolla, Missouri, in 2007. From 2007 to 2008, he was a post-doctoral research associate at University of Missouri-Rolla, Rolla, Missouri. From 2008 to 2009, he was a system engineer with Caterpillar Inc. From 2009 to 2010, he was a post-doctoral research associate at University of Connecticut, Mansfield, Connecticut. Since 2010, he has been with the State Key Laboratory of Industrial Control Technology, the College of Control Science and Engineering, Zhejiang University, China, where he is currently a professor. He has also held visiting positions in University of Toronto, Canada and Lehigh University, Bethlehem, Pennsylvania. He has been serving as an associate editor for the *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, the *IEEE Transactions on Neural Networks and Learning Systems*, the *Transactions of the Institute of Measurement and Control*, and the *Automatica Sinica*. His research interests include intelligent control, renewable energy systems, smart grid, and industrial Big Data.

Zaiyue Yang (Member, IEEE) received the BS and MS degrees from the Department of Automation, University of Science and Technology of China, Hefei, China, in 2001 and 2004, respectively, and the PhD degree from the Department of Mechanical Engineering, University of Hong Kong, Hong Kong, in 2008. He was a postdoctoral fellow and research associate with the Department of Applied Mathematics, Hong Kong Polytechnic University, Hong Kong, before joining the College of Control Science and Engineering, Zhejiang University, Hangzhou, China, in 2010. Then, he joined the Department of Mechanical and Energy Engineering, Southern University of Science and Technology, Shenzhen, China, in 2017. He is currently a professor there. His current research interests include smart grid, signal processing and control theory. He is also an associate editor for the *IEEE Transactions on Industrial Informatics*.