On the Discrepancy between the Theoretical Analysis and Practical Implementations of Compressed Communication for Distributed Deep Learning

Aritra Dutta, El Houcine Bergou*, Ahmed M. Abdelmoniem, Chen-Yu Ho, Atal Narayan Sahu, Marco Canini, Panos Kalnis
KAUST

Abstract

Compressed communication, in the form of sparsification or quantization of stochastic gradients, is employed to reduce communication costs in distributed data-parallel training of deep neural networks. However, there exists a discrepancy between theory and practice: while theoretical analysis of most existing compression methods assumes compression is applied to the gradients of the entire model, many practical implementations operate individually on the gradients of each layer of the model.

In this paper, we prove that layer-wise compression is, in theory, better, because the convergence rate is upper bounded by that of entire-model compression for a wide range of biased and unbiased compression methods. However, despite the theoretical bound, our experimental study of six well-known methods shows that convergence, in practice, may or may not be better, depending on the actual trained model and compression ratio. Our findings suggest that it would be advantageous for deep learning frameworks to include support for both layer-wise and entire-model compression.

1 Introduction

Despite the recent advances in deep learning and its widespread transformative successes, training deep neural networks (DNNs) remains a computationally-intensive and time-consuming task. The continuous trends towards larger volumes of data and bigger DNN model sizes require to scale out training by parallelizing the optimization algorithm across a set of workers. The most common scale out strategy is data parallelism where each worker acts on a partition of input data. In each iteration of the optimization algorithm – typically the stochastic gradient descent (SGD) algorithm – every worker processes a mini-batch of the input data and produces corresponding stochastic gradients. Then, gradients from all workers are aggregated to produce an update to model parameters to be applied prior to the next iteration. The gradient aggregation process involves network communication and is supported via a parameter-server architecture or collective communication routines (e.g., `all_reduce`).

A major drawback of distributed training across parallel workers is that training time can negatively suffer from high communication costs, especially at large scale, due to the transmission of stochastic gradients. To alleviate this problem, several lossy compression techniques have been proposed (Seide et al. 2014; Dettmers 2016; Alistarh et al. 2017; Wen et al. 2017; Alistarh et al. 2018; Bernstein et al. 2018; Tang et al. 2018; Wangni et al. 2018; Horváth et al. 2019).

Two main classes of compression approaches are sparsification and quantization. *Sparsification* communicates only a subset of gradient elements. For instance, this is obtained by selecting uniformly at random $k\%$ elements or the top $k\%$ elements by magnitude (Alistarh et al. 2018). *Quantization*, on the other hand, represents gradient elements with lower precision, thus using fewer bits for each element. For instance, this is done by transmitting only the sign of each element (Bernstein et al. 2018), or by randomized rounding to a discrete set of values (Alistarh et al. 2017).

In theory, such methods can reduce the amount of communication and their analysis reveal that they provide convergence guarantees (under certain analytic assumptions). Further, such methods preserve model accuracy across a range of settings in practice.

However, we observe a discrepancy between the theoretical analysis and practical implementation of existing compression methods. To the best of our knowledge, the theoretical analysis of every prior method appears to assume that compression is applied to the gradient values of the *entire model*. However, from our study of existing implementations (Zhang et al. 2017; Shi, Wang, and Chu 2017; Lim, Andersen, and Kaminsky 2019; Shi, Chu, and Li 2019; Horváth et al. 2019) and experience with implementing compression methods, we observe that compression is applied *layer by layer*, as illustrated in Figure 1. In fact, based on the existing programming interfaces in modern distributed machine learning toolkits such as PyTorch (pytorch.org) and TensorFlow (tensorflow.org), a layer-wise implementation is typically most straightforward because wait-free backpropagation (Zhang et al. 2017) – where gradients are sent as soon as they are available – is a commonly used optimization.

Importantly, layer-wise compression in general differs from entire-model compression (though for certain quantiza-

*The author is also with INRA and has an equal contribution with the first author.

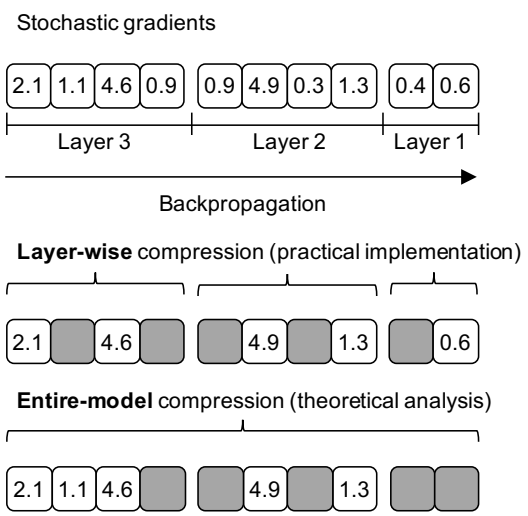


Figure 1: Contrived example of compressed communication illustrating how layer-wise Top k compression (with $k = 50\%$) differs from entire-model compression.

tion methods the results are identical). For example, Figure 1 shows the effects of Top k with a sparsification ratio of 50%, highlighting that when entire-model compression is used, no gradient for the last layer is transmitted at that specific step, which may affect convergence. This suggests that the choice of a compression approach may warrant careful consideration in practice.

In particular, this discrepancy has important implications that motivate this paper. First, since implementation artifacts differ from what has been theoretically analyzed, do theoretical guarantees continue to hold? Second, how does the convergence behavior in the layer-wise compression setting theoretically compare to entire-model compression? Third, in practice, are there significant differences in terms of convergence behavior when entire-model or layer-wise compression is applied? And if so, how do these differences vary across compression methods, compression ratios, and DNN models? To the best of our knowledge, this is the first paper to observe the above discrepancy and explore these questions. To answer these questions, this paper makes the following contributions.

Layer-wise bidirectional compression analysis: We introduce a unified theory of convergence analysis for distributed SGD with *layer-wise* compressed communication. Our analysis encompasses the majority of existing compression methods and applies to both biased (e.g., Top k, Random k, signSGD (Bernstein et al. 2018)) and unbiased methods (e.g., QSGD (Alistarh et al. 2017), TernGrad (Wen et al. 2017), $\mathcal{C}_{\rm NAT}$ (Horváth et al. 2019)). Additionally, our analysis considers *bidirectional compression*, that is, compression at both the worker side and parameter server side, mimicking the bidirectional strategy used in several compression methods (Bernstein et al. 2018; Horváth et al. 2019). Our theoretical analysis gives a proof of *tighter convergence bounds* for layer-wise compression as compared to entire-model compression.

Evaluation on standard benchmarks: We confirm our analytical findings by empirically evaluating a variety of compression methods (Random k, Top k, TernGrad, Adaptive Threshold, Threshold v, and QSGD) with standard CNN benchmarks (Coleman et al. 2017) for a range of models (AlexNet, ResNet-9, and ResNet-50) and datasets (CIFAR-10 and ImageNet). We mainly find that, in many cases, layer-wise compression is better or comparable to entire-model compression in terms of test accuracy at model convergence. However, despite the theoretical findings, our empirical results reveal that in practice there are cases, such as the Top k method with small sparsification ratio k and small model sizes, where layer-wise compression performs worse than entire-model compression. This suggests that the current practice of implementing compression methods in a layer-wise fashion out of implementation expedience may not be optimal in all cases. Thus, it would be advantageous for distributed deep learning toolkits to include support for both layer-wise and entire-model compression.

2 Preliminaries

Distributed DNN training builds on the following optimization problem:

$$\min_{x\in\mathbb{R}^d} f(x) := \frac{1}{n}\min_{x\in\mathbb{R}^d}\sum_{i=1}^{n}\underbrace{\mathbb{E}_{\xi\sim\mathcal{D}_i}F_i(x,\xi)}_{:=f_i(x)}. \quad (1)$$

Without loss of generality, consider the above problem as a classical empirical risk minimization problem over n workers, where $\mathcal{D}_i$ is the local data distribution for worker i, ξ is a random variable referring to a sample data. These problems typically arise in deep neural network training in the synchronous data-parallel distributed setting, where each worker has a local copy of the DNN model. Each worker uses one of n non-intersecting partitions of the data, D_i, to jointly update the model parameters $x \in \mathbb{R}^d$, typically the weights and biases of a DNN model. In this paradigm, the objective function $f(x)$ is non-convex but has Lipschitz-continuous gradient. One of the most popular algorithms for solving (1) is the stochastic gradient descent (SGD) algorithm (Robbins and Monro 1951). For a sequence of iterates $\{x_k\}_{k\geq 0}$ and a step-size parameter $\eta_k > 0$ (also called learning rate), SGD iterates are of the form: $x_{k+1} = x_k - \eta_k g(x_k)$, where $g(x_k)$ is an unbiased estimator of the gradient of f, that is, for a given x_k we have $\mathbb{E}(g(x_k)) = \nabla f(x_k)$.

Notations. We write the matrices in bold uppercase letters and denote vectors and scalars by simple lowercase letters. We denote a vector norm of $x \in \mathbb{R}^d$ by $\|x\|$, the ℓ_1-norm by $\|x\|_1$, the ℓ_2-norm by $\|x\|_2$, and for a positive definite matrix $\mathbf{M}$, we define $\|x\|_{\mathbf{M}} := \sqrt{x^\top \mathbf{M} x}$. By $x_k^{i,j}$, we denote a vector that results from the $k^{\rm th}$ iteration, at the $i^{\rm th}$ worker, and represents the $j^{\rm th}$ layer of the deep neural network. Similar notation follows for the stochastic gradients. When j is implied, we simplify the notation as x_k^i and vice-versa. Also, by $x_k^{1:n}$ we denote a collection of n vectors x_k^i, where $i = 1, \ldots, n$. Further, for the ease of notation, denote $f(x_k) = f_k$.

Algorithm 1 Layer-wise gradient compression framework

Input: Number of workers n, learning rate η, compression operators Q_W (worker side) and Q_M (master side)
Output: The trained model x

On each worker i:
for $k = 0, 1, \ldots$ **do**
  Calculate stochastic gradient $g_k^{i,j}$ of each layer j
  $\tilde{g}_k^{i,j} = Q_{W|j}(g_k^{i,j})$
  Send compressed gradient $\tilde{g}_k^{i,j}$
  Receive aggregated gradient $\tilde{g}_k^j$
  Collate entire-model gradient $\tilde{g}_k = \tilde{g}_k^{1:L}$
  $x_{k+1} = x_k - \eta_k \tilde{g}_k$
end for
return x

On master node, at each step k and for each layer j:
Receive $\tilde{g}_k^{i,j}$ from each worker
$\tilde{g}_k^j = Q_{M|j}(\frac{1}{n}\sum_{i=1}^n \tilde{g}_k^{i,j})$
Broadcast aggregated compressed gradient $\tilde{g}_k^j$

3 Layer-wise Gradient Compression

We define a general bidirectional compression framework that is instantiated via two classes of user-provided functions: (1) a compression operator Q_W at each *worker* (which for simplicity, we assume is the same at each worker and for every layer), and (2) a compression operator Q_M at the *master* node. The master node abstracts a set of parameter servers. We now introduce the framework and then formalize the setup under which we analyze it. Our analysis follows in the next section.

Conceptually, based on its local copy of the model at step k, each worker first computes the local stochastic gradient $g_k^{i,j}$ of each layer j (from 1 to L) and then performs layer-wise compression to produce $\tilde{g}_k^{i,j} = Q_{W|j}(g_k^{i,j})$. After that, each worker transmits $\tilde{g}_k^{i,j}$ to the master. The master collects all the gradients from the workers, aggregates them (via averaging), and then uses the compression operator Q_M to generate $\tilde{g}_k^j := Q_{M|j}(\frac{1}{n}\sum_{i=1}^n \tilde{g}_k^{i,j})$. The master then broadcasts the results back to all workers. Each worker recovers the entire-model gradient $\tilde{g}_k$ by collating the aggregated gradient of each layer $\tilde{g}_k^j$ and then updates the model parameters via the following rule (where η is the learning rate):

$$x_{k+1} = x_k - \eta_k \tilde{g}_k. \tag{2}$$

This process continues until convergence. Algorithm 1 lists the steps of this process.

We note that this framework is agnostic to the optimization algorithm. We consider SGD in this paper. However, given access to $x_k^{i,j}$ and $\tilde{g}_k^j$, Algorithm 1 can be adapted to any other popular optimizer used to train DNNs, such as ADAM (Kingma and Ba 2015), ADAGrad (Duchi, Hazan, and Singer 2011) or RMSProp.

Moreover, the framework supports different compression operators at the worker side and master side as our general theory supports it. In the limit, the compression operator may also differ between layers, including the identity function as an operator for specific layers to avoid compressing those. This is also covered by our theory.

Finally, while we cast our framework on the parameter-server architecture, it is easy to see that it generalizes to collective routines (specifically, `all_reduce`) since in that case, there is no master and this behavior is modeled by taking Q_M as the identity function.

3.1 Setup

We now formalize the above concepts and state the general assumptions we make (several of which are classical ones).

Assumption 1. *(Lower bound) The function f is lower bounded; that is, there exists an $f^\star \in \mathbb{R}$ such that $f(x) \geq f^\star$, for all x.*

Assumption 2. *($\mathcal{L}$-smoothness) The function f is $\mathcal{L}$ smooth if its gradient is $\mathcal{L}$-Lipschitz continuous, that is, for all $x, y \in \mathbb{R}^d$, $\|\nabla f(x) - \nabla f(y)\| \leq \mathcal{L}\|x - y\|$.*

Assumption 3. *(Unbiasedness of stochastic gradient) The stochastic gradient is unbiased, that is,*

$$\mathbb{E}(g_k|x_k) = \nabla f_k. \tag{3}$$

If one assumes that the stochastic gradient has bounded variance denoted as Σ, then, for a given symmetric positive definite (SPD) matrix $\mathbf{A}$, one has

$$\mathbb{E}(\|g_k\|_{\mathbf{A}}^2|x_k) = \mathrm{Trace}(\mathbf{A}\Sigma) + \|\nabla f_k\|_{\mathbf{A}}^2,$$

where $\mathrm{Trace}(\mathbf{X})$ denotes the sum of the diagonal elements of a matrix $\mathbf{X}$. A relaxed assumption of the bounded variance is the *strong growth condition on stochastic gradient*.

Assumption 4. *(Strong growth condition on stochastic gradient) For a given SPD matrix $\mathbf{A}$, a general strong growth condition with an additive error is*

$$\mathbb{E}(\|g_k\|_{\mathbf{A}}^2|x_k) \leq \rho\|\nabla f_k\|_{\mathbf{A}}^2 + \sigma^2, \tag{4}$$

where $\rho > 0$ and $\sigma > 0$.

A similar assumption was proposed in (Vaswani, Bach, and Schmidt 2019; Bottou, Curtis, and Nocedal 2018; Bertsekas and Tsitsiklis 1996) when $\mathbf{A}$ is the identity matrix, that is, for the ℓ_2-norm. For overparameterized models such as DNNs, it is common practice to assume $\sigma = 0$; and the condition says that the growth of stochastic gradients is *relatively* controlled by the gradient ∇f_k (Vaswani, Bach, and Schmidt 2019). That is, there exists a $\rho > 0$ such that

$$\mathbb{E}(\|g_k\|_{\mathbf{A}}^2) \leq \rho\|\nabla f_k\|_{\mathbf{A}}^2.$$

Before defining compression operators formally, below we introduce an assumption that compressor operators should obey. Consider a compression operator $Q(\cdot) : \mathbb{R}^d \to \mathbb{R}^d$.

Assumption 5. *(Compression operator) For all vectors $x \in \mathbb{R}^d$ the compression operator $Q(\cdot)$ satisfies*

$$\mathbb{E}_Q\|Q(x)\|_2^2 \leq (1 + \Omega)\|x\|_2^2. \tag{5}$$

where the expectation $\mathbb{E}_Q(\cdot)$ is taken over the internal randomness of the operator $Q(\cdot)$ and $\Omega > 0$.

Remark 1. A broad range of compression operators, whether biased or unbiased, satisfy Assumption 5. In particular, existing compression operators such as Random k, Top k, signSGD, unbiased Random k, QSGD, $\mathcal{C}_{\rm NAT}$, TernGrad, stochastic rounding, adaptive compression, respect this assumption. Moreover, if there is no compression, then $\Omega = 0$.

Following (Bergou, Gorbunov, and Richtárik 2019), we generalize their assumption (c.f. Assumption 3.3) that imposes a descent property to the stochastic gradient. This assumption lower bounds the expected inner product of the stochastic gradient $\tilde{g}_k$ with the gradient ∇f_k with a positive quantity depending on a power of the gradient norm while allowing a small residual on the lower bound.

Assumption 6. *There exists $0 < \alpha \leq 2$ such that*

$$\mathbb{E}\left[\tilde{g}_k^\top \nabla f_k\right] \geq \mathbb{E}\|\nabla f_k\|^\alpha + R_k, \tag{6}$$

where $\|\cdot\|$ is a vector norm in $\mathbb{R}^d$ and R_k is a small scalar residual which may appear due to the numerical inexactness of some operators or due to other computational overheads.

By setting $\alpha = 1$ and $R_k = 0$, we recover the key assumption made by (Bergou, Gorbunov, and Richtárik 2019).

In light of our framework and the assumptions made, we now define a general *layer-wise compression operator*.

Definition 1. (Layer-wise compression operator.) Let $Q(\cdot) : \mathbb{R}^d \to \mathbb{R}^d$ be a layer-wise compression operator such that $Q := (Q_1 \; Q_2 \cdots Q_L)$, where each $Q_j(\cdot) : \mathbb{R}^{d_j} \to \mathbb{R}^{d_j}$, for $j = 1, 2, \cdots, L$ with $\sum_{j=1}^L d_j = d$ be a compression operator.

The following lemma characterizes the compression made by biased layer-wise compression operators.

Lemma 1. *Let $Q(\cdot) : \mathbb{R}^d \to \mathbb{R}^d$ be layer-wise biased compression operator with $Q := (Q_1 \; Q_2 \cdots Q_L)$, such that, each $Q_j(\cdot) : \mathbb{R}^{d_j} \to \mathbb{R}^{d_j}$ for $j = 1, 2, \cdots, L$ satisfies Assumption 5 with $\Omega = \Omega_j$. Then we have*

$$\begin{aligned} \mathbb{E}_Q\left(\|Q(x)\|_2^2\right) &\leq \sum_{1\leq j\leq L}(1+\Omega_j)\|x^j\|_2^2 \\ &\leq \max_{1\leq j\leq L}(1+\Omega_j)\|x\|_2^2. \end{aligned} \tag{7}$$

4 Convergence Analysis

We now establish the convergence of the above-defined layer-wise bidirectional compression scheme. The proofs are available in a companion technical report (Dutta et al. 2019). Let the matrix $\mathbf{W}_W := \mathrm{diag}((1+\Omega_W^1)\mathbf{I}_1 \; (1+\Omega_W^2)\mathbf{I}_2 \cdots (1+\Omega_W^L)\mathbf{I}_L)$ be a diagonal matrix that characterizes the layer-wise compression at each *worker*, such that for each $j = 1, 2, \cdots, L$, $\mathbf{I}_j$ be a $d_j \times d_j$ identity matrix. Similarly, to characterize the layer-wise compression at the *master* node, we define $\mathbf{W}_M$. Given that $\Omega_W^j, \Omega_M^j \geq 0$ for each $j = 1, 2, \cdots, L$, therefore, $\mathbf{W}_W$ and $\mathbf{W}_M$ are SPD matrices. Denote $\Omega_M := \max_{1\leq j\leq L} \Omega_M^j$, $\Omega_W := \max_{1\leq j\leq L} \Omega_W^j$. Further define $\mathbf{A} := \mathbf{W}_M\mathbf{W}_W$.

In the next lemma, we consider several compression operators that satisfy Assumption 6. For instance, these include unbiased compression operators, as well as Random k and signSGD.

Lemma 2. *We note the following:*

i. (For unbiased compression) If $\tilde{g}_k$ is unbiased (the case when Q_M and Q_W are unbiased), then

$$\mathbb{E}\left[\tilde{g}_k^\top \nabla f_k\right] = \mathbb{E}\|\nabla f_k\|_2^2. \tag{8}$$

Therefore, $\tilde{g}_k$ satisfies Assumption 6 with $\alpha = 2$, $\|\cdot\| = \|\cdot\|_2$ and $R_k = 0$.

ii. If Q_M and Q_W are the Random k compression operator with sparsification ratios k_M and k_W, respectively, then

$$\mathbb{E}\left[\tilde{g}_k^\top \nabla f_k\right] = \tfrac{k_M k_W}{d^2}\mathbb{E}\|\nabla f_k\|_2^2. \tag{9}$$

Therefore, $\tilde{g}_k$ satisfies Assumption 6 with $\alpha = 2$, $\|\cdot\| = \frac{k_M k_W}{d^2}\|\cdot\|_2$, and $R_k = 0$.

iii. Let Q_M be the layer-wise Random k_{M_j} compression operator for each layer j. Similarly, Q_W is the layer-wise Random k_{W_j} compression operator for each layer j, then

$$\mathbb{E}\left[\tilde{g}_k^\top \nabla f_k\right] = \mathbb{E}\|\nabla f_k\|_{\mathbf{B}}^2, \tag{10}$$

where $\mathbf{B} = \mathrm{diag}\left(\frac{k_{M_1}k_{W_1}}{d_1^2}\mathbf{I}_1 \; \frac{k_{M_2}k_{W_2}}{d_2^2}\mathbf{I}_2 \cdots \frac{k_{M_L}k_{W_L}}{d_L^2}\mathbf{I}_L\right)$. Therefore, $\tilde{g}_k$ satisfies Assumption 6 with $\alpha = 2$, $\|\cdot\| = \|\cdot\|_{\mathbf{B}}$ and $R_k = 0$.

iv. Let Q_M and Q_W be the sign *function, similar to that in signSGD, then*

$$\mathbb{E}\left[\tilde{g}_k^\top \nabla f_k\right] \geq \mathbb{E}\|\nabla f_k\|_1 + R_k. \tag{11}$$

Therefore, $\tilde{g}_k$ satisfies Assumption 6 with $\alpha = 1$, $\|\cdot\| = \|\cdot\|_1$, and $R_k = \mathcal{O}\left(\frac{1}{BS}\right)$, where BS is the size of used batch to compute the signSGD.

Similar to the cases mentioned in Lemma 2, we can characterize several other well-known compression operators or their combinations. Our next lemma gives an upper bound on the compressed stochastic gradient $\tilde{g}_k$.

Lemma 3. *Let Assumption 4 hold. With the notations defined above, we have*

$$\mathbb{E}\|\tilde{g}_k\|_2^2 \leq \rho\|\nabla f_k\|_{\mathbf{A}}^2 + \sigma^2. \tag{12}$$

Remark 2. If g_k^i has bounded variance, Σ, say, then $\mathrm{Trace}(\mathbf{A}\Sigma) = \sigma^2$.

Now we quote our first general inequality that the iterates of (2) satisfy. This inequality does not directly yield convergence of the scheme in (2). However, this is a first necessary step to show convergence. We note that the matrix $\mathbf{A}$ and σ quantify layer-wise compression.

Proposition 1. *With the notations and the framework defined before, the iterates of* (2) *satisfy*

$$\begin{aligned} \eta_k\left(\mathbb{E}\|\nabla f_k\|^\alpha - \tfrac{\mathcal{L}\eta_k}{2}\mathbb{E}\|\nabla f_k\|_{\mathbf{A}}^2\right) &\leq \mathbb{E}(f_k - f_{k+1}) \\ &\quad -\eta_k R_k + \tfrac{\mathcal{L}\eta_k^2\sigma^2}{2}. \end{aligned} \tag{13}$$

Remark 3. If $\tilde{g}_k$ is an unbiased estimator of the gradient, then $\alpha = 2$, $\|\cdot\| = \|\cdot\|_2$, $\mathbf{A} = \mathbf{I}$, and $R_k = 0$. Therefore, (13) becomes

$$\eta_k\mathbb{E}\|\nabla f_k\|_2^2\left(1 - \tfrac{\mathcal{L}\rho\eta_k}{2}\right) \leq \mathbb{E}(f_k - f_{k+1}) + \tfrac{\mathcal{L}\eta_k^2\sigma^2}{2}.$$

The above is the classic inequality used in analyzing SGD.

In the non-convex setting, it is standard to show that over the iterations the quantity $\min_{k\in[K]} \mathbb{E}(\|\nabla f_k\|^2)$ approaches to 0 as $K \to \infty$. Admittedly, this is a weak statement as it only guarantees that an algorithm converges to a local minimum of f. To facilitate this, next we quote two propositions: one is for the special case when $\alpha = 2$; the other one covers all cases $\alpha \in (0, 2]$. In the propositions, for simplicity, we use a fixed step-size η. One can easily derive the convergence of Algorithm 1 under general compression Q to the ϵ-optimum by choosing a sufficiently small or decreasing step-size, similarly to the classical analysis of SGD.

Proposition 2. *(Special case.) Consider $\alpha = 2$, and $R_k = 0$. Let $C > 0$ be the constant due to the equivalence between the norms $\|\cdot\|$ and $\|\cdot\|_{\mathbf{A}}$, and $K > 0$ be the number of iterations. If $\eta_k = \eta = \mathcal{O}\left(\frac{1}{\sqrt{K}}\right) < \frac{2}{LC\rho}$ then*

$$\frac{\sum_{k=1}^{K} \mathbb{E}\|\nabla f_k\|^2}{K} \leq \mathcal{O}\left(\frac{1}{\sqrt{K}}\right).$$

Proposition 3. *(General case.) Assume $\|\nabla f_k\| \leq G$. Let $C > 0$ be the constant coming from the equivalence between $\|\cdot\|$ and $\|\cdot\|_{\mathbf{A}}$. Let $\eta_k = \eta = \mathcal{O}\left(\frac{1}{\sqrt{K}}\right) < \frac{2}{\mathcal{L}C\rho G^{2-\alpha}}$. Let $a := \left(1 - \frac{\mathcal{L}C\rho G^{2-\alpha}\eta}{2}\right) > 0$, then*

$$\frac{\sum_{k=1}^{K} \mathbb{E}\|\nabla f_k\|^\alpha}{K} \leq \mathcal{O}\left(\frac{1}{\sqrt{K}}\right) + \frac{\sum_{k=1}^{K} R_k}{aK}.$$

Remark 4. Note that if we assume small residuals such that for all k, $R_k = \mathcal{O}\left(\frac{1}{\sqrt{K}}\right)$, then we have $\frac{\sum_{k=1}^{K} \mathbb{E}\|\nabla f_k\|^\alpha}{K} \leq \mathcal{O}\left(\frac{1}{\sqrt{K}}\right)$.

Remark 5. From the above propositions, immediately one can observe, $K \min_{k\in[K]} \mathbb{E}(\|\nabla f_k\|^\alpha) \leq \sum_{k=1}^{k} \mathbb{E}(\|\nabla f_k\|^\alpha)$ and hence the above propositions directly imply convergence of the iterative scheme in (2) under layer-wise compression.

Remark 6. Note that for all the cases we mentioned in Lemma 2, except for signSGD, we have $\alpha = 2$, and $R_k = 0$, so we are in Proposition 2. For signSGD, $|R_k| \leq \mathcal{O}(1/BS)$ so if one uses $BS = \mathcal{O}(\sqrt{K})$, then we get $\frac{\sum_{k=1}^{K} \mathbb{E}\|\nabla f_k\|^\alpha}{K} \leq \mathcal{O}\left(\frac{1}{\sqrt{K}}\right)$.

We note that our convergence analysis can be extended to convex and strongly convex cases.

Layer-wise compression vs entire-model compression. From our analysis, a natural question arises: can we relate the noise for layer-wise compression with that of entire-model compression? The answer is yes and we give a sharper estimate of the noise bound. From our convergence results, we see that the error for layer-wise compression is proportional to $\text{Trace}(\mathbf{A}) = \sum_{j=1}^{L}(1 + \Omega_M^j)(1 + \Omega_W^j)$. This is less than or equal to $L \max_j(1+\Omega_M^j)(1+\Omega_W^j)$, which is the error for using bidirectional compression applied to the entire model.

5 Empirical Study

We implement several well-known compression methods and show experimental results contrasting layer-wise with entire-model compression for a range of standard benchmarks.

5.1 Implementation highlights

We base our proof-of-concept implementation on PyTorch.[1] Layer-wise and entire-model compression share the same compression operations. The difference is the inputs used with each invocation of the compressor. As with other modern toolkits, in PyTorch, the gradients are computed layer-by-layer during the backward pass, starting from the last layer of the model. As such, layer j's gradient is available as soon as it is computed and before backward propagation for layer $j-1$ starts. Distributed training typically exploits this characteristic to accelerate training by overlapping some amount of communication with the still ongoing gradient computation. In the layer-wise setting, our implementation invokes the compressor independently for the gradient of each layer as soon as it is available. In contrast, in the entire-model setting, our implementation waits until the end of the backward pass and invokes the compressor once with the entire model gradients as input. Clearly, this introduces an additional delay before communication starts; however, with smaller volumes of transmitted data, the benefits of compressed communication can eclipse this performance penalty.

5.2 Experimental setting

Compression methods. We experiment with the following compression operators: Random k, Top k, Threshold v, TernGrad (Wen et al. 2017), Adaptive Threshold (Chen et al. 2018), and QSGD (Alistarh et al. 2017). Given an input vector, Random k uniformly samples $k\%$ of its elements; Top k selects the largest $k\%$ elements by magnitude; Threshold v selects any element greater or equal to v in magnitude.

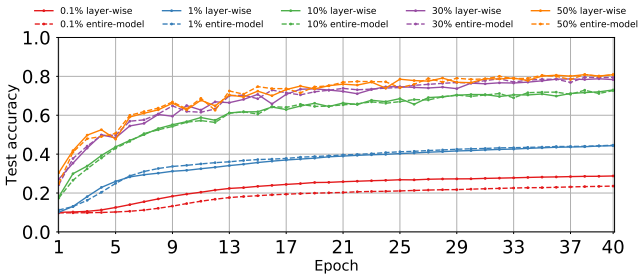
Benchmarks. We adopt DAWNBench (Coleman et al. 2017) as a benchmark for image classification tasks using convolutional neural networks (CNNs). We train AlexNet (Krizhevsky, Sutskever, and Hinton 2012), ResNet-9 and ResNet-50 (Kaiming et al. 2016) models. We use standard CIFAR-10 (Krizhevsky 2009) and ImageNet (Deng et al. 2009) datasets.

Hyperparameters. We set the global mini-batch size to 1,024 (the local mini-batch size is equally divided across workers); the learning rate schedule follows a piecewise-linear function that increases the learning rate from 0.0 to 0.4 during the first 5 epochs and then decreases to 0.0 till the last epoch. Unless otherwise noted, CIFAR-10 experiments run for 24 epochs and ImageNet experiments for 34 epochs. Where applicable, we use ratio k in $\{0.1, 1, 10, 30, 50\}\%$.

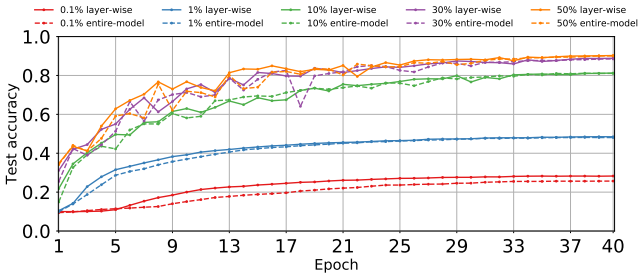
Environment. We perform our experiments on server-grade machines running Ubuntu 18.04, Linux 4.15.0-54, CUDA 10.1 and PyTorch 1.2.0a0_de5a481. The machines are equipped with 16-core 2.6 GHz Intel Xeon Silver 4112 CPU, 512 GB of RAM and 10 Gbps network interface cards. Each machine has an NVIDIA Tesla V100 GPU with 16 GB of memory. We use two machines for CIFAR-10 experiments while we use four machines for ImageNet experiments.

Evaluation metrics. We report the accuracy on a held-out testing dataset evaluated at the end of each epoch during the training process. We compare the test accuracy of layer-wise and entire-model compression.

[1] Available at https://github.com/sands-lab/layer-wise-aaai20.



(a) AlexNet



(b) ResNet-9

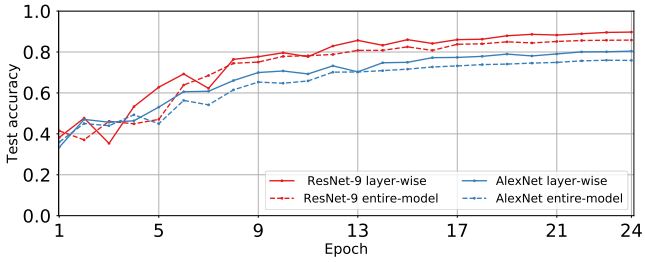
Figure 2: CIFAR-10 test accuracy for Random k compression.

5.3 Experimental Results

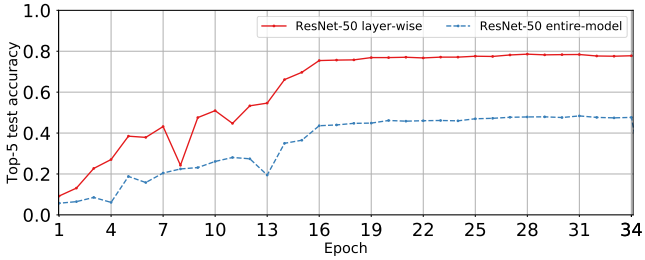
Below we illustrate the results for each compression method. In a nutshell, our results show that both layer-wise and entire-model compression approaches achieve in most cases similar convergence behavior and test accuracy. However, certain compression methods, namely, TernGrad, QSGD, and Adaptive Threshold achieve significantly better accuracy using layer-wise compression. This is because per-layer compression in these cases capitalizes on more fine-grained representations that reduce the overall compression error.

Random k. Figure 2 reports results for Random k compression while training AlexNet and ResNet-9 on CIFAR-10. We observe that layer-wise Random k achieves comparable results to entire-model compression at different sparsification ratios, except for ratio of 0.1% where layer-wise supersedes entire-model compression. This is not surprising because both layer-wise and entire-model compression approaches sample uniformly at random gradient elements, and so, every element has an equal probability of being sampled regardless of its magnitude. We also notice that for ratios less than 10%, Random k has a slower rate of convergence for both compression approaches compared to other compression methods (shown below). This suggests that randomly selecting a sample of gradient elements with no regard to their importance is not ideal, especially for small sparsification ratios.

TernGrad. Figure 3 presents the results of TernGrad compression for several benchmarks. We observe that with TernGrad, layer-wise compression achieves consistently higher test accuracy compared to entire-model compression. Mostly this result is attributed to the following. As an unbiased quantization method, TernGrad scales the gradient values (i.e., three numerical levels $\{-1, 0, 1\}$) by a scalar computed as a function of the gradient vector and its size. For entire-model compression, there is a single scalar and this may be looser than each layer's scalar used in layer-wise compression. Thus, when the model is updated (line 8 of Algorithm 1), entire-model compression has higher probability of error.

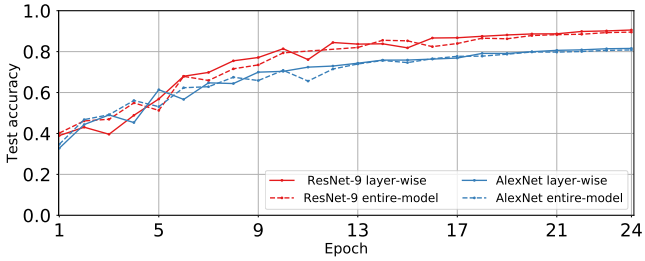


(a) CIFAR-10 test accuracy

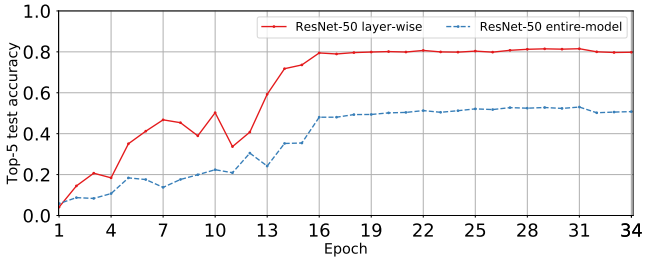


(b) ImageNet top-5 test accuracy

Figure 3: TernGrad compression.



(a) CIFAR-10 test accuracy



(b) ImageNet top-5 test accuracy

Figure 4: QSGD compression (with $s = 256$).

QSGD. The results using QSGD, shown in Figure 4 are similar to TernGrad. We note that ImageNet layer-wise accuracy is $1.52\times$ better than entire-model compression.

Adaptive Threshold. Figure 5 shows the results of Adaptive Threshold compression while training AlexNet and ResNet-9 on CIFAR-10. As before, layer-wise compression achieves

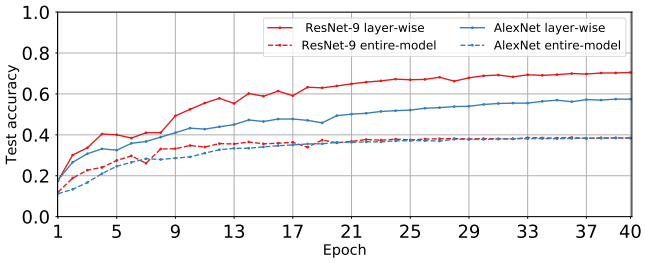


Figure 5: CIFAR-10 test accuracy for Adaptive Threshold.

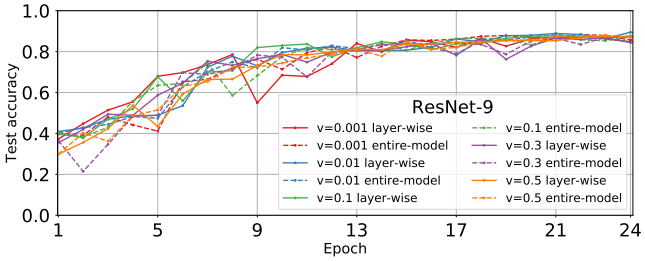


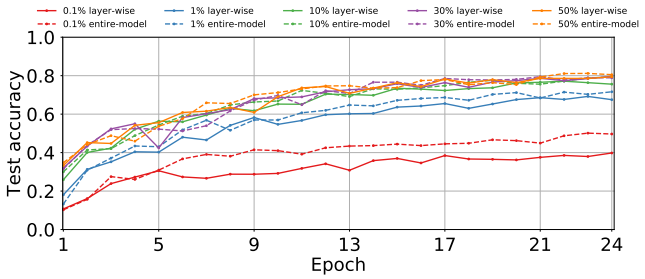
Figure 6: CIFAR-10 test accuracy for Threshold v compression.

better accuracy compared to entire-model compression. The reasoning for this is similar to TernGrad: here, a per-layer threshold chosen with respect to the layer's gradient values performs better than a single threshold selected for the entire-model gradient values. However, we note that this compression method, compared to others, induces slower convergence and only achieves at best $\approx 70\%$ accuracy after 40 epochs.

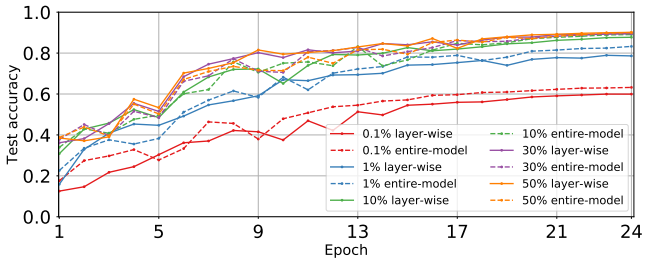
Threshold v. Figures 6 reports results of Threshold v compression while training ResNet-9 on CIFAR-10 (AlexNet is qualitatively similar and omitted). We observe that layer-wise and entire-model compression achieve similar accuracy for a wide range of thresholds across three orders of magnitude. This is expected because every gradient element greater than v in magnitude is transmitted in both approaches.

Top k. Figure 7 presents the results on CIFAR-10 for Top k compression. Similarly to Random k, layer-wise compression achieves comparable results to entire-model compression for a range of sparsification ratios.

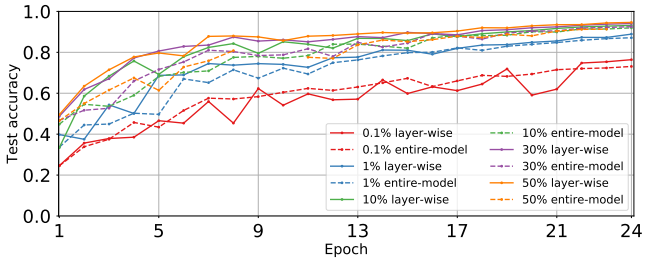
Interestingly, for ratios $\leq 10\%$, where test accuracy is overall low, entire-model compression performs better than layer-wise compression. We could attribute this to the relevance of different layers to the model's convergence, in accordance to a recent study on the relevance of layers in DNNs (Montavon, Samek, and Müller 2018). Unlike layer-wise compression, the top $k\%$ gradient elements picked by entire-model compression could indeed be more important towards the optimization objective, that is, the loss function. Small sparsification ratios and models with a relatively small number of layers stress this behavior. However, to highlight that layer-wise compression is not necessarily inferior in these settings, we repeat the experiment training ResNet-9 by using SGD with Nesterov's momentum (which is outside the scope of our theoretical analysis). Figure 7c shows that



(a) AlexNet



(b) ResNet-9



(c) ResNet-9 by using SGD with Nesterov's momentum

Figure 7: CIFAR-10 test accuracy for Top k compression.

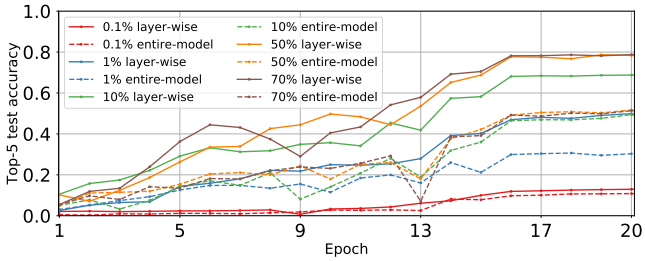


Figure 8: ImageNet top-5 test accuracy for Top k compression.

layer-wise compression is comparable to, if not better than entire-model compression even at small ratios. We leave a thorough analysis of these observations to future work.

Figure 8 shows the top-5 test accuracy for the ResNet-50 model trained on ImageNet (only 20 epochs shown for clarity). Layer-wise compression achieves 1.25-1.63$\times$ better test accuracy. In contrast to CIFAR-10 results, layer-wise compression supersedes entire-model compression even at small ratios (i.e., 0.1%). This reaffirms that layer-wise compression remains more effective for models with a larger number of layers compared to previous experiments.

Training Time. We remark that our focus in this study is the convergence behavior of existing methods. Although total training time is an important factor, we do not present wall-clock results because (1) our unoptimized implementation does not represent a valid proof point for the level of training speedup that well-engineered compression methods can offer, and (2) the literature has already established that there are potentially large speedups to be achieved. Indeed, our measurements show that performance depends on several factors including the model size and depth, the computational costs of compression and layer sizes. While layer-wise compression yields opportunities for overlapping communication and computation, we note that in some cases, overheads are better amortized by combining multiple invocations of the communication routines into a single one. We leave it to future work to thoroughly study the performance implications.

6 Conclusion

We identified a significant discrepancy between the theoretical analysis of the existing gradient compression methods and their practical implementation: while in practice compression is applied layer-wise, theoretical analysis is presumes compression is applied on the entire model. We addressed the lack of understanding of converge guarantees in the layer-wise setting by proposing a bi-directional compression framework that encompasses both biased compression methods and unbiased ones as a special case. We proved tighter bounds on the noise (i.e., convergence) induced on SGD optimizers by layer-wise compression and showed that it is theoretically no worse than entire model compression. We implemented many common compression methods and evaluated their accuracy comparing layer-wise compression to entire-model compression. Conforming to our analysis, our results illustrated that in most cases, layer-wise compression performs no worse than entire-model compression, and in many cases it achieves better accuracy, in particular for larger models.

References

Alistarh, D.; Grubic, D.; Li, J.; Tomioka, R.; and Vojnovic, M. 2017. QSGD: Communication-Efficient SGD via Gradient Quantization and Encoding. In *NeurIPS*.

Alistarh, D.; Hoefler, T.; Johansson, M.; Konstantinov, N.; Khirirat, S.; and Renggli, C. 2018. The Convergence of Sparsified Gradient Methods. In *NeurIPS*.

Bergou, E. H.; Gorbunov, E.; and Richtárik, P. 2019. Stochastic Three Points Method for Unconstrained Smooth Minimization. arXiv 1902.03591. http://arxiv.org/abs/1902.03591.

Bernstein, J.; Wang, Y.; Azizzadenesheli, K.; and Anandkumar, A. 2018. signSGD: Compressed Optimisation for Non-Convex Problems. In *ICML*.

Bertsekas, D. P., and Tsitsiklis, J. N. 1996. *Neuro-Dynamic Programming, 1st edition*. Athena Scientific.

Bottou, L.; Curtis, F. E.; and Nocedal, J. 2018. Optimization Methods for Large-Scale Machine Learning. *SIAM Review* 60(2).

Chen, C. Y.; Choi, J.; Brand, D.; Agrawal, A.; Zhang, W.; and Gopalakrishnan, K. 2018. AdaComp: Adaptive Residual Gradient Compression for Data-Parallel Distributed Training. In *AAAI*.

Coleman, C. A.; Narayanan, D.; Kang, D.; Zhao, T.; Zhang, J.; Nardi, L.; Bailis, P.; Olukotun, K.; Ré, C.; and Zaharia, M. 2017. DAWNBench: An End-to-End Deep Learning Benchmark and Competition. In *NeurIPS - ML Systems Workshop*.

Deng, J.; Dong, W.; Socher, R.; Li, L.-J.; Li, K.; and Fei-Fei, L. 2009. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR*.

Dettmers, T. 2016. 8-Bit Approximations for Parallelism in Deep Learning. In *ICLR*.

Duchi, J. C.; Hazan, E.; and Singer, Y. 2011. Adaptive Subgradient Methods for Online Learning and Stochastic Optimization. *JMLR* 12:2121–2159.

Dutta, A.; Bergou, E. H.; Abdelmoniem, A. M.; Ho, C.-Y.; Sahu, N. A.; Canini, M.; and Kalnis, P. 2019. On the Discrepancy between the Theoretical Analysis and Practical Implementations of Compressed Communication for Distributed Deep Learning. Technical report, KAUST. http://hdl.handle.net/10754/660127.

Horváth, S.; Ho, C.-Y.; Horváth, v.; Sahu, A. N.; Canini, M.; and Richtárik, P. 2019. Natural Compression for Distributed Deep Learning. arXiv 1905.10988. http://arxiv.org/abs/1905.10988.

Kaiming, H.; Xiangyu, Z.; Shaoqing, R.; and Jian, S. 2016. Deep Residual Learning for Image Recognition. In *CVPR*.

Kingma, D. P., and Ba, J. 2015. ADAM: A Method for Stochastic Optimization. In *ICLR*.

Krizhevsky, A.; Sutskever, I.; and Hinton, G. E. 2012. Imagenet Classification with Deep Convolutional Neural Networks. In *NeurIPS*.

Krizhevsky, A. 2009. Learning multiple layers of features from tiny images. Technical report, University of Toronto.

Lim, H.; Andersen, D. G.; and Kaminsky, M. 2019. 3LC: Lightweight and Effective Traffic Compression For Distributed Machine Learning. In *SysML*.

Montavon, G.; Samek, W.; and Müller, K.-R. 2018. Methods for interpreting and understanding deep neural networks. *Digital Signal Processing* 73:1–15.

PyTorch. https://pytorch.org/.

Robbins, H., and Monro, S. 1951. A Stochastic Approximation Method. *Annals of Mathematical Statistics* 22(3):400–407.

Seide, F.; Fu, H.; Droppo, J.; Li, G.; and Yu, D. 2014. 1-Bit Stochastic Gradient Descent and Application to Data-Parallel Distributed Training of Speech DNNs. In *INTERSPEECH*.

Shi, S.; Chu, X.; and Li, B. 2019. MG-WFBP: Efficient Data Communication for Distributed Synchronous SGD Algorithms. In *INFOCOM*.

Shi, S.; Wang, Q.; and Chu, X. 2017. Performance Modeling and Evaluation of Distributed Deep Learning Frameworks on GPUs. arXiv 1711.05979. http://arxiv.org/abs/1711.05979.

Tang, H.; Gan, S.; Zhang, C.; Zhang, T.; and Liu, J. 2018. Communication Compression for Decentralized Training. In *NeurIPS*.

TensorFlow. https://tensorflow.org/.

Vaswani, S.; Bach, F.; and Schmidt, M. 2019. Fast and Faster Convergence of SGD for Over-Parameterized Models (and an Accelerated Perceptron). In *AISTATS*.

Wangni, J.; Wang, J.; Liu, J.; and Zhang, T. 2018. Gradient Sparsification for Communication-Efficient Distributed Optimization. In *NeurIPS*.

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 *NeurIPS*.

Zhang, H.; Zheng, Z.; Xu, S.; Dai, W.; Ho, Q.; Liang, X.; Hu, Z.; Wei, J.; Xie, P.; and Xing, E. P. 2017. Poseidon: An Efficient Communication Architecture for Distributed Deep Learning on GPU Clusters. In *USENIX ATC*.