On the Convergence of Communication-Efficient Local SGD for Federated Learning

Hongchang Gao[1], **An Xu**[2], **Heng Huang**[2,3]
[1] Department of Computer and Information Sciences, Temple University, PA, USA
[2] Department of Electrical and Computer Engineering, University of Pittsburgh, PA, USA
[3] JD Finance America Corporation, Mountain View, CA, USA
hongchang.gao@temple.edu, an.xu@pitt.edu, heng.huang@pitt.edu

Abstract

Federated Learning (FL) has attracted increasing attention in recent years. A leading training algorithm in FL is local SGD, which updates the model parameter on each worker and averages model parameters across different workers only once in a while. Although it has fewer communication rounds than the classical parallel SGD, local SGD still has large communication overhead in each communication round for large machine learning models, such as deep neural networks. To address this issue, we propose a new communication-efficient distributed SGD method, which can significantly reduce the communication cost by the error-compensated double compression mechanism. Under the non-convex setting, our theoretical results show that our approach has better communication complexity than existing methods and enjoys the same linear speedup regarding the number of workers as the full-precision local SGD. Moreover, we propose a communication-efficient distributed SGD with momentum, which also has better communication complexity than existing methods and enjoys a linear speedup with respect to the number of workers. At last, extensive experiments are conducted to verify the performance of our proposed methods.

Introduction

In recent years, Federated Learning (FL) has attracted more and more attention due to data distributed across user devices that requires privacy and finite communication cost. Specifically, in a FL system, there exist multiple workers and a central server. Workers optimize the model with their local data, and the central server coordinates the corporation between different workers. In particular, FL aims at solving the following distributed optimization problem:

$$\min_x f(x) = \frac{1}{K}\sum_{k=1}^{K} f^{(k)}(x)\ , \tag{1}$$

where $x \in \mathcal{R}^d$ represents the model parameter, K is the number of workers, and $f^{(k)}(x) = \mathbb{E}_{\xi\sim\mathcal{D}^{(k)}} F^{(k)}(x;\xi)$ is the loss function on each worker with $\mathcal{D}^{(k)}$ denoting the data distribution on the k-th worker. In this paper, we focus on the non-convex problem. In particular, $f^{(k)}(x)$ is assumed as a smooth non-convex function.

To optimize Eq. (1), a straightforward method is the parallel SGD (P-SGD) method where each worker sends the local stochastic gradient to the central server and gets the averaged gradient from the central server at each iteration. This mechanism will lead to large communication overhead at each iteration, especially when the model is large, such as deep neural networks. To reduce the communication cost, the commonly used optimization method in a FL system is local SGD (Jiang and Agrawal 2018; Yu, Jin, and Yang 2019; Yu, Yang, and Zhu 2019; Wang and Joshi 2018; Haddadpour et al. 2019). Specifically, each worker updates the model parameter for p (where $p > 1$) iterations locally by using SGD and then sends the local parameter to the central server and gets the averaged model parameter from the central server. In this way, the number of communication rounds is reduced from $O(T)$ to $O(T/p)$ (T is the number of iterations). In addition, (Yu, Yang, and Zhu 2019) shows that, same as P-SGD, local SGD enjoys the same convergence rate $O(1/\sqrt{KT})$ with a linear speedup regarding the number of workers for non-convex problems. Therefore, local SGD is more efficient than the traditional P-SGD method, and it has been applied to different applications.

Even though local SGD can decrease the communication overhead by reducing the number of communication rounds, yet it still has large communication cost in each communication round for large models. For instance, the ResNet-152 (He et al. 2016) neural network for ImageNet is with the size of about 240MB, which is a large burden to the communication, especially for the FL system with limited communication capacity. To allievate the communication issue caused by the large model, a common strategy is to compress the gradient in each communication round, such as sparsifying gradients (Strom 2015; Lin et al. 2017; Aji and Heafield 2017) and quantizing gradients (Alistarh et al. 2017; Wen et al. 2017; Bernstein et al. 2018). In particular, gradient sparsification only sends the significant gradient elements, while gradient quantization is to quantize the full-precision gradient to the low-bit one. Both of them can significantly reduce the communication cost in each round. In addition, to reduce the large variance caused by gradient compression, the error compensation technique is usually used (Seide et al. 2014; Stich, Cordonnier, and Jaggi 2018; Stich and Karimireddy 2019; Tang et al. 2019; Zheng, Huang, and Kwok 2019). Previous studies (Tang et al. 2019;

Methods		Non-momentum	Momentum	UG	DC	EC	BC
$p=1$	DoubleSqueeze (Tang et al. 2019)	$O(T)$	-	✗	✓	✓	✓
	dist-EF-SGDM (Zheng, Huang, and Kwok 2019)	-	$O(T)$	✗	✓	✓	✓
$p>1$	Full-precision Local SGD (Yu, Jin, and Yang 2019)	$O(K^{3/2}T^{1/2})$	$O(K^{3/2}T^{1/2})$	✓	-	-	-
	FedPAQ (Reisizadeh et al. 2020)	$O(K^{3/2}T^{1/2})$	-	✓	✗	✗	✗
	Qsparse-local-SGD (Basu et al. 2019)	$O(K^{3/4}T^{3/4})$	-	✗	✗	✓	✓
	Ours	$O(K^{3/2}T^{1/2})$	$O(K^{3/2}T^{1/2})$	✓	✓	✓	✓

Table 1: The number of communication rounds of different methods for non-convex problems. UG denotes *unbounded gradient*. DC represents *double compression*. EC indicates *error compensation*. BC stands for *biased compression*. p is the communication period. T is the number of iterations. K is the number of workers.

Zheng, Huang, and Kwok 2019; Stich, Cordonnier, and Jaggi 2018) show that P-SGD with error-compensated gradient compression has the same convergence rate as the full-precision one. Inspired by that, one would like to know: *Is it possible to apply error-compensated gradient compression to local SGD without jeopardizing the convergence rate?* The recent study (Basu et al. 2019) tried to answer this question and proposed Qsparse-local-SGD to compress the gradient of local SGD. However, its result is suboptimal. Firstly, Qsparse-local-SGD only compresses the gradient sent from workers to the central server and keeps the full-precision gradient sent back to the worker. As a result, its communication overhead is still large. Secondly, the communication complexity given by Qsparse-local-SGD is suboptimal. It requires $O(K^{3/4}T^{3/4})$ communication rounds, while the full-precision local SGD only needs $O(K^{3/2}T^{1/2})$ communication rounds. Thirdly, Qsparse-local-SGD has a pathological assumption: The gradient norm is bounded, which cannot be satisfied in many cases (Khaled, Mishchenko, and Richtárik 2019). What's more, Qsparse-local-SGD only focuses on the regular local SGD method. However, momentum local SGD is more commonly used in practical applications and usually has better convergence and generalization performance (Yu, Jin, and Yang 2019). Thus, it is also necessary to reduce the communication overhead of the momentum local SGD.

In this paper, to apply compressed gradients to local SGD without jeopardizing the convergence rate and communication complexity, we propose two novel communication-efficient local SGD methods. In particular, to reduce the communication cost in each round, we propose to compress all gradients exchanged between the worker and server by the error-compensated compression mechanism. Under the non-convex setting, without the bounded gradient assumption in (Basu et al. 2019; Tang et al. 2019; Zheng, Huang, and Kwok 2019), our theoretical result shows that our proposed method admits the same convergence rate $O(1/\sqrt{KT})$ as the full-precision method. Note that it is more challenging to study the convergence rate without the bounded gradient assumption. In this paper, we addressed this issue by carefully bounding the compression error. As shown in Table 1, our method enjoys the same number of communication rounds $O(K^{3/2}T^{1/2})$ as the full-precision one, which is better than Qsparse-local-SGD. Moreover, we propose a new momentum local SGD with compressed gradients, which also enjoys the same convergence rate and communication complexity as the full-precision method. As far as we know, these theoretical results regarding the convergence rate and communication complexity are the first time to be discovered. At last, extensive experimental results confirmed the effectiveness of our proposed methods. Here, we summarize our contributions as follows:

- We propose a communication-efficient local SGD and a communication-efficient *momentum* local SGD. To the best of our knowledge, both of them are the first ones with double compression mechanism for (momentum) local SGD.
- We propose a new theoretical analysis strategy for analyzing the error-compensated double compression method without the bounded gradient assumption. As far as we know, this is the first work doing that.
- With our new theoretical analysis strategy, we carefully bound the compression error and disclose that our proposed communication-efficient (momentum) local SGD with error-compensated compression mechanism enjoys the same communication complexity with the full-precision method. This is the first work achieving this result.
- Extensive experimental results confirm the efficacy of our proposed two methods.

Related Works

In the distributed machine learning area, to reduce the communication overhead, there are two strategies: reducing the number of communication rounds and reducing the cost in each communication round.

Reduce Communication Rounds To reduce the number of communication rounds, local SGD updates model parameters locally for multiple iterations and then conducts the communication to synchronize the model parameter. Thus, local SGD can reduce the number of communication rounds by doing more local computation and less communication. Under the non-convex setting, the communication complexity is extensively studied in recent works (Jiang and Agrawal 2018; Yu, Jin, and Yang 2019; Yu, Yang, and Zhu 2019; Wang and Joshi 2018; Haddadpour et al. 2019). For instance, when different workers have identical data, (Yu, Yang, and Zhu 2019) theoretically proves that local SGD achieves the

convergence rate $O(1/\sqrt{KT})$ when the number of local updates is in the order of $O(K^{3/4}T^{3/4})$. In the recent work (Yu, Jin, and Yang 2019), the communication complexity is further improved to $O(K^{3/2}T^{1/2})$.

Compress Gradients To reduce the communication cost in each communication round, a commonly used strategy is to compress the gradient, such as sparsifying gradients and quantizing gradients. Although the compression of gradients can reduce the communication cost in each round, it causes large variance, hampering the convergence. To address this issue, (Seide et al. 2014) proposed the error compensation method to reduce the variance of the compressed gradient, which can stabilize the convergence. After that, the error compensation technique has been used in different methods (Stich, Cordonnier, and Jaggi 2018; Tang et al. 2019; Zheng, Huang, and Kwok 2019). For example, DoubleSqueeze (Tang et al. 2019) compresses gradients with error compensation in both passes of the regular P-SGD, and dist-EF-SGDM (Zheng, Huang, and Kwok 2019) applies the same strategy to parallel momentum SGD. However, all aforementioned methods only focus on the regular (momentum) SGD instead of local SGD. In the recent work (Basu et al. 2019), Qsparse-local-SGD applies the compressed gradient with error compensation to local SGD. However, it only conducts compression on the gradient from workers to the central server. Thus, it still has large communication overhead caused by the dense gradient from the central server to workers. Similar with Qsparse-local-SGD, (Reisizadeh et al. 2020; He et al. 2020) also only compresses the gradient sent from workers to the central server and use the full gradient from the central server to workers. Moreover, (Reisizadeh et al. 2020; He et al. 2020) require that the compressor for gradients should be unbiased and do not utilize the error-compensation mechanism. Thus, these two methods are supposed to have a worse convergence performance than those with the error-compensation mechanism (Tang et al. 2019; Karimireddy et al. 2019). More comparison results can be found in Table 1.

Convergence Analysis of Compressed Local SGD When analyzing the convergence rate and the communication complexity, Qsparse-local-SGD (Basu et al. 2019), as well as the compressed P-SGD method (Tang et al. 2019; Zheng, Huang, and Kwok 2019), assumes the gradient is bounded so that it is easy to bound the compression error. However, this assumption is pathological (Khaled, Mishchenko, and Richtárik 2019), which might not be true in some cases. (Reisizadeh et al. 2020; He et al. 2020) didn't use this assumption. However, these two methods require the compressor to be unbiased so that they do not need to bound the compression error, simplifying their theoretical analysis. On the contrary, our method uses the error-compensated compressor so that the compressed gradient in our method can be biased. We need to carefully bound the compression error in both directions without the bounded gradient assumption, which is much more challenging than all existing works.

Preliminaries

Throughout this paper, we have the following assumptions which are commonly used in federated learning (Yu, Jin, and Yang 2019; Liang et al. 2019).

Assumption 1. ***Smoothness**: All local functions $f^{(k)}(\cdot)$ are L-smooth, i.e.,*

$$\|\nabla f^{(k)}(x) - \nabla f^{(k)}(y)\| \leq L\|x-y\|, \quad \forall k, \forall x, \forall y\ . \quad (2)$$

Assumption 2. ***Bounded gradient variance within each worker**: There exists $\sigma > 0$ such that*

$$\mathbb{E}[\|\nabla F^{(k)}(x;\xi) - \nabla f^{(k)}(x)\|^2] \leq \sigma^2, \quad \forall k, \forall x\ . \quad (3)$$

These two assumptions are very common in existing literature. Note that we do not assume the bounded gradient $\mathbb{E}[\|\nabla F^{(k)}(x;\xi)\|^2] \leq G^2$ for a constant G. In this paper, as Qsparse-loal-SGD (Basu et al. 2019), different workers have identical data. We summarize the notations used in this paper. $x_t^{(k)}$ denotes the model parameter of the k-th worker at the t-th iteration. $x_t = \frac{1}{K}\sum_{k=1}^K x_t^{(k)}$ denotes the averaged model parameter across all workers at the t-th iteration. $\nabla F^{(k)}(x_t^{(k)};\xi_t^{(k)})$ represents the stochastic gradient of the k-th worker which is computed on the local model arameter and data at the t-th iteration. $\nabla f^{(k)}(x_t^{(k)}) = \mathbb{E}_{\xi\sim\mathcal{D}^{(k)}}\nabla F^{(k)}(x_t^{(k)};\xi_t^{(k)})$ represents the full gradient of the k-th worker at the t-th iteration. $\nabla f(x_t) = \frac{1}{K}\sum_{k=1}^K \nabla f^{(k)}(x_t^{(k)})$ represents the averaged gradient at the t-th iteration. $m_t^{(k)}$ indicates the momentum of the k-th worker at the t-th iteration. $m_t = \frac{1}{K}\sum_{k=1}^K m_t^{(k)}$ indicates the averaged momentum across all worker at the t-th iteration. p stands for the communication period. f^* denotes minimum of the loss function.

Communication-Efficient Distributed SGD with Compressed Gradients

Local SGD with Compressed Gradient

Our proposed local SGD with compressed gradients is presented in Algorithm 1. The core idea is to compress the gradient sent from workers to the central server and that sent from the central server to workers with a δ-contraction operator, which is defined as follows:

Definition 1. *(Stich, Cordonnier, and Jaggi 2018) A compression operator $Q: \mathcal{R}^d \to \mathcal{R}^d$ is a δ-contraction operator if it satisfies the following property*

$$\|x - Q(x)\|^2 \leq (1-\delta)\|x\|^2\ , \quad (4)$$

where $0 < \delta \leq 1$.

A typical example is the top-k sparsification operator which only selects the k elements with the largest absolute values from the vector $x \in \mathcal{R}^d$.

To alleviate the large variance of compressed gradients, we compensate the compressed gradient by the residual error between the full-precision gradient and the compressed one. Specifically, each worker keeps the local model parameter $x_t^{(k)}$ and the local residual error $e_t^{(k)}$. When mod$(t+$

Algorithm 1 Local SGD with Compressed Gradients

Initialization: $x_0^{(k)} = x_0$, $e_0^{(k)} = 0$, $e_0 = 0$, $p \geq 1$, $\eta > 0$.
for $t = 0, \cdots, T-1$ **do**
  Worker-k:
  $x_{t+1}^{(k)} = x_t^{(k)} - \eta \nabla F^{(k)}(x_t^{(k)}; \xi_t^{(k)})$, $e_{t+1}^{(k)} = e_t^{(k)}$
  if mod$(t+1, p)$=0 **then**
    $v_{t+1}^{(k)} = x_{t+1-p}^{(k)} - x_{t+1}^{(k)} + e_t^{(k)}$
    $e_{t+1}^{(k)} = v_{t+1}^{(k)} - Q(v_{t+1}^{(k)})$
    send $Q(v_{t+1}^{(k)})$ to the server
  end if
  Server:
  $e_{t+1} = e_t$
  if mod$(t+1, p)$=0 **then**
    $v_{t+1} = \frac{1}{K}\sum_{k=1}^{K} Q(v_{t+1}^{(k)}) + e_t$
    $e_{t+1} = v_{t+1} - Q(v_{t+1})$
    Broadcast $Q(v_{t+1})$ to all workers
  end if
  Worker-k:
  if mod$(t+1, p)$=0 **then**
    $x_{t+1}^{(k)} = x_{t+1-p}^{(k)} - Q(v_{t+1})$
  end if
end for

$1, p) \neq 0$, each worker updates the local model parameter based on the local data as follows:

$$x_{t+1}^{(k)} = x_t^{(k)} - \eta \nabla F^{(k)}(x_t^{(k)}; \xi_t^{(k)}) \,, \tag{5}$$

where η is the step size. When mod$(t+1, p) = 0$, each worker computes the error-compensated gradient as follows:

$$v_{t+1}^{(k)} = x_{t+1-p}^{(k)} - x_{t+1}^{(k)} + e_t^{(k)} \,, \tag{6}$$

where $x_{t+1-p}^{(k)} - x_{t+1}^{(k)}$ represents the accumulated gradient in the past p iterations. Then, each worker compresses the local gradient $v_{t+1}^{(k)}$ by using the compression operator $Q(\cdot)$, such as the top-k sparsification operator. Meanwhile, each worker updates its local residual error as follows:

$$e_{t+1}^{(k)} = v_{t+1}^{(k)} - Q(v_{t+1}^{(k)}) \,, \tag{7}$$

where $Q(v_{t+1}^{(k)})$ represents the compressed gradient. After that, workers send the compressed gradient $Q(v_{t+1}^{(k)})$ to the server.

As for the server, it receives all compressed gradients $Q(v_{t+1}^{(k)})$ from workers and computes the global error-compensated gradient as follows:

$$v_{t+1} = \frac{1}{K}\sum_{k=1}^{K} Q(v_{t+1}^{(k)}) + e_t \,, \tag{8}$$

where e_t denotes the global residual error. Note that, e_t is not the average of local residual errors. Instead, the server constructs the global residual error by computing the difference between the global gradient v_{t+1} and its compressed version $Q(v_{t+1})$ as follows:

$$e_{t+1} = v_{t+1} - Q(v_{t+1}) \,. \tag{9}$$

After that, the server broadcasts $Q(v_{t+1})$ to all workers and they reset the local model parameter as follows:

$$x_{t+1}^{(k)} = x_{t+1-p}^{(k)} - Q(v_{t+1}) \,. \tag{10}$$

In summary, all the gradient communicated between workers and the central server are compressed. Therefore, the communication cost in each round is pretty small. On the contrary, the existing works (Basu et al. 2019; He et al. 2020; Reisizadeh et al. 2020) only compress the gradient from local workers to the central server while keeping the full-precision gradient from the central server to local workers. Moreover, compared with (Tang et al. 2019) which performs communication at every iteration, our method conducts communication at every p iterations. Thus, our method has much fewer communication rounds. In the following, we present the convergence rate and communication complexity of our proposed Algorithm 1.

Theorem 1. *Under Assumption 1 and 2, if we choose* $\eta \leq \frac{\sqrt{1+2(a_1+a_2+a_3)p^2}-1}{(a_1+a_2+a_3)p^2 L}$ *where* $a_1 = \frac{384(2-\delta)(1-\delta)}{\delta^4}$, $a_2 = \frac{48(1-\delta)}{\delta^2}$, *and* $a_3 = 16$, *we have*

$$\begin{aligned}\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}[\|\nabla f(x_t)\|^2] \leq{}& \frac{4(f(x_0)-f^*)}{\eta T} + \frac{2\eta\sigma^2 L}{K} \\ &+ \frac{384p\eta^2\sigma^2L^2(2-\delta)(1-\delta)}{\delta^4} + \frac{48p\eta^2\sigma^2L^2(1-\delta)}{\delta^2} \\ &+ 16p\eta^2\sigma^2L^2 \,.\end{aligned} \tag{11}$$

From Theorem 1, it can be seen that the last two terms on the RHS is related to the compression of gradients.

Corollary 1. *Under Assumption 1 and 2, by choosing* $\eta = \frac{\sqrt{K}}{\sqrt{T}}$ *and* $p \leq \frac{1}{L}\frac{T^{1/2}}{K^{3/2}} = O(\frac{T^{1/2}}{K^{3/2}})$, *we have*

$$\begin{aligned}\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}[\|\nabla f(x_t)\|^2] \leq{}& \frac{4(f(x_0)-f^*)+18\sigma^2L}{\sqrt{KT}} \\ &+ \frac{1}{\sqrt{KT}}\Big(\frac{384\sigma^2L(2-\delta)(1-\delta)}{\delta^4} + \frac{48\sigma^2L(1-\delta)}{\delta^2}\Big) \,.\end{aligned} \tag{12}$$

From Corollary 1, we can see that our method has the convergence rate as $O(\frac{1}{\sqrt{KT}})$, indicating a linear speedup regarding the number of workers K as the full-precision local SGD method (Yu, Jin, and Yang 2019). Moreover, it can be seen that the communication complexity of our method is $O(K^{3/2}T^{1/2})$, which is also same as the full-precision method. On the contrary, Qsparse-local-SGD (Basu et al. 2019) method has the communication complexity as large as $O(K^{3/4}T^{3/4})$. Hence, our method is more communication-efficient.

Momentum Local SGD with Compressed Gradient

The momentum technique is widely used for training machine learning models, especially deep neural networks. Recently, (Yu, Jin, and Yang 2019) proposed the local SGD with momentum method, which can be used for Federated

Learning. However, this method has large communication overhead. In addition, existing works (Basu et al. 2019; He et al. 2020; Reisizadeh et al. 2020) didn't consider the momentum method. Thus, it is still unclear whether the momentum local SGD method can use the double compressed gradient with the error-compensated mechanism and preserve the convergence rate and communication rounds as the full-precision one. To address this issue, we propose the momentum local SGD with the error-compensated double compression mechanism, which is summarized in Algorithm 2.

Algorithm 2 Momentum Local SGD with Compressed Gradients

Initialization: $x_0^{(k)} = x_0$, $e_0^{(k)} = 0$, $e_0 = 0$, $m_0^{(k)} = 0$, $p \geq 1$, $\eta > 0$, $\mu > 0$.
for $t = 0, \cdots, T-1$ **do**
  Worker-k:
  $m_{t+1}^{(k)} = \mu m_t^{(k)} + \nabla F^{(k)}(x_t^{(k)}; \xi_t^{(k)})$
  $x_{t+1}^{(k)} = x_t^{(k)} - \eta m_{t+1}^{(k)}$, $e_{t+1}^{(k)} = e_t^{(k)}$
  if mod$(t+1, p)$=0 **then**
    $v_{t+1}^{(k)} = x_{t+1-p}^{(k)} - x_{t+1}^{(k)} + e_t^{(k)}$
    $e_{t+1}^{(k)} = v_{t+1}^{(k)} - Q(v_{t+1}^{(k)})$
    send $Q(v_{t+1}^{(k)})$, $m_{t+1}^{(k)}$ to the server
  end if
  Server:
  $e_{t+1} = e_t$
  if mod$(t+1, p)$=0 **then**
    $v_{t+1} = \frac{1}{K}\sum_{k=1}^{K} Q(v_{t+1}^{(k)}) + e_t$
    $m_{t+1} = \frac{1}{K}\sum_{k=1}^{K} m_{t+1}^{(k)}$
    $e_{t+1} = v_{t+1} - Q(v_{t+1})$
    Broadcast $Q(v_{t+1})$, m_{t+1} to all workers
  end if
  Worker-k:
  if mod$(t+1, p)$=0 **then**
    $x_{t+1}^{(k)} = x_{t+1-p}^{(k)} - Q(v_{t+1})$, $m_{t+1}^{(k)} = m_{t+1}$
  end if
end for

For each worker, when mod$(t+1, p) \neq 0$, it updates the local model parameter based on its local data as follows:

$$m_{t+1}^{(k)} = \mu m_t^{(k)} + \nabla F^{(k)}(x_t^{(k)}; \xi_t^{(k)}) \, , x_{t+1}^{(k)} = x_t^{(k)} - \eta m_{t+1}^{(k)} \, , \tag{13}$$

where $m_{t+1}^{(k)}$ denotes the momentum, $0 < \mu < 1$ represents the momentum coefficient, and $\eta > 0$ is the step size. When mod$(t+1, p) = 0$, instead of compressing the gradient, each worker compresses the momentum as follows:

$$v_{t+1}^{(k)} = x_{t+1-p}^{(k)} - x_{t+1}^{(k)} + e_t^{(k)} \, , e_{t+1}^{(k)} = v_{t+1}^{(k)} - Q(v_{t+1}^{(k)}) \, , \tag{14}$$

where $x_{t+1-p}^{(k)} - x_{t+1}^{(k)}$ represents the accumulated "gradient"[1] in the past p iterations, $v_{t+1}^{(k)}$ is the error-compensated gradient, $Q(v_{t+1}^{(k)})$ denotes the compressed gradient, and $e_{t+1}^{(k)}$ stands for the local residual error.

For the server, it computes the error-compensated gradient and compresses it as follows:

$$v_{t+1} = \frac{1}{K}\sum_{k=1}^{K} Q(v_{t+1}^{(k)}) + e_t \, , e_{t+1} = v_{t+1} - Q(v_{t+1}) \, . \tag{15}$$

After that, the server broadcasts the compressed gradient to all workers. Then, all workers update the local model parameter based on the received gradient. In the following, we present the convergence rate and communication complexity of our proposed Algorithm 2.

Theorem 2. *Under Assumption 1 and 2, if we choose* $\eta \leq \frac{\sqrt{b^2+2(a_1+a_2+a_3+a_4)p^2}-b}{(a_1+a_2+a_3+a_4)p^2L}$ *where* $a_1 = \frac{1536(2-\delta)(1-\delta)}{\delta^4(1-\mu)^2}\Big(1 + \frac{1}{(1-\mu)^2}\Big)$, $a_2 = \frac{192(1-\delta)}{\delta^2(1-\mu)^2}\Big(1 + \frac{1}{(1-\mu)^2}\Big)$, $a_3 = \frac{16}{(1-\mu)^2}$, $a_4 = \frac{12}{(1-\mu)^4}$, *and* $b = \frac{1}{1-\mu}$ *we have*

$$\begin{aligned}\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}[\|\nabla f(x_t)\|^2] &\leq \frac{4(1-\mu)(f(x_0)-f^*)}{\eta T} \\ &+ \frac{2\eta\sigma^2 L}{(1-\mu)K} + \frac{16p\eta^2\sigma^2L^2}{(1-\mu)^2} + \frac{12\eta^2\mu^2\sigma^2L^2}{(1-\mu)^4K} \\ &+ \frac{1536p\eta^2\sigma^2L^2(2-\delta)(1-\delta)}{\delta^4(1-\mu)^2}\Big(1 + \frac{1}{(1-\mu)^2}\Big) \\ &+ \frac{192p\eta^2\sigma^2L^2(1-\delta)}{\delta^2(1-\mu)^2}\Big(1 + \frac{1}{(1-\mu)^2}\Big) \, .\end{aligned} \tag{16}$$

Similar with Theorem 1, compared with the full-precision momentum local SGD (Yu, Jin, and Yang 2019), our convergence result in Theorem 2 has extra terms with respect to δ which is caused by the compression of gradients.

Corollary 2. *Under Assumption 1 and 2, by choosing* $\eta = \frac{\sqrt{K}}{\sqrt{T}}$ *and* $p \leq \frac{1}{L}\frac{T^{1/2}}{K^{3/2}} = O(\frac{T^{1/2}}{K^{3/2}})$, *we have*

$$\begin{aligned}\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}[\|\nabla f(x_t)\|^2] &\leq \frac{4(1-\mu)(f(x_0)-f^*)}{\sqrt{KT}} \\ &+ \frac{2\sigma^2 L}{(1-\mu)\sqrt{KT}} + \frac{16\sigma^2L}{(1-\mu)^2\sqrt{KT}} + \frac{12\mu^2\sigma^2L^2}{(1-\mu)^4\sqrt{KT}} \\ &+ \frac{1}{\sqrt{KT}}\frac{1536\sigma^2L(2-\delta)(1-\delta)}{\delta^4(1-\mu)^2}\Big(1 + \frac{1}{(1-\mu)^2}\Big) \\ &+ \frac{1}{\sqrt{KT}}\frac{192\sigma^2L(1-\delta)}{\delta^2(1-\mu)^2}\Big(1 + \frac{1}{(1-\mu)^2}\Big) \, .\end{aligned} \tag{17}$$

From Corollary 2, it can be seen that our method still enjoys the linear speedup regarding the number of workers as the full-precision momentum local SGD method (Yu, Jin, and Yang 2019). As for the communication complexity $O(K^{3/2}T^{1/2})$, it is also the same as the full-precision method (Yu, Jin, and Yang 2019). Therefore, compared with the full-precision method, our method has the same convergence rate and communication complexity but has much

[1] We call it gradient rather than momentum for convenience.

smaller communication cost in each round due to the gradient compression. Note that, in our theoretical analysis, we follow the i.i.d. setting of Qsparse-local-sgd to improve the communication complexity. But under some assumptions regarding the non i.i.d. distribution, such as the bounded gradient heterogeneity assumption, it is not difficult to handle the heterogeneous gradient and then we can apply our proof technique to the non i.i.d. case smoothly.

Communication period	Local SGD (%)	Alg1 (%)
1	92.68	92.76
4	92.65	92.62
8	92.47	92.58
16	92.86	92.42

Table 2: Top-1 Test Accuracy of ResNet-56 on CIFAR-10.

Communication period	Local SGDM (%)	Alg2 (%)
1	76.04	75.88
4	75.81	76.12
8	76.01	76.06
16	76.01	75.94

Table 3: Top-1 Test Accuracy of ResNet-50 on ImageNet.

Experiments

Experimental Settings

All experiments are implemented in PyTorch (Paszke et al. 2019) and run on a cluster with NVIDIA Tesla P40 GPUs, where nodes are interconnected by a network with 40 Gbps bandwidth. We run each distributed training experiment using 8 workers (GPUs). The compression method used in our experiments is the top-10% sparsification operator. Two benchmark datasets are used in our experiments. The details are described as follows.

CIFAR-10: We test ResNet-56 (He et al. 2016) with all the above mentioned algorithms on CIFAR-10 dataset (Krizhevsky, Hinton et al. 2009). Common data augmentation techniques such as random cropping, random flipping and standardization are performed. The base learning rate is 0.1, the weight decay is 5×10^{-4} and the total batch size is 128. For local SGD, the model is trained for 150 epoch in total, with a learning rate decay of 0.1 at epoch 100. For momentum local SGD, the model is trained for 200 epoch in total, with a learning rate decay of 0.1 at epoch 100 and 150.

ImageNet: We test ResNet-50 (He et al. 2016) on ImageNet dataset (Russakovsky et al. 2015) [2]. Similar data augmentation techniques are performed. The model is trained for 90 epoch in total, with a learning rate decay of 0.1 at epoch 30 and 60. The base learning rate is 0.1, the weight decay is 1×10^{-4} and the total batch size is 256.

Results of Methods without Momentum

To verify the performance of our proposed method in Algorithm 1 (Alg1), we compare our method with the full-precision local SGD method, DoubleSqueeze (Tang et al. 2019), and Qsparse-local-SGD (Basu et al. 2019). Note that we didn't compare our method with (He et al. 2020; Reisizadeh et al. 2020) since existing works (Tang et al. 2019; Karimireddy et al. 2019) have shown that this kind of methods without the error-compensation mechanism has worse convergence and generalization performance. Figure 1(a) and 1(b) show the training loss and testing accuracy regarding the number of epochs under different communication periods p on CIFAR-10 dataset. Table 2 reports the final testing accuracy of ResNet-56 on CIFAR10. From them, we have the following observations. First, comparing with the full-precision method, our method with compressed gradients has similar convergence performance: converging to almost the same value and achieving almost the same accuracy. Hence, although our method uses the compressed gradient, yet the convergence performance does not degenerate. Second, when the communication period p is 1, local-p1 becomes the traditional parallel SGD (P-SGD) method, and our method is equivalent to DoubleSqueeze (Tang et al. 2019) method. Comparing with these two state-of-the-art methods, when increasing p, our method converges to almost the same value and achieves almost the same accuracy as P-SGD and DoubleSqueeze. Furthermore, we also plot the training loss and testing accuracy regarding the communication cost in Figure 1(c) and 1(d). It can be seen that our method with compressed gradients has much less communication cost and achieves almost the same performance comparing with the full-precision methods. Based on these observations, we can conclude that our method does not jeopardize the final model performance even though it compresses the gradient and reduces the number of communication rounds.

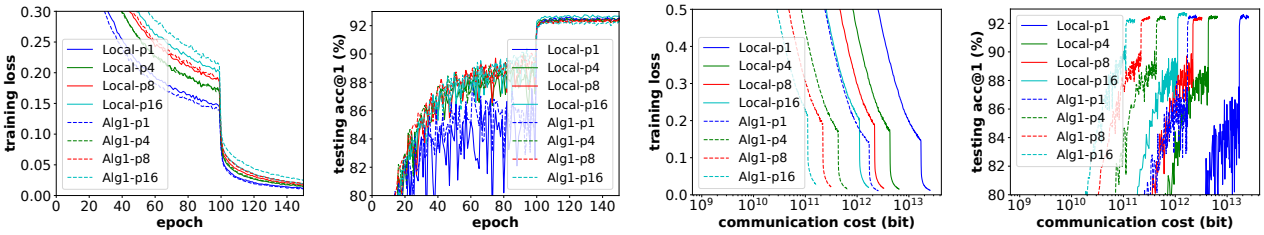
Moreover, to further verify the performance of our proposed method, we compare it with Qsparse-local-SGD (QL-SGD) which only conducts compression on the gradient sent from workers to the server. Note that, to make a fair comparison, we employ the same compression operator: top-10% sparsification. From Figure 2, it can be seen that our method has similar convergence performance with QL-SGD, but has much less communication cost since our method conducts compression on the gradient sent from workers to the server and the averaged gradient sent from the server to workers. Therefore, our method is more communication-efficient than QL-SGD.

Results of Methods with Momentum

In this experiment, we compare our proposed Algorithm 2 (Alg2) with the full-precision momentum local SGD (local SGDM) (Yu, Jin, and Yang 2019) and dist-EF-SGDM (Zheng, Huang, and Kwok 2019). Note that we didn't compare it with Qsparse-local-SGD (Basu et al. 2019) because this method only studied the regular gradient rather than the momentum case. Throughout this experiment, the momentum coefficient μ is set to 0.9.

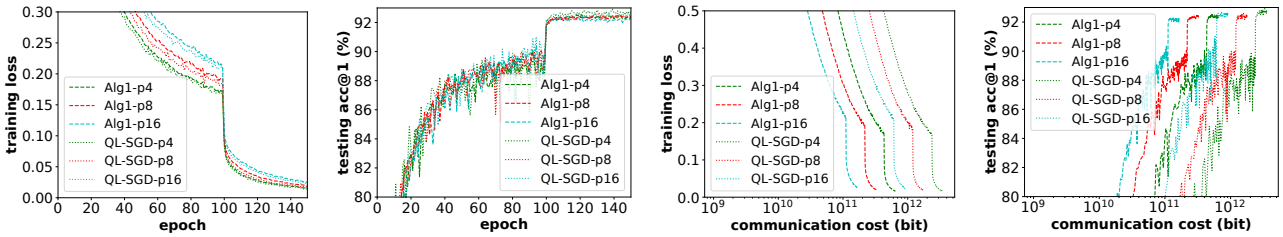
In Figure 3(a) and 3(b), we report the convergence performance regarding the number of epochs on ImageNet. In Figure 3(c) and 3(d), we show the convergence result with respect to the communication cost. In addition, the testing

[2] Since ResNet-50 on ImageNet is trained with momentum SGD in (He et al. 2016), here we only use Algorithm 2 for ImageNet.



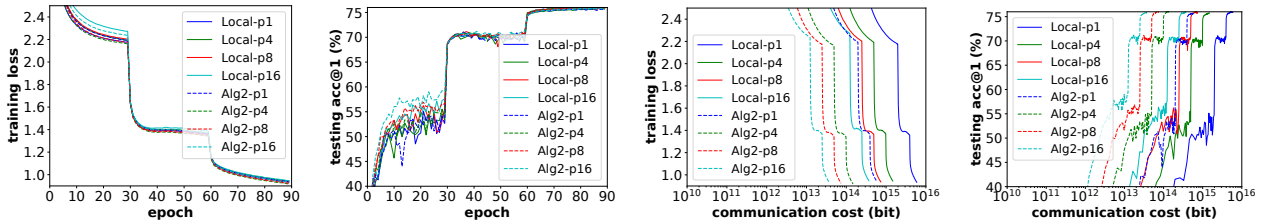
(a) Loss v.s. epochs (b) Acc v.s. epochs (c) Loss v.s. bits (d) Acc v.s. bits

Figure 1: ResNet56@CIFAR10 with Alg1: The training loss and testing accuracy regarding epochs and communication cost. Alg1-p1 is equivalent to DoubleSqueeze.



(a) Loss v.s. epochs (b) Acc v.s. epochs (c) Loss v.s. bits (d) Acc v.s. bits

Figure 2: ResNet56@CIFAR10: The training loss and testing accuracy of Alg1 and QL-SGD.



(a) Loss v.s. epochs (b) Acc v.s. epochs (c) Loss v.s. bits (d) Acc v.s. bits

Figure 3: ResNet50@ImageNet with Alg2: The training loss and testing accuracy regarding epochs and communication cost. Here, local denotes local SGDM. Alg2-p1 is equivalent to dist-EF-SGDM.

accuracy of the learned model is reported in Table 3. Similar to Algorithm 1, from Figure 3(a) and 3(b), we can also find that the model optimized by our method has similar convergence behavior with that learned by the full-precision method. In terms of the final testing accuracy in Table 3, our method and the full-precision local SGD achieve similar performance. It shows that compressed gradient communication is not only beneficial for accelerating the convergence regarding training epochs, but also do not jeopardize the final model performance.

From Figure 3(c) and 3(d), it can be seen that increasing the communication period naturally reduces the communication cost. Compared with local SGD using the same communication period, our method needs much less communication cost, which is most beneficial for the communication constrained federated learning scenario where the uplink and downlink bandwidth between the central server and user devices may be very limited. Compared with dist-EF-SGDM, our method also has much less communication cost but almost the same accuracy, which further confirms our method is efficient in communication and correct in computation.

Conclusions

In this paper, we propose two new communication-efficient distributed SGD methods for Federated Learning. Specifically, to reduce the communication cost of local SGD, we compress the gradient exchanged between the worker and server with the error-compensated compression operator. Our theoretical results show that our proposed methods enjoy the same iteration complexity and communication complexity as the full-precision method. Extensive experimental results have verified that our methods have similar convergence performance as the full-precision method but require much less communication budget.

Acknowledgements

This work was partially supported by NSF IIS 1845666, 1852606, 1838627, 1837956, 1956002, 2040588.

References

Aji, A. F.; and Heafield, K. 2017. Sparse communication for distributed gradient descent. *arXiv preprint arXiv:1704.05021* .

Alistarh, D.; Grubic, D.; Li, J.; Tomioka, R.; and Vojnovic, M. 2017. QSGD: Communication-efficient SGD via gradient quantization and encoding. In *Advances in Neural Information Processing Systems*, 1709–1720.

Basu, D.; Data, D.; Karakus, C.; and Diggavi, S. 2019. Qsparse-local-SGD: Distributed SGD with Quantization, Sparsification and Local Computations. In *Advances in Neural Information Processing Systems*, 14668–14679.

Bernstein, J.; Wang, Y.-X.; Azizzadenesheli, K.; and Anandkumar, A. 2018. signSGD: Compressed optimisation for non-convex problems. *arXiv preprint arXiv:1802.04434* .

Haddadpour, F.; Kamani, M. M.; Mahdavi, M.; and Cadambe, V. 2019. Local SGD with periodic averaging: Tighter analysis and adaptive synchronization. In *Advances in Neural Information Processing Systems*, 11080–11092.

He, C.; Li, S.; So, J.; Zhang, M.; Wang, H.; Wang, X.; Vepakomma, P.; Singh, A.; Qiu, H.; Shen, L.; et al. 2020. Fedml: A research library and benchmark for federated machine learning. *arXiv preprint arXiv:2007.13518* .

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 770–778.

Jiang, P.; and Agrawal, G. 2018. A linear speedup analysis of distributed deep learning with sparse and quantized communication. In *Advances in Neural Information Processing Systems*, 2525–2536.

Karimireddy, S. P.; Rebjock, Q.; Stich, S. U.; and Jaggi, M. 2019. Error feedback fixes signsgd and other gradient compression schemes. *arXiv preprint arXiv:1901.09847* .

Khaled, A.; Mishchenko, K.; and Richtárik, P. 2019. First analysis of local gd on heterogeneous data. *arXiv preprint arXiv:1909.04715* .

Krizhevsky, A.; Hinton, G.; et al. 2009. Learning multiple layers of features from tiny images. Technical report, Citeseer.

Liang, X.; Shen, S.; Liu, J.; Pan, Z.; Chen, E.; and Cheng, Y. 2019. Variance Reduced Local SGD with Lower Communication Complexity. *arXiv preprint arXiv:1912.12844* .

Lin, Y.; Han, S.; Mao, H.; Wang, Y.; and Dally, W. J. 2017. Deep gradient compression: Reducing the communication bandwidth for distributed training. *arXiv preprint arXiv:1712.01887* .

Paszke, A.; Gross, S.; Massa, F.; Lerer, A.; Bradbury, J.; Chanan, G.; Killeen, T.; Lin, Z.; Gimelshein, N.; Antiga, L.; et al. 2019. PyTorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, 8024–8035.

Reisizadeh, A.; Mokhtari, A.; Hassani, H.; Jadbabaie, A.; and Pedarsani, R. 2020. Fedpaq: A communication-efficient federated learning method with periodic averaging and quantization. In *International Conference on Artificial Intelligence and Statistics*, 2021–2031.

Russakovsky, O.; Deng, J.; Su, H.; Krause, J.; Satheesh, S.; Ma, S.; Huang, Z.; Karpathy, A.; Khosla, A.; Bernstein, M.; Berg, A. C.; and Fei-Fei, L. 2015. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)* 115(3): 211–252. doi:10.1007/s11263-015-0816-y.

Seide, F.; Fu, H.; Droppo, J.; Li, G.; and Yu, D. 2014. 1-bit stochastic gradient descent and its application to data-parallel distributed training of speech dnns. In *Fifteenth Annual Conference of the International Speech Communication Association*.

Stich, S. U.; Cordonnier, J.-B.; and Jaggi, M. 2018. Sparsified SGD with memory. In *Advances in Neural Information Processing Systems*, 4447–4458.

Stich, S. U.; and Karimireddy, S. P. 2019. The error-feedback framework: Better rates for SGD with delayed gradients and compressed communication. *arXiv preprint arXiv:1909.05350* .

Strom, N. 2015. Scalable distributed DNN training using commodity GPU cloud computing. In *Sixteenth Annual Conference of the International Speech Communication Association*.

Tang, H.; Lian, X.; Zhang, T.; and Liu, J. 2019. Doublesqueeze: Parallel stochastic gradient descent with double-pass error-compensated compression. *arXiv preprint arXiv:1905.05957* .

Wang, J.; and Joshi, G. 2018. Cooperative SGD: A unified framework for the design and analysis of communication-efficient SGD algorithms. *arXiv preprint arXiv:1808.07576* .

Wen, W.; Xu, C.; Yan, F.; Wu, C.; Wang, Y.; Chen, Y.; and Li, H. 2017. Terngrad: Ternary gradients to reduce communication in distributed deep learning. In *Advances in neural information processing systems*, 1509–1519.

Yu, H.; Jin, R.; and Yang, S. 2019. On the linear speedup analysis of communication efficient momentum sgd for distributed non-convex optimization. *arXiv preprint arXiv:1905.03817* .

Yu, H.; Yang, S.; and Zhu, S. 2019. Parallel restarted SGD with faster convergence and less communication: Demystifying why model averaging works for deep learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, 5693–5700.

Zheng, S.; Huang, Z.; and Kwok, J. 2019. Communication-efficient distributed blockwise momentum SGD with error-feedback. In *Advances in Neural Information Processing Systems*, 11450–11460.