Exploring Fast and Communication-Efficient Algorithms in Large-scale Distributed Networks

Yue Yu
IIIS, Tsinghua University

Jiaxiang Wu
Tencent AI Lab

Junzhou Huang
University of Texas at Arlington

Abstract

The communication overhead has become a significant bottleneck in data-parallel network with the increasing of model size and data samples. In this work, we propose a new algorithm LPC-SVRG with quantized gradients and its acceleration ALPC-SVRG to effectively reduce the communication complexity while maintaining the same convergence as the unquantized algorithms. Specifically, we formulate the heuristic gradient clipping technique within the quantization scheme and show that unbiased quantization methods in related works [3, 33, 38] are special cases of ours. We introduce *double sampling* in the accelerated algorithm ALPC-SVRG to fully combine the gradients of full-precision and low-precision, and then achieve acceleration with fewer communication overhead. Our analysis focuses on the nonsmooth composite problem, which makes our algorithms more general. The experiments on linear models and deep neural networks validate the effectiveness of our algorithms.

1 INTRODUCTION

Recent years has witnessed data explosion and increasing model complexity in machine learning. It becomes difficult to handle all the massive data and large-scale models within one machine. Therefore, large-scale distributed optimization receives growing attention [10, 1, 9, 36]. By exploiting multiple workers, it can remarkably reduce computation time. Distributed data-parallel network is a commonly used large-scale framework which contains N workers and each worker keeps a copy of model parameters. At each iteration, all workers compute their local gradients and communicate gradients with peers to obtain global gradients, and update model parameters. As the number of workers increases, the computation time (for a mini-batch of the same size) can be dramatically reduced, however, the communication cost rises. It has been observed in many distributed learning systems that the communication cost has become the performance bottleneck [7, 28, 30].

To improve the communication efficiency in data-parallel network, generally, there are two orthogonal methods, i.e., gradient quantization and gradient sparsification. Researches on quantization focus on employing low-precision and fewer bits representation of gradients rather than full-precision with 32 bits[1] [28, 34, 38, 33, 3]. And for works on sparsification, they design dropping out mechanisms for gradients to reduce the communication complexity [32, 2].

For most gradient quantization based methods, the unbiased stochastic quantization is adopted to compress gradients into their low-precision counterparts. However, it has been observed in many applications that such unbiased quantization brings larger precision loss or quantization error [33], and therefore leads to lower accuracy. On the other hand, several works [33, 16, 24] reported the heuristic *gradient clipping* technique can effectively improve convergence. However, this breaks the unbiasedness property and no theoretical analysis has been given so far.

In our work, we propose the LPC-SVRG algorithm to embed gradient quantization and clipping techniques into SVRG [15]. Furthermore, we present its acceleration variant, the ALPC-SVRG algorithm. These two algorithms adopt the variance reduction idea to reduce the gradient variance as the algorithm converges, so as to achieve a faster convergence rate. To reduce the communication overhead, we introduce a new quantization scheme which integrates gradient clipping, and provide its theoretical analysis. Furthermore, we propose *double sampling* to fully take advantage of both low-precision and full-precision representations, and achieve lower communication complexity and fast convergence rate at the same time. Detailed contributions are summarized as follows.

Contribution. We consider the following finite-sum com-

Proceedings of the 22nd International Conference on Artificial Intelligence and Statistics (AISTATS) 2019, Naha, Okinawa, Japan. PMLR: Volume 89.

[1] In this paper, we assume that a floating-point number is stored using 32 bits even though it can be 64 bits in many modern systems.

posite minimization problem:

$$\min_{x\in\mathbb{R}^d} P(x) = f(x) + h(x) = \frac{1}{n}\sum_{i=1}^{n} f_i(x) + h(x), \quad (1)$$

where $f(x)$ is an average of n smooth functions $f_i(x)$, and $h(x)$ is a convex but can be nonsmooth function. d is the model dimension. Such formulation generalizes many applications in machine learning and statistics, such as logistic regression and deep learning models.

We try to solve (1) in a data-parallel network with N workers, and analyze the convergence behavior when low-precision quantized gradients are adopted. Our analysis covers both convex and nonconvex objectives, specifically:

(1) We integrate gradient clipping into quantization to reduce the overhead of communication, and show that the unbiased quantization methods adopted in [3, 33, 38] are special cases of our scheme. We also mathematically analyze the quantization error given gradient clipping;

(2) Based on such quantization scheme, we propose LPC-SVRG to solve nonconvex and nonsmooth objective (1). We prove the same convergence rate can be achieved even the numbers are represented by much fewer $O(\log\sqrt{d})$ bits (compared to 32);

(3) We propose an accelerated algorithm ALPC-SVRG based on Katyusha momentum [4]. With *double sampling*, we are able to combine updates both from quantized gradients and full-precision gradients to achieve fast convergence without increasing the communication overhead.

(4) We conduct extensive experiments on both linear regression and deep learning models to validate the effectiveness of our methods.

2 RELATED WORK

Many literatures focus on designing fast and efficient algorithms for large-scale distributed systems [1, 5, 9, 14]. Asynchronous algorithms with stale gradients are also extensively studied [25, 19, 10, 18], which are orthogonal to our work. Researches on reducing communication complexity in data-parallel network can be generally divided into two categories: gradient sparsification and gradient quantization.

Gradient sparsification. [2] introduced a dropping out mechanism where only gradients exceed a threshold being transmitted. [32] formulated the sparsification problem into a convex optimization to minimize the gradient variance. Several heuristic methods such as momentum correction, local gradient clipping and momentum factor masking were adopted in [21] to compensate the error induced by gradient sparsification. [31] proposed a method for atomic sparsification of gradients while minimizing variance.

Gradient quantization. [28] quantized gradients to {-1,1} using zero-thresholding and used an error-feedback scheme to compensate the quantization error. Similar technique was also adopted in [34], where original unbiased quantization method was used to quantize the error-compensated gradients. [33] adopted a 3-level representation, i.e. {-1, 0, 1}, for each element of gradient. They empirically showed the accuracy improvement when gradient clipping was applied, while no theoretical analysis was provided. [3] used an unbiased stochastic quantization method to quantize gradients into s-level. They provided the convergence property of unbiased low-precision SVRG [15] for strongly convex and smooth problems. [8] proposed a low-precision variance reduction method for training in a single machine, with quantized model parameters. An end-to-end low-precision mechanism was also setup in [38, 39], where data samples, models and gradients were all quantized. [38] analyzed the convergence property of an unbiased stochastic quantization for gradients based on a convex problem, and designed an optimal quantization method for data samples.

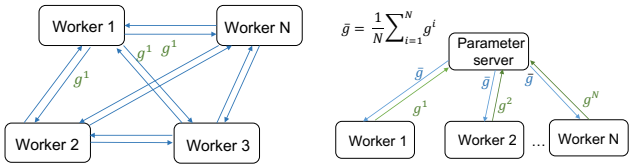


Figure 1: An illustration of two commonly used communication frameworks: broadcast (left) and parameter-server (right).

Our work distinguishes itself from the above results in: (1) considering general nonsmooth composite problems and providing convergence analysis in both convex and nonconvex settings; (2) providing theoretical understandings of a new quantization scheme with gradient clipping; (3) proposing *double sampling* in accelerated algorithms to maintaining both fast convergence and lower communication overhead.

3 PRELIMINARY

3.1 Data-Parallel Network

We consider a data-parallel distributed network with N workers. Each worker maintains a local copy of estimation model and can get access to the whole datasets stored in e.g., HDFS. At each iteration, every worker computes local stochastic gradient on a randomly sampled mini-match. Then they synchronously communicate gradients with peers to compute global gradients and update models. There are two commonly used communication frameworks, as shown in Figure 1, to realize the above data-parallel network, i.e., (1) **broadcast:** each worker broadcasts local gradients to all peers. When one worker gathers all gradients from other workers, it averages them to obtain global gradients; (2) **parameter-server**[2]**:** local gradients computed by all work-

[2]We still use the convention "parameter-server" even though only gradients are transmitted.

ers are synchronously gathered in the parameter server.

Selecting a proper communication framework is determined by real-world requirements. In Section 4.2, we provide the communication overhead comparisons when different frameworks are plugged into our algorithms.

3.2 Low-Precision Representation and Quantization

In this paper, we adopt a low-precision representation, denoted as a tuple (δ, b), of transmitted numbers. The procedure of transforming numbers from full-precision to low-precision is called quantization.

For tuple (δ, b), $b \in \mathbf{N}$ represents the amount of bits (used for storing numbers) and $\delta \in \mathbb{R}$ is the scale factor, its representation domain or quantization codebook is

$$\mathrm{dom}(\delta, b) = \{-2^{b-1} \cdot \delta, ..., -\delta, 0, \delta, ..., (2^{b-1} - 1) \cdot \delta\}.$$

Denote $Q_{(\delta,b)}(x)$ as the quantization function for a number $x \in \mathbb{R}$ given (δ, b). It outputs a point within $\mathrm{dom}(\delta, b)$ in the following rules:

(1) if x is in the convex hull of $\mathrm{dom}(\delta, b)$, without loss of generality, $x \in [z, z + \delta]$, where $z \in \mathrm{dom}(\delta, b)$. It will be stochastically rounded up or down in the following sense:

$$Q_{(\delta,b)}(x) = \begin{cases} z, & \text{with probability } \frac{z+\delta-x}{\delta}, \\ z + \delta, & \text{otherwise}. \end{cases}$$

It can be verified that the above quantization is unbiased, i.e., $\mathbf{E}[Q_{(\delta,b)}(x)] = x$. Moreover, the quantized variance or precision loss can be bounded as:

Lemma 1. *If x is in the convex hull of $\mathrm{dom}(\delta, b)$, then we have*

$$\mathbf{E}[(Q_{(\delta,b)}(x) - x)^2] \leq \delta^2/4.$$

The intuition behind Lemma 1 is that a smaller δ determines more dense quantization points, and as a result incurs less precision loss (or quantization error).

(2) on the other hand, if x is not in the convex hull of $\mathrm{dom}(\delta, b)$, it will be projected to the closest point, in other words, the smallest or largest value in $\mathrm{dom}(\delta, b)$.

In the following sections, we use function $Q_{(\delta,b)}(\cdot)$ to quantize gradient, which means each coordinate is quantized using the same scale factor independently. Also note that low-precision numbers with the same scale factor can be easily added with several bits overflow.

3.3 Notation

Throughout the paper, we denote x^* as the optimal solution of (1). d is the dimension of x. $||\cdot||_\infty$ represents the max norm of a vector. $||\cdot||$ denotes L_2 norm and the base of the logarithmic function is 2 if without special annotation.

Algorithm 1 Low-Precision SVRG with Gradient Clipping: **LPC-SVRG** (for each worker i)

Input: S, m, λ, B, η, $\tilde{x}^0 = x^0$;
for $s = 0, 1, ..., S-1$ **do**
  $x_0^{s+1} = \tilde{x}^s$;
  Communication step 1: cooperates with other workers to compute $\nabla f(\tilde{x}^s)$;
  for $t = 0$ **to** $m - 1$ **do**
    uniformly and independently samples (with replacement) a mini-batch i_B with size B to calculate $u_t^i = \frac{1}{B} \sum_{a \in i_B} [\nabla f_a(x_t^{s+1}) - \nabla f_a(\tilde{x}^s)]$;
    Quantization step: $\tilde{u}_t^i = Q_{(\delta_t^i, b)}(u_t^i)$;
    Communication step 2: cooperates with other workers to compute $\tilde{u}_t$ using equation (4) or (5);
    $v_t^{s+1} = \tilde{u}_t + \nabla f(\tilde{x}^s)$;
    $x_{t+1}^{s+1} = \mathrm{prox}_{\eta h}(x_t^{s+1} - \eta v_t^{s+1})$;
  end for
  $\tilde{x}^{s+1} = x_m^{s+1}$;
end for
Output: Uniformly choosing from $\{\{x_t^{s+1}\}_{t=0}^{m-1}\}_{s=0}^{S-1}$.

Proximal operator. To handle the nonsmooth $h(x)$ in (1), we apply the proximal operator which is formed as $\mathrm{prox}_{\eta h}(x) = \arg\min_y (h(y) + \frac{1}{2\eta}||y - x||^2)$, where $\eta > 0$. Proximal operator can be seen as a generalization of projection. If $h(x)$ is an indicator function of a closed convex set, the proximal operator becomes a projection.

Convergence metric. For convex problem, we simply use the objective gap, i.e., $P(x) - P(x^*)$ as the measurement of convergence, while for nonconvex nonsmooth problem, we adopt a commonly used metric *gradient mapping* [22]: $G_\eta(x) \triangleq \frac{1}{\eta}[x - \mathrm{prox}_{\eta h}(x - \eta \nabla f(x))]$. A point x is defined as an ϵ-accurate solution if $\mathbf{E}||G_\eta(x)||^2 \leq \epsilon$ [26, 29].

3.4 Assumption

We state the assumptions made in this paper, which are mild and are often assumed in the literature, e.g., [15, 26].

Assumption 1. *The stochastic gradient is unbiased, i.e., for a random sample $i \in \{1, ..., n\}$, $\mathbf{E}_i[\nabla f_i(x)] = \nabla f(x)$. Moreover, the random variables sampled in different iterations are independent.*

Assumption 2. *We require each function $f_i(x)$ in (1) is L-(Lipschitz) smooth, i.e., $||\nabla f_i(x) - \nabla f_i(y)|| \leq L||x - y||$.*

4 LOW-PRECISION SVRG WITH GRADIENT CLIPPING

Now we are ready to introduce our new LPC-SVRG algorithm with low-precision quantized gradients. As shown in Algorithm 1, LPC-SVRG divides the optimization process into epochs, similar to full-precision SVRG [15, 35, 26],

and each epoch contains m inner iterations. At the beginning of each epoch, we keep track of a full-batch gradient $\nabla f(\tilde{x})$, where $\tilde{x}$ is a reference point and is updated at the end of the last epoch. The model x is updated in the inner iteration, with a new stochastic gradient formed in Steps 6-9. Here $\tilde{u}_t$ is an averaged quantized gradient and will be specified later based on different communication schemes. We consider a mini-batch setting, with each node i uniformly and independently sampling a mini-batch i_B to compute stochastic gradient. If no gradient quantization is applied, i.e., $\tilde{u}_t = \frac{1}{N}\sum_{i=1}^{N} u_t^i$ in Algorithm 1, [26] has shown:

Theorem 1. *([26], Theorem 5). Suppose Assumptions 1, 2 hold, $h(x)$ is convex and $NB \leq n$. Let T be a multiple of m and $\eta = \rho/L$ where $\rho < 1/2$ and satisfies $4\rho^2 m^2/(NB) + \rho \leq 1$. Then for the output x_{out} of Algorithm 1 (with $\tilde{u}_t = \frac{1}{N}\sum_{i=1}^{N} u_t^i$), we have:*

$$\mathbf{E}[||G_\eta(x_{out})||^2] \leq \frac{2L(P(x^0) - P(x^*))}{\rho(1-2\rho)T}. \tag{2}$$

4.1 Low-Precision with Gradient Clipping

As the scale of data-parallel network and model parameters increases, the communication overhead greatly enlarges and even becomes the system performance bottleneck [38, 28, 30]. In Algorithm 1, we propose a gradient quantization method to reduce communication complexity. Specifically, in Step 7, we independently quantize each coordinate of u_t^i using tuple (δ_t^i, b) before it is transmitted. The values of δ_t^i and b determine the precision loss and the amount of communication bits. In our work, the scale factor δ_t^i is set to be

$$\delta_t^i = \frac{\lambda||u_t^i||_\infty}{2^{b-1}-1}, \tag{3}$$

where $\lambda \in (0, 1]$ plays the role of clipping parameter.

Unbiased quantization. If $\lambda = 1$, it can be verified that each coordinate of u_t^i is in the convex hull of $\mathrm{dom}(\delta_t^i, b)$. Thus, $\mathbf{E}[\tilde{u}_t^i] = u_t^i$. Such unbiased quantization method is equivalent to the quantization function adopted in [3, 33, 38, 34], where the number of quantization levels adopted equals to the number of positive points in $\mathrm{dom}(\delta, b)$.

Biased quantization with gradient clipping. Lemma 1 shows that a larger δ_t^i leads to a bigger quantization error. Thus, for each coordinate in the convex hull of $\mathrm{dom}(\delta_t^i, b)$, the quantization error will be reduced by a factor of λ^2 if $\lambda < 1$. On the other hand, when $\lambda < 1$, there exist coordinates exceeding $\mathrm{dom}(\delta_t^i, b)$, which will be projected to the nearest representable values. That's we call gradient clipping. To demonstrate the intuition of gradient clipping, in Figure 2, we visualize the histograms of u_t^i calculated in ResNet-20 model [11] on dataset CIFAR-10 [17], and logistic regression on MNIST [6]. We can see that the distribution of u_t^i is very close to a Gaussian distribution and very few coordinates exceed region $[-c\sigma, c\sigma]$, where σ is the approximated standard deviation. Therefore, setting $\lambda = 1$ incurs great quantization error since $||u_t^i||_\infty$ is usually far more large.

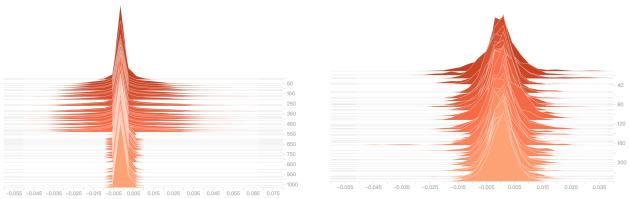


Figure 2: Histograms of gradients u_t^i (see Algorithm 1), left: logistic regression for MNIST; right: the second convolutional layer of ResNet-20 for CIFAR-10. The vertical axis represents the index of training iterations.

Such Gaussian distribution (of gradients in SGD) has also been recorded in [33], and they conducted cross validation to choose the clipping parameter. However, only convergence proof under unbiased quantization with no gradient clipping was provided. In this work, we formally integrate gradient clipping technique into quantization through a parameter λ and provide the theoretical analysis for biased quantization in Section 4.3.

4.2 Communication Schemes and Complexities

In this section, we present the communication schemes for the above quantized low-precision gradients. As shown in Algorithm 1, there are two communication steps. The first one only involves full-precision operations, and its communication overhead can be neglected since it only happens once per epoch. Below, we provide more details for **Communication step** 2. First note that when the i-th worker transmits $\tilde{u}_t^i$, it only needs to send b bits representations of all coordinates and an extra float δ_t^i, with a total number of $(32 + bd)$ bits. It is dramatically smaller than $32d$ bits required by unquantized gradient. Based on the two data-parallel frameworks introduced in Section 3.1, there are three possible communication schemes:

(a.) (**Broadcast**) For each worker i, the scale factor δ_t^i is computed using (3). When receiving low-precision gradients from other workers, it first recovers their full-precision representations and calculates $\tilde{u}_t$ using equation (4). In this case, precision loss only happens in calculating $\tilde{u}_t^i$, and the resulting communication cost is $(32 + bd)N(N-1)$ bits.

$$\tilde{u}_t = \frac{1}{N}\sum_{i=1}^{N} \tilde{u}_t^i. \tag{4}$$

(b.) (**Parameter-server without re-quantizing**) The adding operation can only be conducted among low-precision numbers with the same scale factor. Therefore, $2N$ floating numbers need to be transmitted between server and workers to unify the same scale factor $\delta_t = \max_i\{\delta_t^i\}$.Then all workers adopt δ_t in Step 7, i.e., $\delta_t^i = \delta_t$ for each

Table 1: Comparisons of different schemes for **Communication step** 2 in Algorithm 1. The values of the last column equal to the ratio of bits used by full-precision transmission to that of low-precision, e.g., for **Broadcast** scheme, it equals to $32dN(N-1)/(32+bd)N(N-1)$.

Communication scheme	With re-quantizing	Bits overflow	# of transmitted bits	Communication reduced factor
Broadcast	No	No	$(32+bd)N(N-1)$	$\frac{32}{32/d+b}$
Parameter-server	No	Yes	$N(64+2bd+d\lceil\log N\rceil)$	$\frac{32}{32/d+b+(\lceil\log N\rceil)/2}$
Parameter-server	Yes	No	$N(64+2bd)$	$\frac{32}{32/d+b}$

i. When server calculates the average gradients $\tilde{u}_t$ using (4), it allows for an overflow of $\lceil\log N\rceil$ bits, and then send $\tilde{u}_t$ to workers. The total communication overhead is $N(64+2bd+d\lceil\log N\rceil)$ bits, and the precision loss also comes from calculating $\tilde{u}_t^i$.

(c.) (**Parameter-server with re-quantizing**) Communication scheme is the same as **(b)**, except for

$$\tilde{u}_t = Q_{(\delta_t,b)}(\frac{1}{N}\sum_{i=1}^{N}\tilde{u}_t^i), \quad \delta_t = \max_i\{\delta_t^i\}, \tag{5}$$

i.e., server re-quantizes $(\frac{1}{N}\sum_i \tilde{u}_t^i)$ to a b-bit low-precision representation with scale factor δ_t (to prevent bit overflow). Note that the quantization in (5) is unbiased since all $u_t^{i\prime}$s are also quantized with δ_t. In this case, there exist two levels of quantization errors and the total communication complexity equals to $N(64+2bd)$. The comparisons of three communication schemes are summarized in Table 1.

4.3 Theoretical Analysis

Based on the above low-precision representation and communication schemes, in this section, we provide the convergence analysis of Algorithm 1. We begin with several lemmas which bound the variance of low-precision gradient.

Lemma 2. *If Assumptions 1, 2 hold,*
(i). under communication scheme ***(a)*** *or* ***(b)*** *with* $\tilde{u}_t = \frac{1}{N}\sum_i \tilde{u}_t^i$, *we have*

$$\mathbf{E}||v_t^{s+1}-\nabla f(x_t^{s+1})||^2 \leq$$
$$2L^2[\frac{d\lambda^2}{4(2^{b-1}-1)^2}+d_\lambda(1-\lambda)^2+\frac{1}{NB}]\mathbf{E}||x_t^{s+1}-\tilde{x}^s||^2;$$

(ii). under communication scheme ***(c)*** *with* $\tilde{u}_t = Q_{(\delta_t,b)}(\frac{1}{N}\sum_{i=1}^{N}\tilde{u}_t^i)$, *we obtain a variance bound of*

$$\mathbf{E}||v_t^{s+1}-\nabla f(x_t^{s+1})||^2 \leq$$
$$2L^2[\frac{3d\lambda^2}{8(2^{b-1}-1)^2}+d_\lambda(1-\lambda)^2+\frac{1}{NB}]\mathbf{E}||x_t^{s+1}-\tilde{x}^s||^2,$$

where $d_\lambda = \max_i\{d_\lambda^i\}$. d_λ^i *is the number of coordinates in* u_t^i *exceeding* $\mathrm{dom}(\delta_t^i, b)$. *Note that* $\delta_t^i = \delta_t$ *in schemes* ***(b)*** *and* ***(c)***.

Remarks. The coefficients of variance bound in Lemma 2, e.g., *(i)* can be decomposed into three items, i.e., $A = \frac{d\lambda^2}{4(2^{b-1}-1)^2}$, $B = d_\lambda(1-\lambda)^2$, $C = \frac{1}{NB}$, where A and B come from quantization and the last item C inherits the gradient variance of SVRG [26, 15]. If without quantization, $A = B = 0$, and Lemma 2 becomes equivalent to Lemma 3 in [26]. Moreover, Lemma 2 theoretically shows the benefit of quantization with gradient clipping, i.e., with proper choice of $\lambda < 1$, $(A|_{\lambda<1}+B) < A|_{\lambda=1}$. Such condition can always hold in the case where gradients follow a Gaussian distribution. Specifically, when $A|_{\lambda=1}$ is dominating, a smaller λ can effectively shrink its value while fewer coordinates, i.e., very small d_λ, comes into B.

Theorem 2. *Suppose Assumptions 1, 2 hold,* $h(x)$ *is convex,* $T = Sm$, $\eta = \frac{\rho}{L}$, $\rho < \frac{1}{2}$, *and all parameters satisfy*

$$8m^2\rho^2[\frac{d\lambda^2}{4(2^{b-1}-1)^2}+d_\lambda(1-\lambda)^2+\frac{1}{NB}]+\rho \leq 1 \tag{6}$$

when using communication scheme ***(a)*** *or* ***(b)*** (d_λ *is defined in Lemma 2). For the output* x_{out} *of Algorithm 1, we have*

$$\mathbf{E}||G_\eta(x_{out})||^2 \leq \frac{2L(P(x^0)-P(x^*))}{\rho(1-2\rho)T}. \tag{7}$$

Moreover, if communication scheme ***(c)*** *is adopted, the same rate (7) can be obtained if*

$$8m^2\rho^2[\frac{3d\lambda^2}{8(2^{b-1}-1)^2}+d_\lambda(1-\lambda)^2+\frac{1}{NB}]+\rho \leq 1. \tag{8}$$

Corollary 1. *If* $b = O(\log\sqrt{d})$ *and* $\eta = O(1/mL)$, *there exists* $\lambda \in (0,1]$ *that guarantees constraints (6) and (8).*

The above theorem shows the same asymptotic convergence rate (as Theorem 1) can be achieved with low-precision representation. In the following section, we propose a faster and communication-efficient algorithm with *double sampling*.

5 ACCELERATED LOW-PRECISION ALGORITHM

Recently momentum or Nesterov [23] technique has been successfully combined with SVRG to achieve faster convergence in real-world applications [4, 12, 20]. In this section, we propose an accelerated method named ALPC-SVRG, based on Katyusha momentum [4], to obtain both faster running speed and high communication-efficiency. As shown in

Algorithm 2 Accelerated LPC-SVRG: **ALPC-SVRG** (for each worker i, strongly convex case)

Input: S, m, λ, B, τ_1, τ_2, α, $y_0 = z_0 = x_0 = \tilde{x}^0$;
for $s = 0, 1, ..., S-1$ **do**
  Communication step 1: cooperates with other workers to compute $\nabla f(\tilde{x}^s)$;
  for $t = 0$ **to** $m-1$ **do**
    $k \leftarrow (sm + t)$;
    $x_{k+1} = \tau_1 z_k + \tau_2 \tilde{x}^s + (1 - \tau_1 - \tau_2) y_k$;
    uniformly and independently samples (with replacement) a mini-batch i_B with size B to calculate $u^i_{k+1} = \frac{1}{B} \sum_{a \in i_B} [\nabla f_a(x_{k+1}) - \nabla f_a(\tilde{x}^s)]$;
    Quantization step: $\tilde{u}^i_{k+1} = Q_{(\delta^i_{k+1}, b)}(u^i_{k+1})$;
    Communication step 2: cooperates with other workers to compute $\tilde{u}_{k+1} = \frac{1}{N} \sum_{i=1}^{N} \tilde{u}^i_{k+1}$;
    $v_{k+1} = \tilde{u}_{k+1} + \nabla f(\tilde{x}^s)$;
    Double sampling: independently (to i_B) samples a mini-batch J with size B to compute $\hat{v}_{k+1} = \frac{1}{B} \sum_{j \in J} (\nabla f_j(x_{k+1}) - \nabla f_j(\tilde{x}^s)) + \nabla f(\tilde{x}^s)$;
    $y_{k+1} = \arg\min_y \{2L||y - x_{k+1}||^2 + \langle v_{k+1}, y\rangle + h(y)\}$; $\diamondsuit$ *with low-precision gradient*
    $z_{k+1} = \arg\min_z \{\frac{1}{2\alpha}||z - z_k||^2 + \langle \hat{v}_{k+1}, z\rangle + h(z)\}$; $\diamondsuit$ *with local full-precision gradient*
  end for
  $\tilde{x}^{s+1} = \left(\sum_{t=0}^{m-1} (1 + \alpha\sigma)^t\right)^{-1} \cdot \left(\sum_{t=0}^{m-1} (1 + \alpha\sigma)^t \cdot y_{sm+t+1}\right)$;
end for
Output: $\tilde{x}^S$.

Algorithm 2, at the beginning of inner iterations, we initialize x_{k+1} as a convex combination of the reference point $\tilde{x}^s$ and two auxiliary variables z_k and y_k. We propose *double sampling* to compute the stochastic gradients for z_k and y_k, where the gradient v_{k+1} used in y_k-update is the same as that of Algorithm 1, and a new gradient $\hat{v}_{k+1}$ with independent *double sampling* is applied in computing z_{k+1}. Note that $\hat{v}_{k+1}$ has full-precision and is calculated locally in each worker without communication. To make sure the consistency of model x in all workers, we assume the mini-batches J used by all workers are the same. This can be achieved by setting the identical random seeds for J.

In this section, quantization with gradient clipping is also considered and λ is the clipping parameter, i.e., $\delta^i_{k+1} = \frac{\lambda||u^i_{k+1}||_\infty}{2^{b-1}-1}$. Without loss of generality, we use the communication scheme **(a)** for the analysis below, and it is easy to extend the following conclusions to the other two communication schemes.

Theorem 3. *Suppose Assumptions 1, 2 hold and each $f_i(x)$ is convex, $h(x)$ is σ-strongly convex, i.e., there exists $\sigma > 0$ such that $\forall x, y$,*

$$h(y) \geq h(x) + \langle \nabla h(x), y - x\rangle + \sigma/2||y - x||^2.$$

Denote $\zeta \triangleq \frac{d\lambda^2}{4(2^{b-1}-1)^2} + d_\lambda(1-\lambda)^2 + \frac{1}{NB}$, where $d_\lambda = \max_i\{d^i_\lambda\}$, d^i_λ is the number of coordinates in u^i_{k+1} exceeding $\mathrm{dom}(\delta^i_{k+1}, b)$. Let $\alpha = \frac{1}{6\tau_1 L}$, $m \leq \frac{3L}{2\sigma}$, if τ_1, τ_2 satisfy

$$\tau_1 = \sqrt{\frac{m\sigma}{6L}}, \quad \tau_2 = \frac{5\zeta}{3} + \frac{1}{2B} \leq \frac{1}{2}.$$

Then under Algorithm 2, we obtain

$$\mathbf{E}[P(\tilde{x}^S) - P(x^*)] \leq O((1 + \sqrt{\frac{\sigma}{Lm}})^{-Sm})(P(x_0) - P(x^*)).$$

It can be verified that if $b = O(\log\sqrt{d})$, there exist $\lambda \in (0, 1]$ and mini-batch size B that make sure $\tau_2 \leq \frac{1}{2}$, and we obtain the same convergence rate as the full-precision algorithm (Katyusha, Case 1) in [4].

Algorithm 3 Accelerated LPC-SVRG: **ALPC-SVRG** (for each worker i, general convex case)

Input: S, m, λ, B, τ_2, $y_0 = z_0 = x_0 = \tilde{x}^0$;
for $s = 0, 1, ..., S-1$ **do**
  updates $\tau_{1,s} = \frac{2}{s+4}$, $\alpha_s = \frac{1}{6L\tau_{1,s}}$;
  performs the same Steps 3-14 as in Algorithm 2;
  $\tilde{x}^{s+1} = \frac{1}{m} \sum_{t=1}^{m} y_{sm+t}$;
end for
Output: $\tilde{x}^S$.

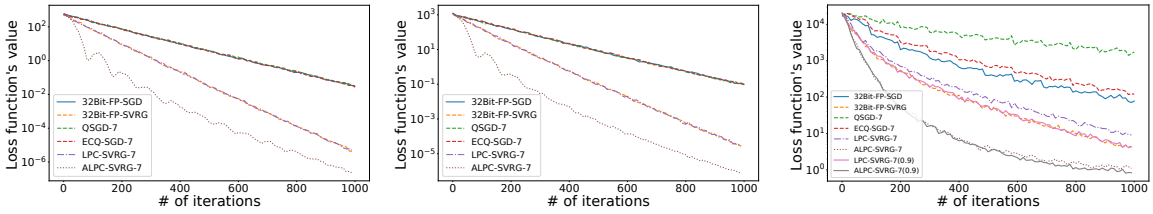
The following theorem provides the convergence rate for ALPC-SVRG without assuming strong convexity. As shown in Algorithm 3, we update the value of τ_1 and α at each epoch s, and τ_2 is the same as Algorithm 2. The update of $\tilde{x}^{s+1}$ is also adjusted to conduct telescoping summation. Based on the same communication scheme and low-precision representations, we have the following result.

Theorem 4. *Denote $\zeta \triangleq \frac{d\lambda^2}{4(2^{b-1}-1)^2} + d_\lambda(1-\lambda)^2 + \frac{1}{NB}$. Let $\tau_2 = \frac{5\zeta}{3} + \frac{1}{2B}$, if Assumptions 1, 2 hold, each $f_i(x)$ and $h(x)$ are convex and $\tau_2 \leq \frac{1}{2}$, then under Algorithm 3, we have*

$$\mathbf{E}[P(\tilde{x}^S) - P(x^*)] \leq O(\frac{1}{mS^2})[m(P(x_0) - P(x^*)) + L||x_0 - x^*||^2].$$

6 EXPERIMENTS

In this section, we conduct extensive experiments to validate the effectiveness of our proposed algorithms. Firstly, we evaluate linear models on various datasets and then extend our algorithms to train deep neural networks on datasets CIFAR-10 [17] and ILSVRC-12 [27]. In all evaluations, the communication scheme **(a)** described in Section 4.2 is adopted.



(a) Syn-512 (b) Syn-1024 (c) Syn-20k

Figure 3: The training curves of compared algorithms on three synthetic datasets. Here the suffix denotes the number of quantization levels and the number in bracket is the value of clipping parameter.

6.1 Evaluation on Linear Model

We begin with linear regression on three synthetic datasets: Syn-512, Syn-1024, Syn-20k[3], each containing 10k, 10k, 50k training samples. Here the suffix denotes the dimension of model parameters. We compare LPC-SVRG and ALPC-SVRG with 32 bits full-precision algorithms: SGD, SVRG [15], and the state-of-the-art low-precision algorithms: QSGD [3], ECQ-SGD [34]. Among the above algorithms, QSGD and ECQ-SGD are the low-precision variants of SGD, and LPC/ALPC-SVRG is based on full-precision SVRG [15, 26]. Similar to related works [38, 4, 26], we use diminishing/constant learning rate (*lrn_rate* for short) for SGD/SVRG based algorithms respectively. In our experiments, we have tuned the *lrn_rates* for the best performance of each algorithm. The hyper-parameters in ALPC-SVRG are setting to be $\tau_1 = \frac{2}{s+4}$, $\alpha = lrn\_rate/\tau_1$, $\tau_2 = \frac{1}{2}$.

For low-precision algorithms, we use max norm in the scaling factor and adopt the entropy encoding scheme [13] to further reduce the communication overhead. Here for the consistency of notation, we adopt *levels* mentioned in [3, 34] to represent the extent of low-precision representation. As mentioned in Section 4.1, the quantization *levels* equal to the number of positive points in $\mathrm{dom}(\delta, b)$.

Convergence. Figure 3(a) and 3(b) depict the training curves of six algorithms on datasets syn-512 and syn-1024, where the suffix of low-precision algorithm denotes the number of quantization levels. It shows that all low-precision algorithms have comparable convergence to their corresponding full-precision algorithms, which verifies the redundancy of 32-bit representation. On the other hand, we can see that LPC-SVRG converges faster than QGD and ECQ-SGD with the same quantization levels. And the optimal convergence of ALPC-SVRG is validated in these two figures.

Gradient clipping. We conduct gradient clipping on a larger dataset syn-20k, where for LPC/ALPC-SVRG-7(0.9), 7 represents the quantization levels and 0.9 is the clipping parameter, i.e., λ. The value of λ is determined by cross validation and we only conduct it once. As shown in Figure 3(c), LPC-SVRG without clipping converges slower than SVRG because the precision loss becomes significant for the larger dataset syn-20k. The same circumstance is also presented in QSGD and ECQ-SGD. In this case, gradient clipping comes into effect. With more dense quantization points, it can reduce the quantization error and achieve better performance.

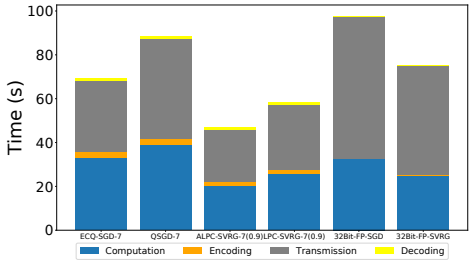


Figure 4: Comparisons on the time consumption of different algorithms on dataset syn-20k.

Communication time. To better demonstrate the effectiveness of LPC/ALPC-SVRG, we analyze the time consumption for different algorithms on dataset syn-20k. Specifically, we decompose the total running time into 4 parts, including *computation*, *encoding*, *transmission*, *decoding*, and the time is reported till similar convergence (i.e. first training loss below 10^2). As shown in Figure 4, compared to full precision SVRG, LPC-SVRG saves the total running time by significantly reducing the transmission overhead, and ALPC-SVRG requires the fewest computation and transmission time among all algorithms.

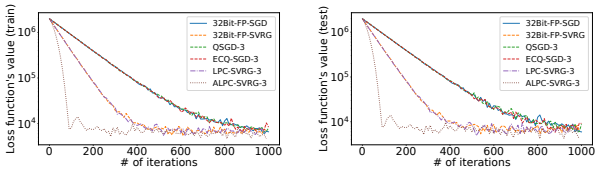


Figure 5: Validation on dataset *YearPredictionMSD* (left: training loss, right: test loss).

Real-world dataset and communication overhead. We

[3]These datasets are generated using the same method as in [34] with *i.i.d* random noise.

Table 2: Communication costs of different algorithms for linear regression on dataset *YearPredictionMSD*. All low-precision algorithms below use three quantization levels.

Algorithm	Training loss	# of bits	Ratio
32-bit-FP-SGD	$5.74e3$	$3.25e6$	–
32-bit-FP-SVRG	$5.55e3$	$1.41e6$	$2.3\times$
QSGD	$5.96e3$	$1.61e5$	$20.19\times$
ECQ-SGD	$5.90e3$	$1.60e5$	$20.31\times$
LPC-SVRG	$5.80e3$	$7.04e4$	$46.16\times$
ALPC-SVRG	$5.86e3$	$3.50e4$	$92.86\times$

further evaluate linear regression on public dataset *YearPredictionMSD* [6]. In this case, fewer (three) levels are used since the feature dimension is smaller (i.e., 90). As shown in Figure 5, all low-precision algorithms achieve similar convergence as their corresponding full-precision algorithms. Moreover, the empirical results validate the fast convergence of LPC/ALPC-SVRG.

In Table 2, we record the total number of communication costs of various algorithms for achieving similar training loss. It shows LPC-SVRG and ALPC-SVRG can save up to $46.16\times$ and $92.86\times$ communication costs compared with the benchmark, which validates the effectiveness of our proposed algorithms.

6.2 Evaluation on Deep Learning Model

We further extend our algorithms to train deep neural networks. The public dataset CIFAR-10 [17] is used, with 50k training and 10k test images. We setup experiments on TensorFlow [1] with ResNet-20 model [11], and adopt a decreasing learning rate, i.e., starting from 0.1 and divided by 10 at 40k and 60k iterations. All low-precision algorithms use the same quantization levels, i.e., 7. The other training hyper-parameters are tuned to achieve similar loss as the benchmark full-precision SGD. Figure 6 shows the training statistics of each algorithm. The left graph reports the training curves versus iterations and the values in bracket are test classification error. These curves show that our algorithms converge with a faster speed. The right graph plots the total amount of transmitted bits of compared algorithms (till similar convergence, i.e., training loss first below 0.17).

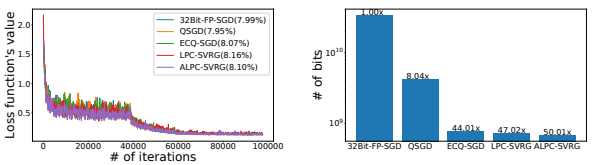


Figure 6: Validation on CIFAR-10 with ResNet-20 model, left: training curves, right: the amount of transmitted bits.

In this experiment, a clipping parameter with value 0.85 is adopted in LPC-SVRG and ALPC-SVRG. We discover that our proposed algorithms effectively save the communication overhead.

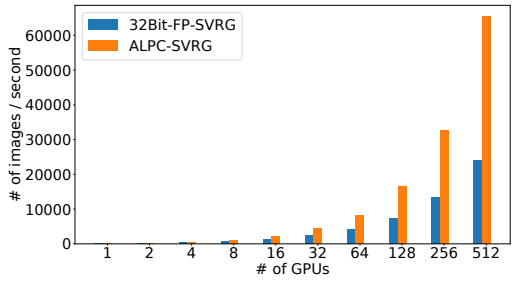


Figure 7: Performance model: training throughput of ResNet-50 on ILSVRC-12 with the increasing of GPUs.

Performance Model. To verify the scalability of ALPC-SVRG, we use the performance model proposed in [37]. In this experiment, the performance denotes the training speed. Similar to [34, 33], we use the lightweight profiling on the computation and communication time of a single machine to estimate the performance for larger distributed systems. The major hardware statistics are as follows: Nvidia Tesla P40 GPU (8 units per node), Intel Xeon E5-2680 CPU and Mellanox ConnectX-3 Pro network card (with 40Gbps network connections).

We setup experiments on ResNet-50 model on dataset ILSVRC-12 [27]. Figure 7 presents the training throughput with the increasing of GPUs. It shows that our algorithm can achieve significant speedup over full precision SVRG when applied to large-scale networks.

7 CONCLUSION

In this paper, we propose LPC-SVRG and its acceleration ALPC-SVRG to achieve both fast convergence and lower communication complexity. We present a new quantization method with gradient clipping adopted and mathematically analyze its convergence. With *double sampling*, we are able to combine the gradients of both full-precision and low-precision and then achieve acceleration. Our analysis covers general nonsmooth composite problems, and shows the same asymptotic convergence rate can be attained with only $O(\log \sqrt{d})$ bits (compared to full-precision of 32 bits). The experiments on linear models and deep neural networks demonstrate the fast convergence and communication efficiency of our algorithms.

Acknowledgements

The work by Yue Yu is supported in part by the National Natural Science Foundation of China Grants 61672316. We would like to thank Chaobing Song for the useful discussions and thank anonymous reviewers for their insightful suggestions.

References

[1] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, et al. Tensorflow: a system for large-scale machine learning. In *OSDI*, volume 16, pages 265–283, 2016.

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

[3] 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*, pages 1709–1720, 2017.

[4] Z. Allen-Zhu. Katyusha: The first direct acceleration of stochastic gradient methods. In *STOC*, pages 1200–1205, 2017.

[5] R. Bekkerman, M. Bilenko, and J. Langford. *Scaling up machine learning: Parallel and distributed approaches*. Cambridge University Press, 2011.

[6] C.-C. Chang and C.-J. Lin. Libsvm: a library for support vector machines. *ACM transactions on intelligent systems and technology (TIST)*, 2(3):27, 2011.

[7] T. M. Chilimbi, Y. Suzue, J. Apacible, and K. Kalyanaraman. Project adam: Building an efficient and scalable deep learning training system. In *OSDI*, volume 14, pages 571–582, 2014.

[8] C. De Sa, M. Leszczynski, J. Zhang, A. Marzoev, C. R. Aberger, K. Olukotun, and C. Ré. High-accuracy low-precision training. *arXiv preprint arXiv:1803.03383*, 2018.

[9] C. M. De Sa, C. Zhang, K. Olukotun, and C. Ré. Taming the wild: A unified analysis of hogwild-style algorithms. In *Advances in neural information processing systems*, pages 2674–2682, 2015.

[10] J. Dean, G. Corrado, R. Monga, K. Chen, M. Devin, M. Mao, A. Senior, P. Tucker, K. Yang, Q. V. Le, et al. Large scale distributed deep networks. In *Advances in neural information processing systems*, pages 1223–1231, 2012.

[11] K. He, X. Zhang, S. Ren, and J. Sun. Identity mappings in deep residual networks. In *European conference on computer vision*, pages 630–645. Springer, 2016.

[12] L. T. K. Hien, C. V. Nguyen, H. Xu, C. Lu, and J. Feng. Accelerated stochastic mirror descent algorithms for composite non-strongly convex optimization. *arXiv preprint arXiv:1605.06892*, 2016.

[13] D. A. Huffman. A method for the construction of minimum-redundancy codes. *Proceedings of the IRE*, 40(9):1098–1101, 1952.

[14] F. N. Iandola, M. W. Moskewicz, K. Ashraf, and K. Keutzer. Firecaffe: near-linear acceleration of deep neural network training on compute clusters. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2592–2600, 2016.

[15] R. Johnson and T. Zhang. Accelerating stochastic gradient descent using predictive variance reduction. In *Advances in neural information processing systems*, pages 315–323, 2013.

[16] S. Kanai, Y. Fujiwara, and S. Iwamura. Preventing gradient explosions in gated recurrent units. In *Advances in Neural Information Processing Systems*, pages 435–444, 2017.

[17] A. Krizhevsky and G. Hinton. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.

[18] M. Li, D. G. Andersen, A. J. Smola, and K. Yu. Communication efficient distributed machine learning with the parameter server. In *Advances in Neural Information Processing Systems*, pages 19–27, 2014.

[19] X. Lian, Y. Huang, Y. Li, and J. Liu. Asynchronous parallel stochastic gradient for nonconvex optimization. In *Advances in Neural Information Processing Systems*, pages 2737–2745, 2015.

[20] H. Lin, J. Mairal, and Z. Harchaoui. A universal catalyst for first-order optimization. In *Advances in Neural Information Processing Systems*, pages 3384–3392, 2015.

[21] Y. Lin, S. Han, H. Mao, Y. Wang, and W. J. Dally. Deep gradient compression: Reducing the communication bandwidth for distributed training. In *ICLR*, 2018.

[22] Y. Nesterov. *Introductory lectures on convex optimization: A basic course*, volume 87. Springer Science & Business Media, 2013.

[23] Y. E. Nesterov. A method for solving the convex programming problem with convergence rate o (1/kˆ 2). In *Dokl. Akad. Nauk SSSR*, volume 269, pages 543–547, 1983.

[24] R. Pascanu, T. Mikolov, and Y. Bengio. On the difficulty of training recurrent neural networks. In *International Conference on Machine Learning*, pages 1310–1318, 2013.

[25] B. Recht, C. Re, S. Wright, and F. Niu. Hogwild: A lock-free approach to parallelizing stochastic gradient descent. In *Advances in neural information processing systems*, pages 693–701, 2011.

[26] S. J. Reddi, S. Sra, B. Poczos, and A. Smola. Fast stochastic methods for nonsmooth nonconvex optimization. *arXiv preprint arXiv:1605.06900*, 2016.

[27] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.

[28] 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 *Fifteenth Annual Conference of the International Speech Communication Association*, 2014.

[29] S. Sra. Scalable nonconvex inexact proximal splitting. *Advances in Neural Information Processing Systems*, pages 539–547, 2012.

[30] N. Strom. Scalable distributed dnn training using commodity gpu cloud computing. In *Sixteenth Annual Conference of the International Speech Communication Association*, 2015.

[31] H. Wang, S. Sievert, Z. Charles, D. Papailiopoulos, and S. Wright. Atomo: Communication-efficient learning via atomic sparsification. *Advances in Neural Information Processing Systems*, 2018.

[32] J. Wangni, J. Wang, J. Liu, and T. Zhang. Gradient sparsification for communication-efficient distributed optimization. *arXiv preprint arXiv:1710.09854*, 2017.

[33] 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*, pages 1509–1519, 2017.

[34] J. Wu, W. Huang, J. Huang, and T. Zhang. Error compensated quantized sgd and its applications to large-scale distributed optimization. In *International Conference on Machine Learning*, 2018.

[35] L. Xiao and T. Zhang. A proximal stochastic gradient method with progressive variance reduction. *SIAM Journal on Optimization*, 24(4):2057–2075, 2014.

[36] E. P. Xing, Q. Ho, W. Dai, J. K. Kim, J. Wei, S. Lee, X. Zheng, P. Xie, A. Kumar, and Y. Yu. Petuum: A new platform for distributed machine learning on big data. *IEEE Transactions on Big Data*, 1(2):49–67, 2015.

[37] F. Yan, O. Ruwase, Y. He, and T. Chilimbi. Performance modeling and scalability optimization of distributed deep learning systems. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1355–1364. ACM, 2015.

[38] 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 *International Conference on Machine Learning*, pages 4035–4043, 2017.

[39] S. Zhou, Y. Wu, Z. Ni, X. Zhou, H. Wen, and Y. Zou. Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients. *arXiv preprint arXiv:1606.06160*, 2016.