SQuARM-SGD: Communication-Efficient Momentum SGD for Decentralized Optimization

Navjot Singh*, Deepesh Data*, Jemin George†, and Suhas Diggavi*
*University of California, Los Angeles, USA; navjotsingh@ucla.edu, deepesh.data@gmail.com, suhas@ee.ucla.edu
†US Army Research Lab, Maryland, USA; jemin.george.civ@mail.mil

***Abstract*—In this paper, we propose and analyze SQuARM-SGD, a communication-efficient algorithm for decentralized training of large-scale machine learning models over a network. In SQuARM-SGD, each node performs a fixed number of local SGD steps using Nesterov's momentum and then sends sparsified and quantized updates to its neighbors regulated by a locally computable triggering criterion. We provide convergence guarantees of our algorithm for general smooth objectives, which, to the best of our knowledge, is the first theoretical analysis for compressed decentralized SGD with momentum updates. We show that SQuARM-SGD converges at rate $\mathcal{O}(1/\sqrt{nT})$, matching that of vanilla distributed SGD. We empirically show that SQuARM-SGD saves significantly in total communicated bits over state-of-the-art without sacrificing much on accuracy.**

A full version of this paper is accessible at: https://arxiv.org/abs/2005.07041

I. Introduction

As machine learning gets deployed over edge (wireless) devices (in contrast to datacenter applications), the problem of building learning models on local (heterogeneous) data with communication-efficient training becomes important. These applications motivate learning when data is collected/available locally, but devices collectively help build a model through wireless links with significant communication rate (bandwidth) constraints[1]. Several methods have been developed recently to obtain communication-efficiency in *distributed* stochastic gradient descent (SGD). These methods can be broadly divided into two categories. In the first one, workers *compress* information/gradients before communicating - either with *sparsification* [2]–[6], *quantization* [7]–[11], or both [12]. Another way to reduce communication is to skip communication rounds while performing a certain number of *local SGD* steps, thus trading-off computation and communication time [13]–[15]. Since momentum-based methods generally converge faster and generalize well, they have been adopted ubiquitously for training large-scale machine learning models [16].

To reduce communication load on the central-coordinator in the distributed framework, a *decentralized* setting has been considered in literature [17], where the central coordinator is absent, and training is performed collaboratively across workers, which are connected by a (sparse) graph.[2] Compressed communication has been studied recently for decentralized training as well [18]–[22]. These papers only employ either quantization or sparsification (without local iterations) in a decentralized setting; importantly, they do not incorporate momentum in their theoretical analyses. In this paper, we propose and analyze SQuARM-SGD,[3] an SGD algorithm for decentralized optimization that incorporates Nesterov's momentum and reduced communication, which is achieved by compression as well as local iterations with event-triggered communication. To the best of our knowledge, ours is the first theoretical analysis for a compressed decentralized learning algorithm incorporating momentum.

For compression, SQuARM-SGD uses both sparsification and quantization. For event-triggered communication, each worker first performs a certain number of *local SGD* iterations with momentum updates; then in order to further reduce communication, it only does so if there is a significant change in the local model parameters (greater than a prescribed threshold) since its last communication. If there is a significant model change, the worker communicates a sparsified and quantized version of (the difference of) its local parameters (model) to its neighbors. Therefore, this combines lazy updates along with quantization and sparsification to enable communication-efficient decentralized training.

Our contributions. In this paper, we provide convergence guarantees of SQuARM-SGD for general smooth objectives. We show convergence rate of $\mathcal{O}\left(1/\sqrt{nT}\right)$ where n is the number of worker nodes and T is the number of iterations, thus matching the convergence rate of vanilla distributed SGD. We note that compression and event triggered communication do affect our convergence rate expressions, but they appear only in the higher order terms; thus, for a large enough T, we can converge at the same rate as that of distributed vanilla SGD while enjoying the savings in communication from our method essentially for free; see Theorem 1 and comments after that for details. As mentioned earlier, we use Nesterov's momentum in SQuARM-SGD and theoretically analyze its convergence rate; a first theoretical analysis of convergence of such compressed gradient updates with momentum in the decentralized setting. In order to achieve this, we had to solve several technical difficulties; see Section IV in our full paper [23] and also relationship of our work to literature below, where we also

[1]This is also motivated by federated learning [1], which is studied mostly for the client-server model.

[2]This can also be motivated through learning over local wireless mesh (or ad hoc) networks.

[3]Acronym stands for Sparsified and Quantized Action Regulated Momentum Stochastic Gradient Descent. See Algorithm 1 for a description of SQuARM-SGD.

compare SQuARM-SGD with CHOCO-SGD [20] – the state-of-the-art in efficient decentralized training. Our numerical results for decentralized training of ResNet20 [24] model on CIFAR-10 [25] dataset shows significant savings in total bits communicated in SQuARM-SGD compared to CHOCO-SGD to reach a target accuracy while converging at a similar rate. For a test accuracy of around 90%, SQuARM-SGD saves total communicated bits by a factor of $40\times$ compared to CHOCO-SGD [20] and around $3\text{K}\times$ compared to vanilla SGD. We provide additional experimental results in our full paper [23].

Related work. Communication-efficient decentralized training has received recent attention; see [18]–[20], [26]–[29] and references therein. The current state-of-the-art in communication efficient decentralized training is CHOCO-SGD [20], [21], which considers sparsification or quantization of the model parameters, without incorporating momentum in their theoretical analyses.[4] Our convergence analyses are very different and significantly more involved than that of CHOCO-SGD, as apart from studying local iterations and event-triggered communication in decentralized SGD, unlike [20], [21], we provide our analyses using virtual sequences (see (3) in Section IV), specifically, to handle the use of momentum. The use of local iterations with momentum updates in decentralized setting is studied in [29], [30], but without any compression of exchanged information. [31] studied momentum SGD with compressed updates (but no local iterations or event-triggering) for the *distributed* setting only, assuming that all workers have access to unbiased gradients. For distributed setting, [12] considers compressed communication with local iterations, but without incorporating momentum updates in theoretical analysis. Extending the analysis to the *decentralized* setting (where different workers may have local data, potentially generated from different distributions) while incorporating compression, local iterations and event triggered communication in SQuARM-SGD while analyzing the resulting algorithm with momentum updates poses several challenges; see our full paper [23] for a detailed discussion. The idea of event-triggering has been explored in the control community [32]–[36] and in the optimization literature [37]–[39]. These papers focus on continuous-time, deterministic optimization algorithms for convex problems; in contrast, we propose event-driven stochastic gradient descent algorithms for general smooth objectives, e.g., large-scale deep learning. [40] propose an adaptive scheme to skip gradient computations in a *distributed* setting for *deterministic* gradients; moreover, their focus is on saving communication rounds, without compressed communication. In summary, to the best of our knowledge, ours is the first paper to develop and analyze convergence of momentum-based decentralized stochastic optimization, using compressed lazy communication (as described earlier). Moreover, our numerics demonstrate a significant advantage over the state-of-the-art in terms of communication efficiency.

Paper organization. The problem setup and SQuARM-SGD are described in Section II. Section III states our main convergence result and we provide a proof sketch for it in Section IV. Section V gives numerical results comparing our algorithm to the state-of-the-art. Omitted proofs and additional numerics can be found in our full paper [23].

[4]They do report numerics with momentum, and we compare the numerical performance SQuARM-SGD with it in Section V.

II. Problem Setup and Our Algorithm

We first formalize the decentralized optimization setting that we work with and set up the notation we follow throughout the paper. Consider an undirected connected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with $\mathcal{V} = [n] := \{1, 2, \ldots, n\}$, where node $i \in [n]$ corresponds to worker i and we denote the neighbors of node i by $\mathcal{N}_i := \{(i,j) : (i,j) \in \mathcal{E}\}$. To each node $i \in [n]$, we associate a dataset $\mathcal{D}_i$ and an objective function $f_i : \mathbb{R}^d \to \mathbb{R}$. We allow the datasets and objective functions to be different for each node and assume that for $i \in [n]$, the objective function f_i has the form $f_i(\mathbf{x}) = \mathbb{E}_{\xi_i \sim \mathcal{D}_i}[F_i(\mathbf{x}, \xi_i)]$ where $\xi_i \sim \mathcal{D}_i$ denotes a random sample from $\mathcal{D}_i$, $\mathbf{x}$ denotes the parameter vector, and $F_i(\mathbf{x}, \xi_i)$ denotes the risk associated with sample ξ_i with respect to (w.r.t.) the parameter vector $\mathbf{x}$. Consider the following empirical risk minimization problem, where $f : \mathbb{R}^d \to \mathbb{R}$ is called the global objective function:

$$\arg\min_{\mathbf{x} \in \mathbb{R}^d} \Big(f(\mathbf{x}) := \frac{1}{n} \sum_{i=1}^{n} f_i(\mathbf{x}) \Big), \tag{1}$$

The nodes in $\mathcal{G}$ wish to minimize (1) collaboratively in a communication-efficient manner.

We now state the notation relevant to describing our algorithm. Let $W \in \mathbb{R}^{n \times n}$ denote the connectivity matrix of $\mathcal{G}$, where for every $(i,j) \in \mathcal{E}$, the (i,j)'th entry of W denotes the weight w_{ij} on the edge (i,j) – e.g., w_{ij} may represent the strength of the connection on the edge (i,j) – and for other pairs $(i,j) \notin \mathcal{E}$, the weight w_{ij} is zero. We assume that W is symmetric and doubly stochastic, which means it has non-zero entries with each row and column summing up to 1. Consider the ordered eigenvalues of W, $|\lambda_1(W)| \geq |\lambda_2(W)| \geq \ldots \geq |\lambda_n(W)|$. For such a W associated with a connected graph $\mathcal{G}$, it is known that $\lambda_1(W) = 1$ and $\lambda_i(W) \in (-1, 1)$ for all $i \in \{2, \ldots, n\}$. The spectral gap $\delta \in (0, 1]$ is defined as $\delta := 1 - |\lambda_2(W)|$. Simple matrices W having $\delta \in (0, 1]$ are known to exist for connected graphs [21].

To achieve compression on the communication exchanged between workers, we use arbitrary compression operators as defined next.

Definition 1 (Compression, [5])**.** *A (possibly randomized) function $\mathcal{C} : \mathbb{R}^d \to \mathbb{R}^d$ is called a* compression *operator, if there exists a positive constant $\omega \in (0, 1]$, such that for every $\mathbf{x} \in \mathbb{R}^d$:*

$$\mathbb{E}_{\mathcal{C}}[\|\mathbf{x} - \mathcal{C}(\mathbf{x})\|_2^2] \leq (1 - \omega)\|\mathbf{x}\|_2^2, \tag{2}$$

where expectation is taken over the randomness of $\mathcal{C}$. We assume that $\mathcal{C}(\mathbf{0}) = \mathbf{0}$.

We now list some important sparsifiers and quantizers following the above definition of a compression operator:
(i) Top_k and $Rand_k$ sparsifiers (where only k entries are

selected and the rest are set to zero) with $\omega = k/d$ [5], (ii) Stochastic quantizer Q_s from [7][5] with $\omega = (1-\beta_{d,s})$ for $\beta_{d,s} < 1$, and (iii) Deterministic quantizer $\frac{\|\mathbf{x}\|_1}{d} Sign(\mathbf{x})$ from [10] with $\omega = \frac{\|\mathbf{x}\|_1^2}{d\|\mathbf{x}\|_2^2}$. For $Comp_k \in \{Top_k, Rand_k\}$, the following are compression operators[6]: (iv) $\frac{1}{(1+\beta_{k,s})} Q_s(Comp_k)$ with $\omega = \left(1 - \frac{k}{d(1+\beta_{k,s})}\right)$ for any $\beta_{k,s} \geq 0$, and (v) $\frac{\|Comp_k(\mathbf{x})\|_1 Sign(Comp_k(\mathbf{x}))}{k}$ with $\omega = \max\left\{\frac{1}{d}, \frac{k}{d}\left(\frac{\|Comp_k(\mathbf{x})\|_1^2}{d\|Comp_k(\mathbf{x})\|_2^2}\right)\right\}$ [12].

A. Our Algorithm: SQuARM-SGD

We propose SQuARM-SGD to minimize (1), which is a decentralized algorithm that combines compression and Nesterov's momentum, together with event-driven communication exchange, where compression is achieved by sparsifying *and* quantizing the exchanges. Each worker is required to complete a fixed number of *local SGD* steps with *momentum*, and communicate *compressed* updates to its neighbors when there is a *significant change* in its local parameters since the last communication round. SQuARM-SGD can be seen as a significant extension of CHOCO-SGD [20], [21], which only performs compression using sparsification *or* quantization, without local iterations, event-triggered communication, or momentum.

To realize exchange of compressed parameters between workers, for each node $i \in [n]$, all nodes $j \in \mathcal{N}_i$ maintain an estimate $\hat{\mathbf{x}}_i$ of $\mathbf{x}_i$. Each node $i \in [n]$ has access to $\hat{\mathbf{x}}_j$ for all $j \in \mathcal{N}_i$. Our algorithm runs for T iterations and the set of synchronization indices is defined as $\mathcal{I}_T = \{I_{(1)}, \ldots, I_{(k)}, \ldots\} \subseteq [T]$, which are same for all workers and denote the time steps at which workers are allowed to communicate, provided they satisfy a triggering condition[7]. For $\mathcal{I}_T$, we define its gap as $gap(\mathcal{I}_T) := \max_m \{I_{(m)} - I_{(m-1)}\}$. We assume that $gap(\mathcal{I}_T) = H$. SQuARM-SGD is described below in Algorithm 1.

For a given connected graph $\mathcal{G}$ with connectivity matrix W, we first initialize a consensus step-size γ (see Theorem 1 for definition), momentum factor β, learning rate η, triggering threshold sequence $\{c_t\}_{t=0}^T$, and momentum vector $\mathbf{v}_i$ for each node i initialized to $\mathbf{0}$. We initialize the copies of all the nodes $\hat{\mathbf{x}}_i = \mathbf{0}$ and allow each node to communicate in the first synchronization round. At each time step t, each worker $i \in [n]$ samples a stochastic gradient $\nabla F(\mathbf{x}_i^{(t)}, \xi_i)$ and takes a local SGD step on parameter $\mathbf{x}_i^{(t)}$ using Nesterov's momentum to form an intermediate parameter $\mathbf{x}_i^{(t+1/2)}$ (line 3-5). If the next iteration corresponds to a synchronization index, i.e., $(t+1) \in \mathcal{I}_T$, then each worker checks the triggering condition (line 8). If satisfied, that worker communicates the compressed change in its copy to all its neighbors $\mathcal{N}_i$ (line 9-10); otherwise, it does not communicate in that round (denoted by 'Send $\mathbf{0}$' in our algorithm for illustration, line 12). After receiving the compressed updates of copies from all neighbors, the node i updates the locally available copies and its own copy (line 14). With these updated copies, the worker nodes finally take a consensus (line 16) with appropriate weighting decided by entries of W. In the case when $(t+1) \notin \mathcal{I}_T$, the nodes maintain their copies and move on to next iteration (line 18); thus no communication takes place.

Algorithm 1 SQuARM-SGD: Sparsified and Quantized Action Regulated Momentum SGD

Parameters: $G = ([n], E)$, W

1: **Initialize:** For every $i \in [n]$, set arbitrary $\mathbf{x}_i^{(0)} \in \mathbb{R}^d$, $\hat{\mathbf{x}}_i^{(0)} := \mathbf{0}$, $\mathbf{v}_i^{(-1)} := \mathbf{0}$. Fix the momentum coefficient β, consensus step-size γ, learning rate η, triggering thresholds $\{c_t\}_{t=0}^T$, and synchronization set $\mathcal{I}_T$.
2: **for** $t = 0$ **to** $T-1$ in parallel for all workers $i \in [n]$ **do**
3: Sample $\xi_i^{(t)}$, stochastic gradient $\mathbf{g}_i^{(t)} := \nabla F_i(\mathbf{x}_i^{(t)}, \xi_i^{(t)})$
4: $\mathbf{v}_i^{(t)} = \beta \mathbf{v}_i^{(t-1)} + \mathbf{g}_i^{(t)}$
5: $\mathbf{x}_i^{(t+\frac{1}{2})} := \mathbf{x}_i^{(t)} - \eta(\beta \mathbf{v}_i^{(t)} + \mathbf{g}_i^{(t)})$
6: **if** $(t+1) \in I_T$ **then**
7: **for** neighbors $j \in \mathcal{N}_i \cup i$ **do**
8: **if** $\|\mathbf{x}_i^{(t+\frac{1}{2})} - \hat{\mathbf{x}}_i^{(t)}\|_2^2 > c_t \eta^2$ **then**
9: Compute $\mathbf{q}_i^{(t)} := \mathcal{C}(\mathbf{x}_i^{(t+\frac{1}{2})} - \hat{\mathbf{x}}_i^{(t)})$
10: Send $\mathbf{q}_i^{(t)}$ and receive $\mathbf{q}_j^{(t)}$
11: **else**
12: Send $\mathbf{0}$ and receive $\mathbf{q}_j^{(t)}$
13: **end if**
14: $\hat{\mathbf{x}}_j^{(t+1)} := \mathbf{q}_j^{(t)} + \hat{\mathbf{x}}_j^{(t)}$
15: **end for**
16: $\mathbf{x}_i^{(t+1)} = \mathbf{x}_i^{(t+\frac{1}{2})} + \gamma \sum_{j \in \mathcal{N}_i} w_{ij}(\hat{\mathbf{x}}_j^{(t+1)} - \hat{\mathbf{x}}_i^{(t+1)})$
17: **else**
18: $\hat{\mathbf{x}}_i^{(t+1)} = \hat{\mathbf{x}}_i^{(t)}$, $\mathbf{x}_i^{(t+1)} = \mathbf{x}_i^{(t+\frac{1}{2})}$ for all $i \in [n]$
19: **end if**
20: **end for**

Memory-efficient version of Algorithm 1: At a first glance, it may seem that in Algorithm 1, every node has to store estimates of all its neighbors' parameters in order to perform the consensus step, which may be impractical in large-scale learning. Note that in the consensus step (line 16), nodes only require the weighted sum of their neighbors' parameters. So, it suffices for each node to store only the weighted sum of all its neighbors' parameters (in addition to its own local parameters and its estimate), and thus avoiding the need to store all neighbor parameters. We provide a memory-efficient version of SQuARM-SGD in our full paper [23].

Equivalence to error-feedback mechanisms: In Algorithm 1, though nodes do not explicitly perform local error-compensation (as in [10], [12]), the error-compensation happens implicitly. To see this, note that nodes maintain copies of their neighbors' parameters and update them as $\hat{\mathbf{x}}_j^{(t+1)} = \hat{\mathbf{x}}_j^{(t)} + \mathcal{C}(\mathbf{x}_j^{(t+\frac{1}{2})} - \hat{\mathbf{x}}_j^{(t)})$ (line 14) and then take the consensus

[5] $Q_s : \mathbb{R}^d \to \mathbb{R}^d$ is a stochastic quantizer, if for every $\mathbf{x} \in \mathbb{R}^d$, we have (i) $\mathbb{E}[Q_s(\mathbf{x})] = \mathbf{x}$ and (ii) $\mathbb{E}[\|\mathbf{x} - Q_s(\mathbf{x})\|_2^2] \leq \beta_{d,s}\|\mathbf{x}\|_2^2$. Q_s from [7] satisfies this definition with $\beta_{d,s} = \min\left\{\frac{d}{s^2}, \frac{\sqrt{d}}{s}\right\}$.

[6] [12] show that the composition of sparsification and quantization operators is also a valid compression operator, outperforming its individual components in terms of communication savings while maintaining similar performance.

[7] The Zeno phenomenon [32] does not occur in our setup as we have a discrete sampling period as well as a fixed number of local iterations, giving a lower bound to the event intervals of at least H times the sampling period.

step (line 16). Thus, the error gets accumulated into $\hat{\mathbf{x}}_j^{(t)}$ and is compensated by the term $\mathcal{C}(\mathbf{x}_j^{(t+\frac{1}{2})} - \hat{\mathbf{x}}_j^{(t)})$ in the next round.

III. Main Results

In this section, we state the convergence rate for SQuARM-SGD for general smooth objectives. We provide our result under the following assumptions:
(i) *Bounded variance:* For every $i \in [n]$, we have $\mathbb{E}_{\xi_i}\|\nabla F_i(\mathbf{x}, \xi_i) - \nabla f_i(\mathbf{x})\|^2 \leq \sigma_i^2$, for some finite σ_i, where $\nabla F_i(\mathbf{x}, \xi_i)$ denotes an unbiased stochastic gradient at worker i with $\mathbb{E}_{\xi_i}[\nabla F_i(\mathbf{x}, \xi_i)] = \nabla f_i(\mathbf{x})$. We define $\bar{\sigma}^2 := \frac{1}{n}\sum_{i=1}^n \sigma_i^2$.
(ii) *Bounded second moment:* For every $i \in [n]$, we have $\mathbb{E}_{\xi_i}\|\nabla F_i(\mathbf{x}, \xi_i)\|^2 \leq G^2$, for some finite G. This is a standard assumption in stochastic optimization with compressed gradients [5], [12], [20], [31].
(iii) *Smoothness of objectives:* We assume that each local objective f_i is L-smooth[8] for all $i \in [n]$.

Theorem 1. *Let $\mathcal{C}$ be a compression operator with parameter $\omega \in (0, 1]$ and let $gap(\mathcal{I}_T) \leq H$. Consider running SQuARM-SGD under assumptions* (i)-(iii) *above with consensus step-size $\gamma = \frac{2\delta\omega}{64\delta + \delta^2 + 16\lambda^2 + 8\delta\lambda^2 - 16\delta\omega}$, (where $\lambda = max_i\{1 - \lambda_i(W)\}$), momentum coefficient $\beta \in [0, 1)$, learning rate $\eta = (1-\beta)\sqrt{\frac{n}{T}}$ and a threshold function $c_t \leq \frac{c_0}{\eta^{1-\epsilon}}$ for all t where $\epsilon \in (0, 1)$. If $T \geq \frac{8L^2\beta^4 n}{(1-\beta)^2}$, then the averaged iterates $\bar{\mathbf{x}}^{(t)} := \frac{1}{n}\sum_{i=0}^n \mathbf{x}_i^{(t)}$ satisfy:*

$$\frac{\sum_{t=0}^{T-1}\mathbb{E}\|\nabla f(\bar{\mathbf{x}}^{(t)})\|_2^2}{T} = \mathcal{O}\left(\frac{J^2 + \bar{\sigma}^2}{\sqrt{nT}}\right) + \mathcal{O}\left(\frac{c_0 n^{(1+\epsilon)/2}}{\delta^2 T^{(1+\epsilon)/2}} + \frac{nH^2G^2}{T\delta^4\omega^2} + \frac{\beta^4\bar{\sigma}^2}{T(1-\beta)^2}\right),$$

where $J^2 < \infty$ is a constant such that $\mathbb{E}[f(\bar{\mathbf{x}}^{(0)})] - f^* \leq J^2$.

We use simplified convergence rate expressions in the above results, and provide a precise rate expression in full paper [23].

Effects of parameters on convergence: Observe that the factors H, c_0, ω, δ to achieve communication efficiency – H, c_0 for the event-triggered communication, ω for compression, and δ for the connectivity of the underlying graph – do not affect the dominant term in Theorem 1 and appear only in the higher order terms. This implies that if we run SQuARM-SGD for sufficiently long, precisely, for at least $T_0 := C_0 \times \max\left\{\left(\frac{c_0^2 n^{(2+\epsilon)}}{(J^2+\bar{\sigma}^2)^2\delta^4}\right)^{1/\epsilon}, \frac{n}{(J^2+\bar{\sigma}^2)^2}\left(\frac{nG^2H^2}{\omega^2\delta^4} + \frac{\beta^4\bar{\sigma}^2}{(1-\beta)^2}\right)^2\right\}$ iterations for a sufficiently large constant C_0, then SQuARM-SGD converges at a rate of $\mathcal{O}\left(1/\sqrt{nT}\right)$. Note that this is the convergence rate of distributed *vanilla* SGD with the same speed-up w.r.t. the number of nodes n. Thus, we essentially converge at the same rate as vanilla SGD, while saving significantly in terms of total communication bits; this can be seen in our numerical results in Section V as well.

Theoretical justification for communication gain: We claim that convergence results for SQuARM-SGD show savings in communication compared to CHOCO-SGD [20], [21]. For the sake of argument, consider the case when SQuARM only performs local iterations without threshold based triggering (i.e., $c_t = 0, \forall t$) and without momentum ($\beta = 0$). For the same compression coefficient ω used for both SQuARM-SGD and CHOCO-SGD, to transmit the same number of bits (i.e., having the same number of communication rounds), T iterations of CHOCO would correspond to $T \times H$ iterations of SQuARM (due to H local SGD steps). Thus for the same number of bits transmitted, the bound on gradient norm for CHOCO-SGD is $\mathcal{O}(1/\sqrt{T} + 1/T)$, whereas for SQuARM-SGD it is $\mathcal{O}(1/\sqrt{HT} + H/T^{(1+\epsilon)/2})$ where $\epsilon \in (0, 1)$. Thus, for the same number of communication rounds, for a large value of T, SQuARM-SGD has a better performance compared to CHOCO-SGD (as the first dominant term is affected by H). Thus, SQuARM-SGD has a better performance while using less bits for communication, and this claim is also supported through our experiments.

Note on Consensus: Theorem 1 states that the average iterate $\bar{\mathbf{x}}^{(t)}$ converges to a stationary point. For decentralized learning, it is important to guarantee that the individual node iterates reach to a consensus. We show such consensus property to the average iterate in Lemma 1 provided in Section IV.

IV. Proof Outlines

Due to lack of space, below we only provide a proof-sketch of Theorem 1. We defer the interested reader to the complete proof in our full paper [23], where we also discuss technical challenges involved and a comparison to related works.

Proof outline of Theorem 1. Define virtual sequences $\widetilde{\mathbf{x}}_i^{(t)}$ for all $i \in [n]$ as:

$$\widetilde{\mathbf{x}}_i^{(t)} = \mathbf{x}_i^{(t)} - \frac{\eta\beta^2}{(1-\beta)}\mathbf{v}_i^{(t-1)}. \tag{3}$$

Consider the collection of iterates $\{\mathbf{x}_i^{(t)}\}_{t=0}^{T-1}$, $i \in [n]$ generated by Algorithm 1 at time t. Let $\overline{\mathbf{x}}^{(t)} = \frac{1}{n}\sum_{i=1}^n \mathbf{x}_i^{(t)}$. We first argue that $\overline{\mathbf{x}}^{(t+1)} = \overline{\mathbf{x}}^{(t+\frac{1}{2})}$. This trivially holds when $(t+1) \notin \mathcal{I}_T$ (line 18). For the other case, i.e., $(t+1) \in \mathcal{I}_T$ (line 18), this follows because $\sum_{i=1}^n\sum_{j=1}^n w_{ij}(\hat{\mathbf{x}}_j^{(t+1)} - \hat{\mathbf{x}}_i^{(t+1)}) = 0$, as W is doubly stochastic. Thus, using the definition of $\mathbf{x}_i^{(t+\frac{1}{2})}$ from line 5, we get

$$\overline{\mathbf{x}}^{(t+1)} = \overline{\mathbf{x}}^{(t)} - \frac{\eta}{n}\sum_{j=1}^n(\beta\mathbf{v}_j^{(t)} + \nabla F_j(\mathbf{x}_j^{(t)}, \xi_j^{(t)})). \tag{4}$$

Define $\widetilde{\mathbf{x}}^{(t)} = \frac{1}{n}\sum_1^n \widetilde{\mathbf{x}}_i^{(t)}$. Taking an average of (3) gives $\widetilde{\mathbf{x}}^{(t)} = \overline{\mathbf{x}}^{(t)} - \frac{\eta\beta^2}{(1-\beta)}\frac{1}{n}\sum_{i=1}^n \mathbf{v}_i^{(t-1)}$. Together with (4), we can show the following recurrence relation on $\widetilde{\mathbf{x}}^{(t)}$:

$$\widetilde{\mathbf{x}}^{(t+1)} = \widetilde{\mathbf{x}}^{(t)} - \frac{\eta}{(1-\beta)}\frac{1}{n}\sum_{i=1}^n \nabla F_i(\mathbf{x}_i^{(t)}, \xi_i^{(t)}). \tag{5}$$

Evaluating the objective function f at $\widetilde{\mathbf{x}}^{(t+1)}$ and using (5) along with some algebraic manipulations, we get:

$$Z_1\|\frac{1}{n}\sum_{i=1}^n \nabla f_i(\mathbf{x}_i^{(t)})\|^2 \leq f(\widetilde{\mathbf{x}}^{(t)}) - f(\widetilde{\mathbf{x}}^{(t+1)})$$

[8] f_i is L-smooth, if for every $\mathbf{x}, \mathbf{y} \in \mathbb{R}^d$, we have $f_i(\mathbf{y}) \leq f_i(\mathbf{x}) + \langle\nabla f_i(\mathbf{x}), \mathbf{y} - \mathbf{x}\rangle + \frac{L}{2}\|\mathbf{y} - \mathbf{x}\|^2$.

$$+\frac{L\eta^2\bar{\sigma}^2}{2n(1-\beta)^2}+\frac{L^2\eta}{(1-\beta)n}\left(R_1(t)+R_2(t)\right) \quad (6)$$

where $Z_1=\frac{\eta}{2(1-\beta)}-\frac{L\eta^2}{2(1-\beta)^2}$, $R_1(t)=\sum_{i=1}^n\|\mathbf{x}_i^{(t)}-\overline{\mathbf{x}}^{(t)}\|^2$, $R_2(t)=\sum_{i=1}^n\|\overline{\mathbf{x}}^{(t)}-\widetilde{\mathbf{x}}^{(t)}\|^2$. Taking expectation and using Lemma 1 (stated below and proved in our full paper [23]) to bound $\mathbb{E}[R_2(t)]$ with $\eta\leq\frac{(1-\beta)^2}{2L}$, we get:

$$\Sigma:=\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\|\frac{1}{n}\sum_{i=1}^n\nabla f_i(\mathbf{x}_i^{(t)})\|^2\leq$$
$$\frac{4(1-\beta)}{\eta}\frac{(\mathbb{E}[f(\mathbf{x}^{(0)})]-f^*)}{T}+\frac{4L^2}{nT}\sum_{t=0}^{T-1}\mathbb{E}[R_1(t)]+Z_2 \quad (7)$$

where $Z_2=\frac{2L\eta\bar{\sigma}^2}{n(1-\beta)}+\frac{4L^2\eta^2\beta^4\bar{\sigma}^2}{n(1-\beta)^4}$. Note that (7) bounds $\Sigma=\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\|\frac{1}{n}\sum_{i=1}^n\nabla f_i(\mathbf{x}_i^{(t)})\|^2$, whereas, our goal is to bound $\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\|\nabla f(\overline{\mathbf{x}}^{(t)})\|^2$. Using L-Lipschitz gradient property of f gives $\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\|\nabla f(\overline{\mathbf{x}}^{(t)})\|^2\leq\frac{2L^2}{nT}\sum_{t=0,}^{T-1}\mathbb{E}[R_1(t)]+2\Sigma$. Substituting the bound on Σ from (7), we get

$$\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\|\nabla f(\overline{\mathbf{x}}^{(t)})\|^2\leq\frac{8(1-\beta)}{\eta}\frac{(\mathbb{E}[f(\mathbf{x}^{(0)})]-f^*)}{T}$$
$$+\frac{10L^2}{nT}\sum_{t=0}^{T-1}\mathbb{E}[R_1(t)]+2Z_2. \quad (8)$$

Lemma 1. *Consider the assumptions of Theorem 1 and let $I_{(t)}\in\mathcal{I}_T$ be a synchronization index. Let $A=\frac{p}{2}\left(2H^2G^2\left(1+\frac{\beta^2}{(1-\beta)^2}\right)\left(\frac{16}{\omega}+\frac{4}{p}\right)+\frac{2\omega c_0}{\eta^{(1-\epsilon)}}\right)$. Then:*

$$\sum_{j=1}^n\mathbb{E}\|\overline{\mathbf{x}}^{I_{(t)}}-\mathbf{x}_j^{I_{(t)}}\|^2\leq\frac{4nA\eta^2}{p^2}$$
$$\mathbb{E}[R_2(t)]=\sum_{i=1}^n\mathbb{E}\|\overline{\mathbf{x}}^{(t)}-\widetilde{\mathbf{x}}^{(t)}\|^2\leq\frac{n\beta^4\eta^2}{(1-\beta)^3}\sum_{\tau=0}^{t-1}\phi(t,\tau)$$

where $\phi(t,\tau)=\beta^{t-\tau-1}\mathbb{E}\left\|\frac{1}{n}\sum_{i=1}^n\nabla F_i(\mathbf{x}_i^{(\tau)},\xi_i^{(\tau)})\right\|^2$.

Let $I_{(t_0)}\in\mathcal{I}_T$ denote the last synchronization index before time t. The assumption $gap(\mathcal{I}_T)\leq H$ implies $t-I_{(t_0)}\leq H$. Using this and assumption (ii), we can bound:

$$\mathbb{E}[R_1(t)]=\sum_{j=1}^n\mathbb{E}\|\overline{\mathbf{x}}^{(t)}-\mathbf{x}_j^{(t)}\|^2\leq2\mathbb{E}\sum_{j=1}^n\|\overline{\mathbf{x}}^{I_{(t_0)}}-\mathbf{x}_j^{I_{(t_0)}}\|^2$$
$$+4\eta^2H^2nG^2\left(1+\frac{\beta^2}{(1-\beta)^2}\right) \quad (9)$$

Now use Lemma 1 to get a bound on $\mathbb{E}[R_1(t)]$ and then substitute the resulting bound into (8). We then expand on the value of A from Lemma 1 and set $\eta=(1-\beta)\sqrt{\frac{n}{T}}$ for $T\geq\frac{8L^2\beta^4n}{(1-\beta)^2}$. Using $p\geq\frac{\delta^2\omega}{644}$ completes the proof of Theorem 1.

V. Experiments

In this section, we compare SQuARM-SGD against CHOCO-SGD [20], which is the state-of-the-art in communication-efficient decentralized training, and also against vanilla decentralized SGD [41]. We provide additional experiments, including comparison of wall clock training time in our full paper [23].

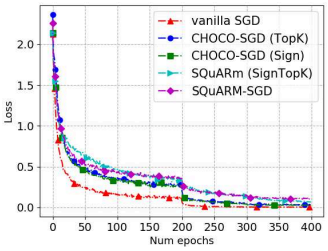

Fig. 1 Comparison of training loss against number of epochs for different schemes.

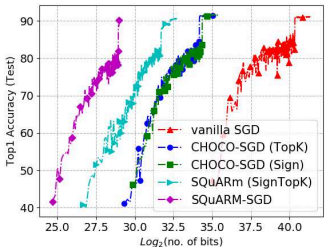

Fig. 2 Comparison of test accuracy with total number of bits communicated between nodes.

Fig. 3 Performance comparison for training ResNet-20 model on CIFAR-10 dataset for different algorithms.

Setup. We match the setting in CHOCO-SGD and perform our experiments on the CIFAR-10 [25] dataset to train a ResNet20 [42] model with $n=8$ nodes connected in a ring topology. The training is done on TitanRTX GPUs. Learning rate is initialized to 0.1 and follows a schedule consisting of a warmup period of 5 epochs and has a piecewise decay of 5 at epoch 200 and 300, and we stop training at epoch 400. The SGD algorithm is implemented with momentum with a factor of 0.9 and mini-batch size of 256. SQuARM-SGD consists of $H=5$ local iterations and we take top 1% elements of each tensor and only transmit the sign and norm of the result. The triggering threshold follows a schedule piecewise constant: initialized to 2.5 and increases by 1.5 after every 20 epochs till 350 epochs are complete, while maintaining that $c_t<1/\eta$ for all t. We compare performance of SQuARM-SGD against CHOCO-SGD with $Sign$, $TopK$ compression (taking top 1% of elements of the tensor) and decentralized vanilla SGD [17].

Results. We plot global loss function evaluated at average parameter across nodes in Figure 1, where we observe SQuARM-SGD converging at a similar rate as CHOCO-SGD and vanilla decentralized SGD. Figure 2 shows the performance for a given bit-budget, where we show the Top-1 test accuracy[9] as a function of the total bits communicated. For a test-accuracy of around 90%, SQuARM-SGD requires about 40× less bits than CHOCO-SGD with $Sign$ or $TopK$ compression, and around 3K× less bits than vanilla decentralized SGD to achieve the same Top-1 accuracy.

Acknowledgments

This work was partially supported by NSF grant #2007714 and by UC-NL grant LFR-18-548554 and by Army Research Laboratory under Cooperative Agreement W911NF-17-2-0196. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Laboratory or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation here on.

[9]Here, Top-1 accuracy corresponds to the percentage of test examples for which the correct label is equal to the label considered most probable by the model.

REFERENCES

[1] J. Konečnỳ, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, "Federated learning: Strategies for improving communication efficiency," in *Advances in Neural Information Processing Systems, NIPS*, 2016.

[2] N. Strom, "Scalable distributed DNN training using commodity GPU cloud computing," in *Annual Conference of the International Speech Communication Association, INTERSPEECH*, 2015, pp. 1488–1492.

[3] A. F. Aji and K. Heafield, "Sparse communication for distributed gradient descent," in *Proceedings of Conference on Empirical Methods in Natural Language Processing, EMNLP*, 2017, pp. 440–445.

[4] Y. Lin, S. Han, H. Mao, Y. Wang, and W. J. Dally, "Deep gradient compression: Reducing the communication bandwidth for distributed training," in *International Conference on Learning Representations, ICLR*, 2018.

[5] S. U. Stich, J.-B. Cordonnier, and M. Jaggi, "Sparsified SGD with Memory," in *Advances in Neural Information Processing Systems, NeurIPS*, 2018, pp. 4447–4458.

[6] D. Alistarh, T. Hoefler, M. Johansson, N. Konstantinov, S. Khirirat, and C. Renggli, "The convergence of sparsified gradient methods," in *Advances in Neural Information Processing Systems, NeurIPS*, 2018, pp. 5973–5983.

[7] D. Alistarh, D. Grubic, J. Li, R. Tomioka, and M. Vojnovic, "QSGD: communication-efficient SGD via gradient quantization and encoding," in *Advances in Neural Information Processing Systems, NIPS*, 2017, pp. 1709–1720.

[8] W. Wen, C. Xu, F. Yan, C. Wu, Y. Wang, Y. Chen, and H. Li, "Terngrad: Ternary gradients to reduce communication in distributed deep learning," in *Advances in Neural Information Processing Systems, NIPS*, 2017, pp. 1508–1518.

[9] A. T. Suresh, F. X. Yu, S. Kumar, and H. B. McMahan, "Distributed mean estimation with limited communication," in *International Conference on Machine Learning, ICML*, 2017, pp. 3329–3337.

[10] S. P. Karimireddy, Q. Rebjock, S. U. Stich, and M. Jaggi, "Error feedback fixes signsgd and other gradient compression schemes," in *International Conference on Machine Learning, ICML*, 2019, pp. 3252–3261.

[11] J. Bernstein, Y.-X. Wang, K. Azizzadenesheli, and A. Anandkumar, "signsgd: Compressed optimisation for non-convex problems," in *International Conference on Machine Learning, ICML*, 2018, pp. 560–569.

[12] D. Basu, D. Data, C. Karakus, and S. N. Diggavi, "Qsparse-local-sgd: Distributed SGD with quantization, sparsification and local computations," in *Advances in Neural Information Processing Systems, NeurIPS*, 2019, pp. 14 668–14 679.

[13] S. U. Stich, "Local SGD Converges Fast and Communicates Little," in *International Conference on Learning Representations, ICLR*, 2019.

[14] H. Yu, S. Yang, and S. Zhu, "Parallel restarted SGD with faster convergence and less communication:demystifying why model averaging works for deep learning," in *AAAI Conference on Artificial Intelligence, AAAI*, 2019, pp. 5693–5700.

[15] G. F. Coppola, "Iterative parameter mixing for distributed large-margin training of structured predictors for natural language processing," Ph.D. dissertation, University of Edinburgh, UK, 2015.

[16] Y. Yan, T. Yang, Z. Li, Q. Lin, and Y. Yang, "A unified analysis of stochastic momentum methods for deep learning," in *Proceedings of the International Joint Conference on Artificial Intelligence, IJCAI*, 2018, pp. 2955–2961.

[17] X. Lian, C. Zhang, H. Zhang, C.-J. Hsieh, W. Zhang, and J. Liu, "Can decentralized algorithms outperform centralized algorithms? a case study for decentralized parallel stochastic gradient descent," in *Advances in Neural Information Processing Systems, NIPS*, 2017, pp. 5330–5340.

[18] H. Tang, S. Gan, C. Zhang, T. Zhang, and J. Liu, "Communication compression for decentralized training," in *Advances in Neural Information Processing Systems, NeurIPS*, 2018, pp. 7663–7673.

[19] N. Singh, D. Data, J. George, and S. Diggavi, "Sparq-sgd: Event-triggered and compressed communication in decentralized optimization," in *2020 59th IEEE Conference on Decision and Control (CDC)*. IEEE, 2020, pp. 3449–3456.

[20] A. Koloskova, T. Lin, S. U. Stich, and M. Jaggi, "Decentralized Deep Learning with Arbitrary Communication Compression," in *International Conference on Learning Representations, ICLR*, 2020.

[21] A. Koloskova, S. U. Stich, and M. Jaggi, "Decentralized Stochastic Optimization and Gossip Algorithms with Compressed Communication," in *International Conference on Machine Learning, ICML*, 2019, pp. 3478–3487.

[22] H. Tang, C. Yu, X. Lian, T. Zhang, and J. Liu, "Doublesqueeze: Parallel stochastic gradient descent with double-pass error-compensated compression," in *International Conference on Machine Learning, ICML*, 2019, pp. 6155–6165.

[23] N. Singh, D. Data, J. George, and S. Diggavi, "Squarm-sgd: Communication-efficient momentum sgd for decentralized optimization," *arXiv preprint arXiv:2005.07041*, 2020.

[24] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *IEEE Conference on Computer Vision and Pattern Recognition, CVPR*, 2016, pp. 770–778.

[25] A. Krizhevsky, V. Nair, and G. Hinton, "Cifar-10," *Canadian Institute for Advanced Research*, 2009. [Online]. Available: http://www.cs.toronto.edu/~kriz/cifar.html

[26] A. Reisizadeh, A. Mokhtari, H. Hassani, and R. Pedarsani, "Quantized decentralized consensus optimization," in *IEEE Conference on Decision and Control, CDC*, 2018, pp. 5838–5843.

[27] M. Assran, N. Loizou, N. Ballas, and M. Rabbat, "Stochastic gradient push for distributed deep learning," in *International Conference on Machine Learning, ICML*, 2019, pp. 344–353.

[28] T. Tatarenko and B. Touri, "Non-convex distributed optimization," *IEEE Transactions on Automatic Control*, vol. 62, no. 8, pp. 3744–3757, 2017.

[29] H. Yu, R. Jin, and S. Yang, "On the linear speedup analysis of communication efficient momentum SGD for distributed non-convex optimization," in *International Conference on Machine Learning, ICML*, 2019, pp. 7184–7193.

[30] J. Wang, V. Tantia, N. Ballas, and M. Rabbat, "Slowmo: Improving communication-efficient distributed sgd with slow momentum," in *International Conference on Learning Representations, ICLR*, 2020.

[31] S. Zheng, Z. Huang, and J. Kwok, "Communication-efficient distributed blockwise momentum sgd with error-feedback," in *Advances in Neural Information Processing Systems, NeurIPS*, 2019, pp. 11 446–11 456.

[32] W. P. M. H. Heemels, K. H. Johansson, and P. Tabuada, "An introduction to event-triggered and self-triggered control," in *IEEE Conference on Decision and Control, CDC*, 2012, pp. 3270–3285.

[33] D. V. Dimarogonas, E. Frazzoli, and K. H. Johansson, "Distributed event-triggered control for multi-agent systems," *IEEE Transactions on Automatic Control*, vol. 57, no. 5, pp. 1291–1297, 2012.

[34] G. S. Seyboth, D. V. Dimarogonas, and K. H. Johansson, "Event-based broadcasting for multi-agent average consensus," *Automatica*, vol. 49, no. 1, pp. 245–252, 2013.

[35] A. Girard, "Dynamic triggering mechanisms for event-triggered control," *IEEE Transactions on Automatic Control*, vol. 60, no. 7, pp. 1992–1997, 2015.

[36] Y. Liu, C. Nowzari, Z. Tian, and Q. Ling, "Asynchronous periodic event-triggered coordination of multi-agent systems," in *IEEE Conference on Decision and Control, CDC*, 2017, pp. 6696–6701.

[37] S. S. Kia, J. Cortés, and S. Martínez, "Distributed convex optimization via continuous-time coordination algorithms with discrete-time communication," *Automatica*, vol. 55, pp. 254–264, 2015.

[38] W. Chen and W. Ren, "Event-triggered zero-gradient-sum distributed consensus optimization over directed networks," *Automatica*, vol. 65, pp. 90–97, 2016.

[39] W. Du, X. Yi, J. George, K. H. Johansson, and T. Yang, "Distributed optimization with dynamic event-triggered mechanisms," in *IEEE Conference on Decision and Control, CDC*, 2018, pp. 969–974.

[40] T. Chen, G. Giannakis, T. Sun, and W. Yin, "Lag: Lazily aggregated gradient for communication-efficient distributed learning," in *Advances in Neural Information Processing Systems, NeurIPS*, 2018, pp. 5050–5060.

[41] X. Lian, W. Zhang, C. Zhang, and J. Liu, "Asynchronous decentralized parallel stochastic gradient descent," in *International Conference on Machine Learning, ICML*, 2017, pp. 3043–3052.

[42] W. Wen, C. Wu, Y. Wang, Y. Chen, and H. Li, "Learning structured sparsity in deep neural networks," in *Advances in Neural Information Processing Systems, NIPS*, 2016, pp. 2074–2082.