CONTRACTIVE ERROR FEEDBACK FOR GRADIENT COMPRESSION

Anonymous authors
Paper under double-blind review

ABSTRACT

On-device memory concerns in distributed deep learning are becoming more severe due to i) the growth of model size in multi-GPU training, and ii) the adoption of neural networks for federated learning on IoT devices with limited storage. In such settings, this work deals with memory issues emerging with communication efficient methods. To tackle associated challenges, key advances are that i) instead of EFSGD that inefficiently manages memory, the sweet spot of convergence and memory usage can be attained via what is here termed contractive error feedback (ConEF); and, ii) communication efficiency in ConEF should be achieved by biased and allreducable gradient compression. ConEF is validated on various learning tasks that include image classification, language modeling, and machine translation. ConEF saves 80% – 90% of the extra memory in EFSGD with almost no loss on test performance, while also achieving 1.3x – 5x speedup of SGD.

1 INTRODUCTION

Many modern machine learning tasks can be cast as an optimization problem

$$\min_{\mathbf{x}\in\mathbb{R}^d} f(\mathbf{x}) := \frac{1}{N}\sum_{i=1}^{N} \mathbb{E}_{\xi\sim\mathcal{D}}\big[f_i(\mathbf{x};\xi)\big]. \tag{1}$$

Typically, the possibly nonconvex loss function $f_i(\mathbf{x},\xi)$ is continuously differentiable with respect to (w.r.t.) $\mathbf{x}$. We assume that $f^* > -\infty$ is a global minimum of (1). In this work, we will focus on training neural networks (NN) by solving (1) in two important scenarios: i) data parallelism in a multi-GPU setting, and ii) federated learning where clients could be edge devices such as smart phones. In both cases, N denotes the number of workers (GPUs or devices) and f^* is no smaller than 0 given common choices of loss functions such as cross entropy and mean square.

While training NNs is a resource-hungry task, *bandwidth* and *memory* prevent us from fully harvesting the merits of parallelism. Limited bandwidth slows down the speed of information (e.g., gradients) exchange among workers. In multi-GPU settings, it is observed that communication of gradients becomes a bottleneck that significantly drags down the training speed. While for federated learning, the system capacity is usually confined by communication latency between the server and workers. Memory constraints are less studied compared with bandwidth (Sohoni et al., 2019). However, given the trend of an increasing model size in both computer vision and natural language processing, e.g., ViT (0.6B) (Dosovitskiy et al., 2020), Megatron-LM (8.3B) (Shoeybi et al., 2019), T5 (11B) (Raffel et al., 2019), it evidences that memory is becoming an unavoidable constraint for training large models in current GPU generation with 16/32GB memory. For federated learning, it is also necessary to reduce the memory footprint on workers especially for those IoT applications.

In this work, we will focus on the *parsimonious* setting, jointly dealing with limited bandwidth and memory. The priority is still communication efficiency for speeding up training time, otherwise one can simply rely on SGD with almost no additional memory consumption. Many existing works are unable to cope with such a challenging problem because memory is not carefully taken care of.

Communication efficiency. A well-documented approach to alleviate communication bottleneck is to compress the (stochastic) gradients such that training can be made faster with reduced overhead (Seide et al., 2014; Alistarh et al., 2017; Karimireddy et al., 2019). Depending on whether the gradient compressor is biased, these schemes are categorized as follows.

Unbiased compressors maintain the unbiasedness of stochastic gradients at the price of enlarged variance. For example, uniform quantization of stochastic gradients is studied in QSGD (Alistarh et al., 2017). A concurrent work (Wen et al., 2017) focuses on a special case of QSGD by quantizing each entry of the gradient into $\{\pm 1, 0\}$. Other variants include natural quantization (Horvath et al., 2019), non-uniform quantization (Ramezani-Kebrya et al., 2021) and adaptive quantization (Faghri et al., 2020). Another route to obtain an unbiased compressor is through (scaled) gradient sparsification (Wangni et al., 2018) or its generalized form, atomic decomposition (Wang et al., 2018). However, such methods may be practically slow with performance degradation (Vogels et al., 2019). Many unbiased gradient compressors are also hindered by the fact that they are not allreducable, hence NCCL AllReduce with GPU remote direct memory access (RDMA) cannot be applied. This renders them inefficient for the parsimonious setup, especially when the number of workers is large.

Biased gradient compressors, on the other hand, have been more successful in practice since not only do they support a more impressive compression ratio, but also test accuracy comparable with SGD can be obtained (Vogels et al., 2019). Examples of such compressors contain signSGD (Bernstein et al., 2018a;b), top-k, unscaled random-k (Lin et al., 2018; Alistarh et al., 2019), and powerSGD (Vogels et al., 2019), where the last two support AllReduce. Due to the bias, such compressors rely on error feedback (EF) to ensure convergence (Stich et al., 2018; Karimireddy et al., 2019).

Memory concerns in communication efficient methods. Memory concerns are highly entangled with communication efficiency in distributed training. On the one hand, memory footprint can be alleviated through a smaller batch size (Krause et al., 2016; Spring et al., 2019), leading to increased iterations as well as communication rounds per epoch. Hence, communication efficient methods are particularly useful for accelerating training time in such cases. On the other hand, existing methods on gradient compression are challenged by limited memory for several reasons. First, a smaller batch size, which enlarges the variance, degrades the applicability of unbiased gradient compressors. Confirmed in our experiments, the exploded variance typically calls for a small step size which decelerates convergence. Memory is also not handled well in methods with biased gradient compressors due to the need of error feedback, where additional memory the same as model size has to be consumed to keep track of compression error. Note that although no memory footprint is introduced in local-SGD (Stich, 2019), it suffers from blow-up variance as well. More importantly, local-SGD cannot be integrated with ZeRO 3 (Rajbhandari et al., 2020), state of the art method for coping with limited memory, due to partitioned optimizer states among workers.

In this work, we will focus on EF based methods because of the capability of adopting a biased gradient compressor, which is typically contractive thus more robust to the gradient variance in a small batch size setting. The additional memory consumption of EFSGD over SGD is the local error vector, which is delicate and critical for ensuring convergence. Meanwhile, the designed manner for memory efficiency can only come with lightweight and negligible runtime overhead, otherwise it violates the ultimate goal of communication efficient methods. Contractive error feedback is thus introduced to address these challenges. In a nutshell, our contributions can be summarized as:

- To the best of our knowledge, this is the first work that systematically studies memory concerns in communication efficient methods. Contractive error feedback (ConEF), is introduced to effectively manage memory given the preference of an allreducable and biased gradient compressor.
- Convergence of ConEF is established. Our theoretical results suggest a tradeoff between memory efficiency and a faster convergence, and ConEF finds the sweet spot.
- ConEF is capable of saving 80% – 90% additional memory of EFSGD on various learning problems such as image classification, language modeling, and machine translation. With almost the same runtime of EFSGD, ConEF achieves 1.3x – 5x speedup over SGD. Though test performance slightly drops on smaller models, the proposed method is more useful and reliable for larger networks where improved test accuracy over EFSGD is observed.

Notation. Bold lowercase (uppercase) letters denote vectors (matrices); $\|\mathbf{x}\|$ stands for the ℓ_2 norm of $\mathbf{x}$; and $\langle \mathbf{x}, \mathbf{y} \rangle$ denotes the inner product of $\mathbf{x}$ and $\mathbf{y}$. In addition, we use $[\mathbf{x}]_i$ ($[\mathbf{X}]_{i,j}$) to denote the i-th entry of vector $\mathbf{x}$ (i, j-th entry of matrix $\mathbf{X}$).

2 PRELIMINARIES

This section briefly recaps EFSGD (Stich et al., 2018; Karimireddy et al., 2019) listed under Alg. 1. We use $\mathbf{g}_t^i$ to denote the stochastic gradient at $\mathbf{x}_t$ on worker i, and assume that the stochastic gradients are mutually independent among different workers per iteration. In line 5, the scaled

Algorithm 1 EFSGD

1: **Initialize:** $\mathbf{x}_0 \in \mathbb{R}^d, \mathbf{e}_0^i = \mathbf{0} \in \mathbb{R}^d, \forall i, \eta$
2: **for** $t = 0, 1, \ldots, T-1$ **do**
3: assert $\mathbf{x}_t = \mathbf{x}_t^i$ for every worker i
4: **for** worker $i = 1, \ldots, N$ in parallel **do**
5: $\mathbf{p}_t^i = \eta \mathbf{g}_t^i + \mathbf{e}_t^i$
6: $\mathbf{\Delta}_t^i = \mathcal{Q}(\mathbf{p}_t^i)$
7: $\mathbf{\Delta}_t = \text{Aggregate}(\mathbf{\Delta}_t^i, \forall i)$
8: $\mathbf{x}_{t+1} = \mathbf{x}_t - \mathbf{\Delta}_t$
9: $\mathbf{e}_{t+1}^i = \mathbf{p}_t^i - \mathbf{\Delta}_t^i$
10: **end for**
11: **end for**

Algorithm 2 ConEF

1: **Initialize:** $\mathbf{x}_0 \in \mathbb{R}^d, \mathbf{e}_0^i = \mathbf{0} \in \mathbb{R}^d, \forall i, \eta$
2: **for** $t = 0, 1, \ldots, T-1$ **do**
3: assert $\mathbf{x}_t = \mathbf{x}_t^i$ for every worker i
4: **for** worker $i = 1, \ldots, N$ in parallel **do**
5: $\mathbf{p}_t^i = \eta \mathbf{g}_t^i + \mathbf{e}_t^i$
6: $\mathbf{\Delta}_t^i = \mathcal{Q}(\mathbf{p}_t^i)$
7: $\mathbf{\Delta}_t = \text{Aggregate}(\mathbf{\Delta}_t^i, \forall i)$
8: $\mathbf{x}_{t+1} = \mathbf{x}_t - \mathbf{\Delta}_t$
9: $\mathbf{e}_{t+1}^i = \mathcal{C}(\mathbf{p}_t^i - \mathbf{\Delta}_t^i)$
10: **end for**
11: **end for**

stochastic gradient is augmented via accumulated compression error $\mathbf{e}_t^i$, then compressed and communicated. There is no restriction to the bias of gradient compressor $\mathcal{Q}$. Examples of $\mathcal{Q}$ include (scaled) sign, (unscaled) random/top-k, and powerSGD. The "aggregate" in line 7 refers to $\mathbf{\Delta}_t = \frac{1}{N}\sum_i \mathbf{\Delta}_t^i$, and the compression error $\mathbf{e}_{t+1}^i$ is recomputed in line 9 after updating model $\mathbf{x}_t$. More implementation details on communication of gradients, and reasons for the preference of AllReduce, are deferred to Apdx. A.1 to save space.

Besides the widely appreciated allreducable gradient compressors (Vogels et al., 2019), those biased ones are more tailored for settings where the inclination is to rely on a smaller batch size for memory saving. This is because biased gradient compressors are usually contractive; see more details shortly in Assumption 4. However, due to the need of error feedback to ensure convergence, the additional memory consumption of $\mathbf{e}_t^i$ can confine the applicability of biased compressors as well as EFSGD in parsimonious setups, and this issue is somehow omitted by existing works. Other error feedback variants, e.g., (Wu et al., 2018; Basu et al., 2019; Richtárik et al., 2021), also rely on additional vectors to memorize the compression error, hence may benefit from the proposed technique as well.

3 Memory saving via contractive error feedback

To endow EFSGD with memory efficiency, contractive error feedback is introduced in this section. The key idea is to apply another compressor $\mathcal{C}$ on the error vector $\mathbf{e}_t^i$ such that the memory footprint can be mitigated. The proposed method is summarized in Alg. 2, where the name contractive error feedback (ConEF) originates from the fact that the error vector is represented in its compressed format. Note that the encoding and decoding of compression in Alg. 2 are omitted for notational convenience. $\mathcal{Q}$ and $\mathcal{C}$ are adopted to denote the gradient and error compressors, respectively, to highlight their different roles. The gradient compressor $\mathcal{Q}$ is general and a biased one is more recommended for small batch sizes that can squeeze more memory out. An unbiased error compressor $\mathcal{C}$ is preferable since it is helpful for generalization as we shall discuss later in Section 4. Moreover, the implementation of $\mathcal{C}$ has to be lightweight to avoid slowing down runtime. We will focus on theoretical properties first, and practical guidances for the choice of $\mathcal{C}$ are deferred to Section 4.

3.1 Convergence

Convergence of Alg. 2 is established in this subsection under standard assumptions for communication efficient algorithms in the distributed setting (Alistarh et al., 2017; Karimireddy et al., 2019; Stich et al., 2018; Zheng et al., 2019; Abdi & Fekri, 2020; Basu et al., 2019; Alistarh et al., 2019).

Assumption 1. *The objective function $f : \mathbb{R}^d \to \mathbb{R}$ has L-Lipchitz continuous gradients; that is, $\|\nabla f(\mathbf{x}) - \nabla f(\mathbf{y})\| \le L\|\mathbf{x} - \mathbf{y}\|, \forall\, \mathbf{x}, \mathbf{y} \in \mathbb{R}^d$.*

Assumption 2. *The stochastic gradient $\mathbf{g}_t^i$ is unbiased with bounded variance $\mathbb{E}\big[\mathbf{g}_t^i | \mathbf{x}_t\big] = \nabla f(\mathbf{x}_t)$, and $\mathbb{E}[\|\mathbf{g}_t^i - \nabla f(\mathbf{x}_t)\|^2 | \mathbf{x}] \le \sigma^2$. The gradient is also upper bounded $\mathbb{E}[\|\nabla f(\mathbf{x}_t)\|^2] \le G^2$.*

Assumption 3. *(Unbiased error compressor $\mathcal{C}$). For any given $\mathbf{x}$, $\mathbb{E}[\mathcal{C}(\mathbf{x})|\mathbf{x}] = \mathbf{x}$. In addition, there exist $\theta \ge 0$ such that $\mathbb{E}[\|\mathcal{C}(\mathbf{x}) - \mathbf{x}\|^2 | \mathbf{x}] \le \theta \|\mathbf{x}\|^2$.*

Assumption 4. *(Contractive gradient compressor $\mathcal{Q}$). For any given $\mathbf{x}$, there exists $\delta \in (0, 1)$ such that $\mathbb{E}[\|\mathcal{Q}(\mathbf{x}) - \mathbf{x}\|^2 | \mathbf{x}] \le \delta \|\mathbf{x}\|^2$.*

Holding true for many allreducable and biased gradient compressors such as unscaled random-(block)-k and powerSGD, Assumption 4 implies contractivity – the compression error is always smaller than squared norm of the compressed vector. Such compressors are more preferable when working with small batch sizes (for memory saving) since they do not amplify the gradient variance

as explained in Apdx. A.2 and (Stich et al., 2018). Unbiased compressors satisfying Assumption 4 are usually not allreducable; see also Apdx. A.2. The convergence of Alg. 2 is established below.

Theorem 1. *Suppose that Assumptions 1 - 4 hold and T is large. Choosing $\eta = \mathcal{O}\big(\frac{1}{L(\sqrt{T/N}+T^{1/3})}\big)$, Alg. 2 guarantees that*

$$\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\big[\|\nabla f(\mathbf{x}_t)\|^2\big] \leq \mathcal{O}\bigg(\frac{L\big(f(\mathbf{x}_0)-f^*\big)+(\sigma^2+G^2)}{\sqrt{NT}}\bigg)+\mathcal{O}\bigg(\frac{\delta\theta(\sigma^2+G^2)}{\sqrt{NT}}\bigg). \tag{2}$$

The dependence on N demonstrates that linear speedup is achieved by ConEF. The second term in the right hand side of (2) is the additional error term incurred by compression. Next, we compare it with existing works to gain deeper understanding of ConEF. Upon relating Theorem 1 with EFSGD (Karimireddy et al., 2019), the price paid for memory efficiency can be clearly visualized. Since only the single worker case is considered in the EFSGD paper, we let $N = 1$ in Theorem 1 for a fair comparison. From the additional error term of EFSGD, i.e., $\mathcal{O}(\frac{\delta(\sigma^2+G^2)}{T})$, it is observed that ConEF converges slower. However, ConEF still benefits from the locally compressed error vector when further compared to QSGD which does not employ error feedback schemes (Alistarh et al., 2017). Suppose that $\mathcal{Q}$ is unbiased with Assumption 3 satisfied, then the error term in QSGD is $\mathcal{O}\big(\frac{(1+\theta)(\sigma^2+G^2)}{\sqrt{NT}}\big)$. Therefore, ConEF improves the dependence on θ over QSGD. In addition, Theorem 1 is also tighter than (Horváth & Richtárik, 2020) since their error term is $\mathcal{O}(\frac{(1+\delta\theta)(\sigma^2+G^2)}{\sqrt{NT}})$.

ConEF finds the sweet spot. The comparisons above suggest a tradeoff between memory efficiency and a fast convergence. QSGD is at one extreme where no extra memory is needed at the price of slower convergence due to the blown variance introduced by unbiased gradient compressors. Note also that in general biased gradient compressors cannot be applied in QSGD. EFSGD is at another extreme, where the extra memory consumption is induced by $\mathbf{e}_t^i$ with the benefit of a small error term in the convergence rate. Our proposed Alg. 2 lives in between EFSGD and QSGD, it saves memory compared to EFSGD, but incurs an error term smaller than QSGD yet larger than EFSGD.

3.2 Improved ConEF

In this subsection, we show that when $\mathcal{Q}$ (or $\mathcal{C}$) is accurate, the dependence on θ (resp. δ) in Theorem 1 can be improved. The basic idea is to apply contractive error feedback on top of contractive error feedback such that the compression error can be further reduced. The resultant algorithm, improved ConEF (iConEF), is summarized in Alg. 3. Depending on whether $\mathcal{C}$ or $\mathcal{Q}$ is used twice, there are two versions of iConEF as marked in lines 9 – 13.

Algorithm 3 iConEF

1: **Initialize:** $\mathbf{x}_0, \tilde{\mathbf{e}}_0^i = \mathbf{0}, \mathbf{q}_0^i = \mathbf{0}, \eta$
2: **for** $t = 0, 1, \ldots, T-1$ **do**
3: assert $\mathbf{x}_t = \mathbf{x}_t^i$ for every worker i
4: **for** worker $i = 1, \ldots, N$ in parallel **do**
5: $\mathbf{p}_t^i = \eta\mathbf{g}_t^i + \tilde{\mathbf{e}}_t^i + \mathbf{q}_t^i$
6: $\mathbf{\Delta}_t^i = \mathcal{Q}(\mathbf{p}_t^i)$
7: $\mathbf{\Delta}_t = \text{Aggregate}(\mathbf{\Delta}_t^i, \forall i)$
8: $\mathbf{x}_{t+1} = \mathbf{x}_t - \mathbf{\Delta}_t$
9: **if** version 1 **then**:
10: $\tilde{\mathbf{e}}_{t+1}^i = \mathcal{C}(\mathbf{p}_t^i - \mathbf{\Delta}_t^i)$ ▷ iConEF-v1
11: **else**:
12: $\tilde{\mathbf{e}}_{t+1}^i = \mathcal{Q}(\mathbf{p}_t^i - \mathbf{\Delta}_t^i)$ ▷ iConEF-v2
13: **end if**
14: $\mathbf{q}_{t+1}^i = \mathcal{C}(\mathbf{p}_t^i - \mathbf{\Delta}_t^i - \tilde{\mathbf{e}}_{t+1}^i)$
15: **end for**
16: **end for**

Equipping with an accurate $\mathcal{C}$ with $\theta < 1$, the compression error can be reused to retain performance. Mathematically, this error $\mathbf{p}_t^i - \mathbf{\Delta}_t^i - \tilde{\mathbf{e}}_{t+1}^i$ is once again compressed to reduce the variance of $\tilde{\mathbf{e}}_{t+1}^i$; see (20). However, this approach might require to store both $\tilde{\mathbf{e}}_{t+1}^i$ and $\mathbf{q}_{t+1}^i$, thus less efficient in terms of memory consumption compared to Alg. 2 with the exception of cases where $\mathcal{C}$ is a linear compressor such as count sketch.

Theorem 2. *Suppose that Assumption 3 holds with $0 \leq \theta < 1$. Given Assumptions 1, 2 and 4, and choosing $\eta = \mathcal{O}\big(\frac{1}{L(\sqrt{T/N}+T^{1/3})}\big)$, iConEF-v1 guarantees that*

$$\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\big[\|\nabla f(\mathbf{x}_t)\|^2\big] \leq \mathcal{O}\bigg(\frac{L\big(f(\mathbf{x}_0)-f^*\big)+(\sigma^2+G^2)}{\sqrt{NT}}\bigg)+\mathcal{O}\bigg(\frac{\delta\theta^2(\sigma^2+G^2)}{\sqrt{NT}}\bigg).$$

The θ^2 in the last term tightens Theorem 1 showcasing the benefit of the third compressor. Examples of an accurate $\mathcal{C}$ satisfying $0 \leq \theta < 1$ include quantization working in "dense regime" (Alistarh

et al., 2017). In particular, when the quantization level is chosen as $s = 2\sqrt{d}$, we have $\theta = \frac{1}{4}$. Note also that it is possible to encode a compressed vector with less than $3.44d + 32$ bits (Alistarh et al., 2017, Lemma A.6). This means the memory reduction is almost 90%. Another example is natural compression that can offer at least 3.5x memory saving with $\theta \leq \frac{1}{8}$ (Horvath et al., 2019).

Similarly, when $\mathcal{Q}$ is an accurate compressor, i.e., δ small in Assumption 4, the dependence on δ can be improved through Alg. 3, whose convergence is established next.

Theorem 3. *Suppose that Assumptions 1 – 4 hold. Choosing* $\eta = \mathcal{O}\big(\frac{1}{L(\sqrt{T/N}+T^{1/3})}\big)$*, iConEF-v2 guarantees that*

$$\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\big[\|\nabla f(\mathbf{x}_t)\|^2\big] \leq \mathcal{O}\bigg(\frac{L\big(f(\mathbf{x}_0)-f^*\big)+(\sigma^2+G^2)}{\sqrt{NT}}\bigg) + \mathcal{O}\bigg(\frac{\delta^2\theta(\sigma^2+G^2)}{\sqrt{NT}}\bigg).$$

4 A CLOSER LOOK AT ERROR COMPRESSORS

Previous sections have coped with contractive error feedback in its general form. In what follows, we will discuss the error compressor $\mathcal{C}$ in detail to gain more insights and practical guidances.

4.1 GENERALIZATION PROPERTIES

The merits of an unbiased $\mathcal{C}$ over a biased one come from generalization properties. To see this point, consider overparameterized least square problems (Wilson et al., 2017) as an example

$$\min_{\mathbf{x}} f(\mathbf{x}) := \frac{1}{2}\|\mathbf{A}\mathbf{x}-\mathbf{y}\|^2 \tag{3}$$

where $\mathbf{A} \in \mathbb{R}^{n\times d}$ with $d \gg n$ is the data matrix and $\mathbf{y} \in \mathbb{R}^d$ denotes the labels. Since d is larger than the number of data samples n, this overparameterized problem has multiple global minima with zero loss collected in the set $\mathcal{X}^* := \{\mathbf{x}|f(\mathbf{x}) = 0\}$. Define $\mathcal{R}(\mathbf{X})$ as the range of $\mathbf{X}$. Given the fact that any stochastic gradient of (3) lies in $\mathcal{R}(\mathbf{A}^\top)$, it is straightforward that SGD converges to a point in $\mathcal{R}(\mathbf{A}^\top)$ when initialized well. It is further shown that the solution SGD converges to has the smallest norm among $\mathcal{X}^*$, i.e., $\mathbf{x}^* = \arg\min_{\mathbf{x}\in\mathcal{X}^*} = \mathbf{A}^\top(\mathbf{A}\mathbf{A}^\top)^{-1}\mathbf{y}$. This $\mathbf{x}^*$ is also known as the maximum margin solution, which leads to potential generalization benefits (Valiant, 1984; Cortes & Vapnik, 1995). In fact, for problem (3), if the iterates of a converging algorithm lie in $\mathcal{R}(\mathbf{A}^\top)$, finding the maximum margin solution is ensured (Karimireddy et al., 2019, Lemma 9).

EFSGD converges to a point that is sufficiently close to $\mathcal{R}(\mathbf{A}^\top)$ (Karimireddy et al., 2019). This gives the potential generalization benefit when adopting biased gradient compressors $\mathcal{Q}$ compared with e.g., sign-SGD (Bernstein et al., 2018a;b). Next, we show that such a generalization merit is maintained in Alg. 2 and its variants when $\mathcal{C}$ is unbiased.

Theorem 4. *Suppose that* $\mathcal{C}$ *is unbiased. Initialized at* $\mathbf{x}_0 \in \mathcal{R}(\mathbf{A}^\top)$*, ConEF for solving* (3) *ensures* $\mathbb{E}\big[\mathbf{x}_t - \frac{1}{N}\sum_i \mathbf{e}_t^i\big] \in \mathcal{R}(\mathbf{A}^\top)$*; and, iConEF-v1 and v2 guarantee* $\mathbb{E}\big[\mathbf{x}_t - \frac{1}{N}\sum_i \mathbf{q}_t^i - \frac{1}{N}\sum_i \tilde{\mathbf{e}}_t^i\big] \in \mathcal{R}(\mathbf{A}^\top)$.

Note that for ConEF, we have $\|\mathbb{E}[\frac{1}{N}\sum_{i=1}^N \mathbf{e}_t^i]\|^2 \leq \mathbb{E}[\|\frac{1}{N}\sum_{i=1}^N \mathbf{e}_t^i\|^2] = \mathcal{O}(\eta^2)$. This implies that $\mathbf{x}_t$ is close to the maximum margin solution at convergence. We are unable to obtain such a property for a biased $\mathcal{C}$. The potential generalization merits justify the unbiased choice for $\mathcal{C}$.

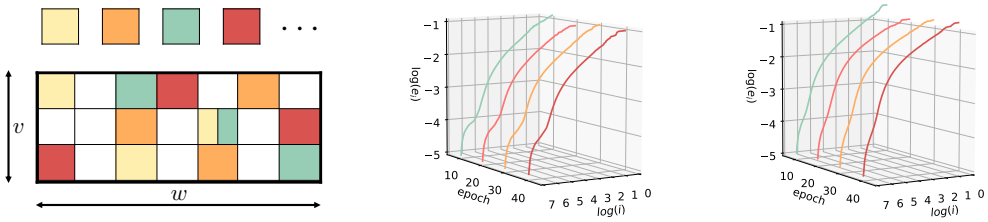
4.2 CHOICES OF ERROR COMPRESSOR

Recall that a prudent error compressor $\mathcal{C}$ for ConEF should i) satisfy Assumption 3; and, ii) have a lightweight implementation to avoid runtime overhead. Two representative examples, quantization and count sketch, will be discussed in detail in this subsection.

4.2.1 QUANTIZATION

Quantization is a large family of compression schemes. For the ease of presentation, here we only describe the basic version (Alistarh et al., 2017). Given s uniformly distributed quantization levels in $[0, 1]$, this approach quantizes the i-the entry of $\mathbf{e}$ as

$$\mathcal{C}([\mathbf{e}]_i) = \|\mathbf{e}\| \cdot \mathrm{sign}([\mathbf{e}]_i) \cdot \xi_i^s([\mathbf{e}]_i) \tag{4}$$



(a) (b) (c)

Figure 1: (a) An illustration of count sketch compressing the first 4 entries of $\mathbf{e}_t^i$. A collision is at $[\mathbf{S}]_{2,5}$ (index starts with 1); (b) magnitude of $\mathbf{e}_t^i$ in EFSGD; (c) magnitude of $\mathbf{e}_t^i$ in partial EFSGD.

where $\xi_i^s(\cdot)$ is a random variable designed to ensure the unbiasedness of $\mathcal{C}([\mathbf{e}]_i)$. In particular, let integer $l \in [0, s]$ such that $\frac{|[\mathbf{e}]_i|}{\|\mathbf{e}\|} \in [\frac{l}{s}, \frac{l+1}{s}]$, and then $\xi_i^s([\mathbf{e}]_i)$ is set to $\frac{l+1}{s}$ with probability $\frac{|[\mathbf{e}]_i|}{\|\mathbf{e}\|}s - l$ and $\frac{l}{s}$ otherwise. Assumption 3 is naturally satisfied according to the proposition below.

Proposition 1. *(Alistarh et al., 2017) For any* $\mathbf{e}$*, quantization in* (4) *satisfies: i)* $\mathbb{E}[\mathcal{C}(\mathbf{e})] = \mathbf{e}$*; ii)* $\mathbb{E}[\|\mathcal{C}(\mathbf{e}) - \mathbf{e}\|^2] \leq \min\{\frac{d}{s^2}, \frac{\sqrt{d}}{s}\}\|\mathbf{e}\|^2$*; and, iii) there exists an encoding scheme that requires* $\tilde{\mathcal{O}}(s^2 + s\sqrt{d})$ *bits in expectation to represent* $\mathcal{C}(\mathbf{e})$.

When setting quantization level $s = \mathcal{O}(1)$, the memory saving is impressive since it only requires $\mathcal{O}(\sqrt{d})$ bits for a d-dimensional vector at the cost of additional implementation burden of e.g., Elias encoding. Stochastic quantization is also suitable for the error compressor $\mathcal{C}$ in iConEF-v1 since it can ensure $\theta < 1$ when $s = \mathcal{O}(\sqrt{d})$.

Other quantization approaches can also be applied as the error compressor. For instance, one can use int-quantization (Mishchenko et al., 2021), where a fp32 number is compressed into int8 or int16; and non-uniform quantization (Ramezani-Kebrya et al., 2021) is also a valid choice.

4.2.2 Count sketch

While quantization is straightforward to come up with, one drawback is that it works on bits rather than words on GPUs. Therefore, we introduce count sketch (Charikar et al., 2002) in this subsection, whose implementation is simply incrementing entries in a matrix.

Count sketch compresses a vector $\mathbf{e}$ by maintaining a matrix $\mathbf{S}$ of size $v \times w < d$, i.e., $\mathcal{C}(\mathbf{e}) := \mathbf{S}$. Both v and w are chosen according to some desirable accuracy guarantees discussed shortly. Count sketch relies on hash functions $h_i : \{1, 2, \ldots, d\} \mapsto \{1, 2, \ldots, w\}$ and random sign functions $s_i : \{1, 2, \ldots, d\} \mapsto \{+1, -1\}$ in i-th row of $\mathbf{S}$. It leverages $[\mathbf{e}]_p$ to update each row i of $\mathbf{S}$ through incrementing $h_i(p)$-th column by $s_i(p)[\mathbf{e}]_p$. Compression of $\mathbf{e}$ can be obtained by updating $\mathbf{S}$ with $\{(p, [\mathbf{e}]_p)\}_{p=1}^d$; see Fig. 1 (a). To decompress and estimate $[\mathbf{e}]_p$ from $\mathbf{S}$, one needs to calculate $\text{Median}(s_i(p) \cdot [\mathbf{S}]_{i,h_i(p)}, \forall i)$. Replacing median with mean also works, but typically median performs better. Count sketch satisfies Assumption 3 as shown in the following proposition.

Proposition 2. *(Charikar et al., 2002) Count sketch is unbiased, i.e.,* $\mathbb{E}[\mathcal{C}(\mathbf{e})] = \mathbf{e}$*. With* $w = \Theta(\frac{1}{\epsilon^2})$ *and* $v = \Theta(\log \frac{d}{p})$*, count sketch ensures* $\|\mathbf{e} - \mathcal{C}(\mathbf{e})\|_\infty \leq \epsilon\|\mathbf{e}\|$ *with probability at least* $1 - p$.

Proposition 2 (Spring et al., 2019) suggests that count sketch is unbiased. Using inequality $\|\mathbf{a}\|_\infty \leq \|\mathbf{a}\| \leq \sqrt{d}\|\mathbf{a}\|_\infty$, it is not difficult to verify that $\|\mathbf{e} - \mathcal{C}(\mathbf{e})\| \leq \epsilon\sqrt{d}\|\mathbf{e}\|$. Although count sketch satisfies a high probability variant of Assumption 3 with $\theta = \epsilon^2 d$, it works well in our experimental studies with the recommendation of $v = 1$ or $v = 3$. Hence, this error compressor is more practical relative to the theoretical bounds. Another useful property of count sketch is its linearity, i.e., $\mathcal{C}(\mathbf{e}_1 + \mathbf{e}_2) = \mathcal{C}(\mathbf{e}_1) + \mathcal{C}(\mathbf{e}_2)$. This is helpful since i) it opens the possibility to squeeze $\tilde{\mathbf{e}}_{t+1}$ and $\mathbf{q}_{t+1}$ in iConEF-v1 to further reduce the memory consumption by adding these two count sketches; and, ii) it eases error reset (Basu et al., 2019; Xie et al., 2020), which synchronizes the local errors every a few iterations among all workers. The second merit is because count sketch is small in size and allreducable, and more on this can be found in Apdx. G.4.

Empirically, count sketch performs best when the compressed vector follows power law[1]. We train ResNet18 on CIFAR10, and plot the magnitude of $\mathbf{e}_t^i$ on the first worker at the end of epochs

[1]The magnitude-index plot is roughly a line in log-log scale.

Table 1: β vs. sketch size for convergence

β	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
smallest sketch size	0.9	0.8	0.7	0.6	0.4	0.4	0.3	0.2	0.1

$\{10, 20, 30, 40\}$ in Fig. 1(b). It is observed that power law is not satisfied. However, in Fig. 1(c) we find that the error vectors in a variant of EFSGD, termed partial EFSGD (Abdi & Fekri, 2020) (see Alg. 4 in Apdx. F due to space limitation), follow broken power law, that is, a piece-wise linear function in the log-log plot. Partial EFSGD adds part of the compression error, i.e., $(1-\beta)\mathbf{e}_t^i$ for $\beta \in [0, 1)$, to stochastic gradients before compressing. As we shall see in numerical experiments, the update in partial EFSGD is helpful to save more memory when working with count sketch.

5 Numerical results

In this section, experimental studies are carried out to validate the proposed algorithms. Since our major goal is to push the limit of memory saving, we will mostly focus on ConEF, and experiments of iConEF can be found in Apdx. G.4. Overall, the experiments are designed to answer three research questions (RQs): 1) how should we choose hyperparameters in ConEF. Since quantization is well-known in community, we put more efforts on the less understood count sketch; 2) when the gradient is aggressively compressed, how much memory can be saved; and, 3) how will the proposed method perform on different learning tasks. Although allreducable gradient compressors are advocated, we also include numerical results for scaled-sign, which requires parameter-server for efficient communication in RQ1. Testing various gradient compressors is also helpful to demonstrate the generality of ConEF. We follow common practice where both gradient and error compressors are applied in their layerwise form. Our implementation utilizes `torch.DistributedDataParallel` so that the bucket trick can be leveraged to overlap gradient computation and communication, and NCCL is chosen as communication backend.

5.1 RQ1: guidance on hyperparameter choice

We focus on the parameter choices of count sketch and quantization in ConEF using scaled-sign gradient compressor $\mathcal{Q}(\mathbf{g}) = \frac{\|\mathbf{g}\|_1}{d}\text{sign}(\mathbf{g})$ as an example. About 97% communication is reduced. This gradient compressor requires parameter-server for communication, hence may put memory burden on the server side. Two-way compression is also required for efficiency (Zheng et al., 2019) desipte it is not the main focus here. We instead avoid it by using AllGather to simulate this setting. The experiments are run with an Amazon EC2 p3.8xlarge instance (4 GPUs), where implementation details are put into Apdx. G.1 given the page limitation. When using allreducable gradient compressors, the general guidance for parameter choices is similar, although the numbers might change; see Apdx. G.1 for illustration of this point. Most of experiments are tested on CIFAR10,[2] which consists of 60000 32x32 images in 10 classes with 50000 training samples and 10000 test ones. Standard data augmentation and preprocessing techniques are employed (He et al., 2016).

We start with comparing contractive error feedback for vanilla and partial EFSGD using count sketches. With $v = 1$, the sketch size has to be 0.95x model size to ensure good convergence for compressing the error vector in vanilla EFSGD. On the other hand, the error vector of partial EFSGD follows the broken power law, and hence is more suitable for count sketches. We list the choice of β and the smallest sketch size for convergence in Table. 1. Note that only $\{0.9, 0.8, 0.7, 0.6, 0.5, 0.4, 0.3, 0.2, 0.1\}$ are tested, though there is no barrier to choose other values. It is observed that a larger β leads to more saved memory. This is possibly because that a larger β helps to suppress the error accumulation in count sketches; see Apdx. F.

Next we focus on choices for v and w in count sketches. We implement the partial EFSGD variant of ConEF (see Alg. 4 in Apdx. F) with memory budget as 0.6x model size. Two specific sketches, $v = 1, w = 0.6$x model size and $v = 3, w = 0.2$x model size, are tested. It can be seen in Fig. 2 that the first sketch with $v = 1$ performs better. This suggests that a larger w, leading to less collision of hash functions, is more numerically beneficial compared with an increased v. Similar observations also appear on allreducable compressors. Hence, $v = 1$ is adopted in most of our experiments.

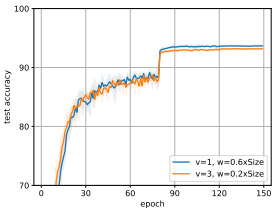


Figure 2: ConEF with different count sketches.

[2]https://www.cs.toronto.edu/ kriz/cifar.html

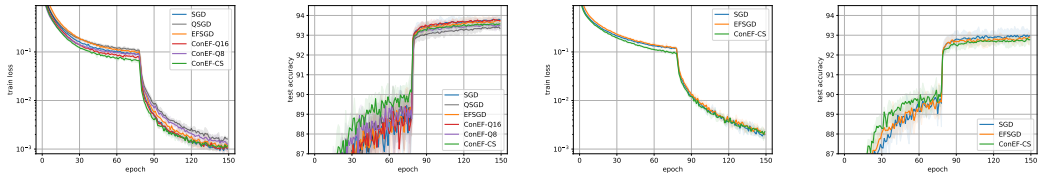


Figure 3: Performance of ConEF on ResNet18 and MobileNetv2. From left to right: (a) train loss and (b) test accuracy on ResNet18; and (c) train loss and (d) test accuracy on MobileNetv2.

After clarifying the parameter choices, the effectiveness of ConEF in parameter-server settings is tested with ResNet18 on CIFAR10 for 150 epochs as shown in Fig. 3. The detailed parameter choices are deferred to Apdx. G.1. We use ConEF-CS, ConEF-Q16 and ConEF-Q8 to denote Alg. 2 with error compressors $\mathcal{C}$ being count sketch, 16-level quantization, and 8-level quantization, respectively. Note that when $\mathcal{C}$ is a stochastic quantizer, we follow common practice to replace the ℓ_2 norm in (4) with max-norm for superior performance (Alistarh et al., 2017). It is observed that on ResNet18, ConEF-Q16 achieves a test accuracy that is almost the same as SGD and EFSGD, demonstrating EFSGD uses unnecessary memory. On the other hand, 44Mb is saved on a 50Mb model with count sketch, while only incurring slight drop of 0.24 on test accuracy compared with SGD. To further validate the potential of ConEF in smart phone based applications, we test MobileNetv2 (Sandler et al., 2018), a more tailored model for the targeted setting. We set $w = 0.1$x model size to save 8Mb on a 14Mb model compared with EFSGD, while almost losing no test accuracy. Another interesting observation is that the convergence of ConEF-CS tends to be faster in the first 80 epochs before decreasing the learning rate. This might be useful in federated learning with IoT devices where each worker may not participate in training for too many iterations.

5.2 RQ2: Memory saving with aggressively compressed gradients

Next we focus on allreducable gradient compressors. We use 16 GPUs on 4 Amazon EC2 p3.8xlarge instances for experiments.

To investigate the performance of ConEF under aggressively compressed gradients, powerSGD is selected (Vogels et al., 2019). PowerSGD finds a rank-r approximation to the gradient matrix $\mathbf{G}$. It is allreducable and different levels of communication efficiency can be achieved by tuning r. This is a challenging setting for ConEF given that i) the performance of the heuristic powerSGD highly depends on $\mathbf{e}_t^i$, and ii) $\mathbf{e}_t^i$ plays a more important role when the gradient is aggressively compressed.

ResNet18 is trained on CIFAR10 with $r = 4$ in powerSGD, which reduces 98.5% communication overhead. Unlike (Vogels et al., 2019), a small batch size, e.g., 16 per worker, is considered here for reducing memory requirement (Krause et al., 2016). Mixed precision trick is also adopted to facilitate memory saving, where we use fp16 numbers on count sketches instead of fp32 (Micikevicius et al., 2018). Two more benchmarks are compared with. *qSGD* (Vogels et al., 2019): SGD with an unbiased and allreducable gradient compressor that has the same communication overhead as powerSGD. In particular, the gradient matrix $\mathbf{G} \in \mathbb{R}^{n\times m}$ is approximated by $(\mathbf{GU})\mathbf{U}^\top$ for a random matrix $\mathbf{U} \in \mathbb{R}^{m\times r}$. Small-sized matrices $\mathbf{GU}$ and $\mathbf{U}^\top$ are communicated. We term such a scheme as qSGD to distinguish it with QSGD (Alistarh et al., 2017). Another benchmark is *hEFSGD*, which is a heuristic method by keeping the top-k elements of $\mathbf{e}_t^i$ in EFSGD for memory saving.

The results can be found in Fig. 4, where we mark the percentage of saved memory with ConEF in the figure legend. There are several observations. ConEF-CS has a similar runtime as EFSGD, and both are 10% faster than SGD. This verifies that the runtime overhead of ConEF is almost negligible. ConEF-CS significantly outperforms qSGD, which confirms the necessity of additional memory to enable good convergence in the memory constrained setting with small batch sizes. When 60% memory is saved, ConEF-CS also outperforms hEFSGD by a large margin in terms of both runtime and test accuracy, validating the generalization merits of an unbiased error compressor (cf. Theorem 4). ConEF-CS has a comparable performance with EFSGD, and more saved memory leads to larger accuracy loss.

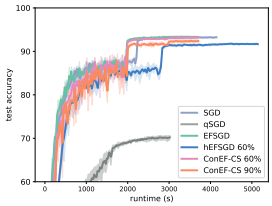


Figure 4: ConEF with a powerSGD gradient compressor.

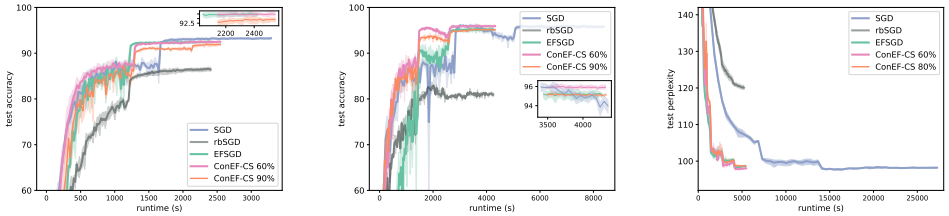


Figure 5: Performance of ConEF with an unscaled random block gradient compressor on: ResNet-18, WideResNet-28-10, and LSTM (left to right).

5.3 RQ3: MEMORY SAVING ON VARIOUS LEARNING TASKS

In this subsection, ConEF is tested on different learning tasks. The (unscaled) random block gradient compressor is adopted, where a continuous block of k coordinates are chosen in a uniformly random fashion to improve 90% communication efficiency. When using the same random seed among GPUs, AllReduce is readily applied. Such a gradient compressor is more suitable for ConEF, since the error vector $\mathbf{e}_t^i$ has $d-k$ zeros. Three different training tasks are considered: i) ResNet-18 on CIFAR10; ii) WideResNet-28-10 on CIFAR10; and iii) LSTM for language modeling on Wikitest-2 (Merity et al., 2016). To demonstrate the necessity of additional memory, an unbiased variant of random block gradient compressor is also implemented, where the unbiasedness is ensured by scaling with importance sampling. We term the corresponding method as *rbSGD*. A setting with small batch sizes is still the main focus here, and the results can be found in Fig. 5.

On ResNet-18, it is observed that EFGSD and ConEF save 22% and 21% training time compared with SGD. The percentage of time saving improves over powerSGD even with a smaller compression ratio here. This is because more time is spent on the complicated encoding in powerSGD. It is observed that ConEF significantly outperforms rbSGD, once again demonstrating the need of additional memory when training with a small batch size of 16. Compared with EFSGD, ConEF-CS drops 0.04 and 0.5 on test accuracy to save 60% and 90% memory, respectively.

For WideResNet-28-10, EFSGD and ConEF are 49.0% and 48.6% faster than SGD. Algorithms with (contractive) error feedback again considerably outperform rbSGD. EFSGD does not catch up with the test accuracy of ConEF-CS with 60% memory saving at a difference of 0.6, suggesting that EFSGD consumes memory inefficiently. ConEF-CS with 90% memory saving only drops 0.1 test accuracy compared with EFSGD. In light of the results on ResNet-18, a larger model (WideResNet) appears to be more robust when used jointly with contractive error feedback.

Next we focus on language modeling with LSTM on WikiText-2. All tested algorithms reduce about 78% training time of SGD. rbSGD fails to match the performance of ConEF-CS and EFSGD. When 60% memory is saved, ConEF-CS outperforms EFSGD by lowering 0.3 of perplexity. ConEF-CS with 80% memory saved performs slightly worse than EFSGD with 0.2 higher perplexity.

We also train a transformer for machine translation on Multi30K (Elliott et al., 2016) in Apdx. G. Adam based heuristic methods with (contractive) error feedback are adopted. In such a case, even 90% memory saved ConEF-CS has slightly better performance than EF-Adam.

Additional experiments. Performance of iConEF-v1 and v2 can be found in Fig. 6. In this experiment, another allreducable gradient compressor, unscaled random-k, is adopted to reduce 90% communication. Both versions of iConEF outperform ConEF, validating our findings in Theorems 2 and 3. Due to space limitation, more implementation details and numerical results regarding random-k gradient compressor are deferred to Apdx. G.4, where we observe that ConEF can reduce 95% additional memory of EFSGD on ResNet18.

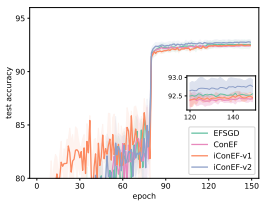


Figure 6: iConEF with a random k gradient compressor.

6 CONCLUSION AND FUTURE WORK

Contractive error feedback (ConEF) is introduced to alleviate memory concerns in communication efficient methods. The key insight is that the error vector can be compressed and an effective error compressor, such as count sketch, saves memory with almost no accuracy drop. ConEF is tested on various image classification and language processing tasks, reducing $80\% - 90\%$ of additional memory footprint in EFSGD. Our next step will focus on saving memory more aggressively.

ETHICS STATEMENT

The central topic of this work is speeding up distributed training on devices with limited memory. Since this problem is tackled from an algorithmic perspective, the impact would be mainly scientific rather than ethical.

REPRODUCIBILITY STATEMENT

All missing proofs can be found in Appendix. Detailed choices for hyperparameters and additional experiments are deferred to Appendix G due to space limitation, and publicly available code will be provided after review process to ensure anonymity.

REFERENCES

Afshin Abdi and Faramarz Fekri. Quantized compressive sampling of stochastic gradients for efficient communication in distributed deep learning. In *Proc. of the AAAI Conf. on Artificial Intelligence*, volume 34, pp. 3105–3112, 2020.

Dan Alistarh, Demjan Grubic, Jerry Li, Ryota Tomioka, and Milan Vojnovic. QSGD: Communication-efficient SGD via gradient quantization and encoding. *Proc. of Neural Information Processing Systems*, 30:1709–1720, 2017.

Dan Alistarh, Torsten Hoefler, Mikael Johansson, Sarit Kririrat, Nikola Konstantinov, and Cedric Renggli. The convergence of sparsified gradient methods. *Proc. of Neural Information Processing Systems*, pp. 5973–5983, 2019.

Debraj Basu, Deepesh Data, Can Karakus, and Suhas Diggavi. Qsparse-local-SGD: Distributed sgd with quantization, sparsification and local computations. *Proc. of Neural Information Processing Systems*, 32, 2019.

Jeremy Bernstein, Yu-Xiang Wang, Kamyar Azizzadenesheli, and Animashree Anandkumar. signSGD: Compressed optimization for non-convex problems. In *Proc. of Intl. Conf. on Machine Learning*, pp. 560–569. PMLR, 2018a.

Jeremy Bernstein, Jiawei Zhao, Kamyar Azizzadenesheli, and Anima Anandkumar. signSGD with majority vote is communication efficient and fault tolerant. In *Proc. of Intl. Conf. on Learning Representations*, 2018b.

Moses Charikar, Kevin Chen, and Martin Farach-Colton. Finding frequent items in data streams. In *Intl. Colloquium on Automata, Languages, and Programming*, pp. 693–703. Springer, 2002.

Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *Proc. of Intl. Conf. on Learning Representations*, 2020.

Desmond Elliott, Stella Frank, Khalil Sima'an, and Lucia Specia. Multi30k: Multilingual english-german image descriptions. In *Proc. of the 5th Workshop on Vision and Language*, pp. 70–74. Association for Computational Linguistics, 2016.

Fartash Faghri, Iman Tabrizian, Ilia Markov, Dan Alistarh, Daniel M Roy, and Ali Ramezani-Kebrya. Adaptive gradient quantization for data-parallel SGD. *Proc. of Neural Information Processing Systems*, 33:3174–3185, 2020.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition*, pp. 770–778, 2016.

Samuel Horváth and Peter Richtárik. A better alternative to error feedback for communication-efficient distributed learning. *arXiv preprint arXiv:2006.11077*, 2020.

Samuel Horvath, Chen-Yu Ho, Ludovit Horvath, Atal Narayan Sahu, Marco Canini, and Peter Richtárik. Natural compression for distributed deep learning. *arXiv preprint arXiv:1905.10988*, 2019.

Sai Praneeth Karimireddy, Quentin Rebjock, Sebastian Stich, and Martin Jaggi. Error feedback fixes signSGD and other gradient compression schemes. In *Proc. of Intl. Conf. on Machine Learning*, pp. 3252–3261. PMLR, 2019.

Ben Krause, Liang Lu, Iain Murray, and Steve Renals. Multiplicative LSTM for sequence modelling. *arXiv preprint arXiv:1609.07959*, 2016.

Yujun Lin, Song Han, Huizi Mao, Yu Wang, and Bill Dally. Deep gradient compression: Reducing the communication bandwidth for distributed training. In *Proc. of Intl. Conf. on Learning Representations*, 2018.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843*, 2016.

Paulius Micikevicius, Sharan Narang, Jonah Alben, Gregory Diamos, Erich Elsen, David Garcia, Boris Ginsburg, Michael Houston, Oleksii Kuchaiev, Ganesh Venkatesh, et al. Mixed precision training. In *Proc. of Intl. Conf. on Learning Representations*, 2018.

Konstantin Mishchenko, Bokun Wang, Dmitry Kovalev, and Peter Richtárik. IntSGD: Floatless compression of stochastic gradients. *arXiv preprint arXiv:2102.08374*, 2021.

Yanghua Peng, Yibo Zhu, Yangrui Chen, Yixin Bao, Bairen Yi, Chang Lan, Chuan Wu, and Chuanxiong Guo. A generic communication scheduler for distributed dnn training acceleration. In *Proc. of ACM Symposium on Operating Systems Principles*, pp. 16–29, 2019.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv preprint arXiv:1910.10683*, 2019.

Samyam Rajbhandari, Jeff Rasley, Olatunji Ruwase, and Yuxiong He. Zero: Memory optimizations toward training trillion parameter models. In *Proc. Intl. Conf. for High Performance Computing, Networking, Storage and Analysis*, pp. 1–16. IEEE, 2020.

Ali Ramezani-Kebrya, Fartash Faghri, Ilya Markov, Vitalii Aksenov, Dan Alistarh, and Daniel M Roy. NUQSGD: Provably communication-efficient data-parallel SGD via nonuniform quantization. *Journal of Machine Learning Research*, 22(114):1–43, 2021.

Peter Richtárik, Igor Sokolov, and Ilyas Fatkhullin. EF21: A new, simpler, theoretically better, and practically faster error feedback. *arXiv preprint arXiv:2106.05203*, 2021.

Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4510–4520, 2018.

Frank Seide, Hao Fu, Jasha Droppo, Gang Li, and Dong Yu. 1-bit stochastic gradient descent and its application to data-parallel distributed training of speech dnns. In *Proc. of Annual Conf. of the Intl. Speech Communication Association*, 2014.

Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, and Bryan Catanzaro. Megatron-lm: Training multi-billion parameter language models using model parallelism. *arXiv preprint arXiv:1909.08053*, 2019.

Nimit Sharad Sohoni, Christopher Richard Aberger, Megan Leszczynski, Jian Zhang, and Christopher Ré. Low-memory neural network training: A technical report. *arXiv preprint arXiv:1904.10631*, 2019.

Ryan Spring, Anastasios Kyrillidis, Vijai Mohan, and Anshumali Shrivastava. Compressing gradient optimizers via count-sketches. In *Proc. of Intl. Conf. on Machine Learning*, pp. 5946–5955. PMLR, 2019.

Sebastian U Stich, Jean-Baptiste Cordonnier, and Martin Jaggi. Sparsified SGD with memory. *Proc. of Neural Information Processing Systems*, 31:4447–4458, 2018.

Sebastian Urban Stich. Local SGD converges fast and communicates little. In *Proc. of Intl. Conf. on Learning Representations*, number CONF, 2019.

Leslie G Valiant. A theory of the learnable. *Communications of the ACM*, 27(11):1134–1142, 1984.

Thijs Vogels, Sai Praneeth Karimireddy, and Martin Jaggi. PowerSGD: Practical low-rank gradient compression for distributed optimization. *Proc. of Neural Information Processing Systems*, 32: 14259–14268, 2019.

Hongyi Wang, Scott Sievert, Shengchao Liu, Zachary Charles, Dimitris Papailiopoulos, and Stephen Wright. ATOMO: Communication-efficient learning via atomic sparsification. *Proc. of Neural Information Processing Systems*, 31:9850–9861, 2018.

Jianqiao Wangni, Jialei Wang, Ji Liu, and Tong Zhang. Gradient sparsification for communication-efficient distributed optimization. *Proc. of Neural Information Processing Systems*, 31:1299–1309, 2018.

Wei Wen, Cong Xu, Feng Yan, Chunpeng Wu, Yandan Wang, Yiran Chen, and Hai Li. TernGrad: Ternary gradients to reduce communication in distributed deep learning. *Proc. of Neural Information Processing Systems*, 30, 2017.

Ashia C Wilson, Rebecca Roelofs, Mitchell Stern, Nathan Srebro, and Benjamin Recht. The marginal value of adaptive gradient methods in machine learning. In *Proc. of Neural Information Processing Systems*, pp. 4151–4161, 2017.

Jiaxiang Wu, Weidong Huang, Junzhou Huang, and Tong Zhang. Error compensated quantized sgd and its applications to large-scale distributed optimization. In *Proc. of Intl. Conf. on Machine Learning*, pp. 5325–5333. PMLR, 2018.

Cong Xie, Shuai Zheng, Oluwasanmi O Koyejo, Indranil Gupta, Mu Li, and Haibin Lin. CSER: Communication-efficient SGD with error reset. *Proc. of Neural Information Processing Systems*, 33, 2020.

Mingchao Yu, Zhifeng Lin, Krishna Narra, Songze Li, Youjie Li, Nam Sung Kim, Alexander Schwing, Murali Annavaram, and Salman Avestimehr. Gradiveq: Vector quantization for bandwidth-efficient gradient aggregation in distributed cnn training. *Proc. of Neural Information Processing Systems*, 31:5123–5133, 2018.

Shuai Zheng, Ziyue Huang, and James Kwok. Communication-efficient distributed blockwise momentum SGD with error-feedback. *Proc. of Neural Information Processing Systems*, 32:11450–11460, 2019.