CocktailSGD: Fine-tuning Foundation Models over 500Mbps Networks

Jue Wang [*1] **Yucheng Lu** [*2] **Binhang Yuan** [1] **Beidi Chen** [3]
Percy Liang [4] **Christopher De Sa** [2] **Christopher Re** [4] **Ce Zhang** [1]

Abstract

Distributed training of foundation models, especially large language models (LLMs), is communication-intensive and so has heavily relied on centralized data centers with fast interconnects. *Can we train on slow networks and unlock the potential of decentralized infrastructure for foundation models?* In this paper, we propose CocktailSGD, a novel communication-efficient training framework that combines three distinct compression techniques—random sparsification, top-K sparsification, and quantization—to achieve much greater compression than each individual technique alone. We justify the benefit of such a hybrid approach through a theoretical analysis of convergence. Empirically, we show that CocktailSGD achieves up to 117× compression in fine-tuning LLMs up to 20 billion parameters without hurting convergence. On a 500Mbps network, CocktailSGD only incurs ∼ 1.2× slowdown compared with data center networks.

1. Introduction

In recent years, foundation models (Bommasani et al., 2021), including large language models (Brown et al., 2020; Chowdhery et al., 2022; Bommasani et al., 2021; Zhang et al., 2022; Liang et al., 2022; Scao et al., 2022), have enabled rapid advancement for various machine learning tasks, especially in natural language processing (Brants et al., 2007; Austin et al., 2021). Such a significant improvement on quality has been fueled by an ever-increasing amount of data and computes that are required in training these models (Kaplan et al., 2020). Today, training even modest scale models requires a significant amount of compute: For example, fine-tuning GPT-J-6B (6 billion parameters) over merely 10 billion tokens would require 6 petaflops-days: 8 A100 GPUs running at 50% capacity for 5 days!

When training foundation models in a distributed way, *communication* is the key bottleneck in scaling. As an example, fine-tuning GPT-J-6B over 10 billion tokens with a batch size of 262K tokens over 4 machines (each with 2 A100 GPUs) would require 915.5 TB data being communicated during the whole training process! The computation time for such a workload is 114 hours, which means that we need to have at least 20 Gbps connections between these machines to bring the communication overhead to the same scale as the computation time. Not surprisingly, today's infrastructure for training and fine-tuning foundation models are largely *centralized*, with GPUs connected via fast 100Gbps–400Gbps connections (Microsoft, 2020).

Such a heavy reliance on centralized networks increases the cost of infrastructure, and makes it incredibly hard to take advantage of cheaper alternatives, including tier 2 to tier 4 clouds, spot instances and volunteer compute. For example, while volunteering compute projects such as Folding@Home can harvest significant amount of computes for embarrassingly parallelizable workloads (e.g., 2.43exaflops in April 2020 (Larson et al., 2009)), it is challenging to harvest these cycles for foundation model training due to the communication bottleneck. Recently, there has been an exciting collection of work focusing on the decentralized training of neural networks, including those that are algorithmic (Lian et al., 2017; Ryabinin & Gusev, 2020; Diskin et al., 2021; Ryabinin et al., 2021; Yuan et al., 2022; Jue et al.) as well as system efforts such as Training Transformer Together (Borzunov et al., 2022b), and PETALS (Borzunov et al., 2022a). However, despite of these recent efforts, communication is still a significant bottleneck, and one can only compress the communication by at most 10-30× in these recent efforts without hurting convergence. To fully close the gap between centralized infrastructure (100Gbps) and decentralized infrastructure (100Mbps-1Gbps), we need to decrease the communication overhead by at least 100×!

Luckily, there have also been rapid development of communication-efficient optimization algorithms and these efforts provide the foundational building blocks of this paper. Researchers have proposed a wide range of

[*]Equal contribution [1]ETH Zürich, Switzerland [2]Cornell University, USA [3]Carnegie Mellon University, USA [4]Stanford University, USA. Correspondence to: Jue Wang <juewang@inf.ethz.ch>.

Proceedings of the 40^{th} International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023.

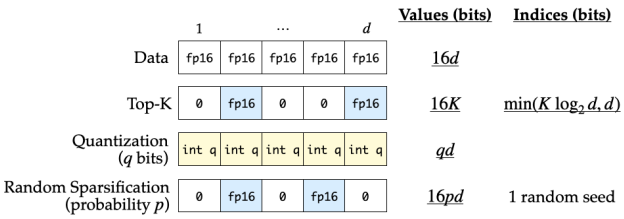


Figure 1: An illustration of three different methods to compress a d-dimensional vector using in `fp16` precision. Different methods bring complementary benefits to compression. Indices are used to store the sparsity pattern.

gradient compression methods to lower the communication overhead, including quantization (Alistarh et al., 2016; Zhang et al., 2017; Bernstein et al., 2018; Wen et al., 2017), sparsification (Wangni et al., 2018; Alistarh et al., 2018; Wang et al., 2018; 2017), sketching (Jiang et al., 2018; Ivkin et al., 2019; Rothchild et al., 2020) error compensation (Stich et al., 2018; Tang et al., 2019; Gorbunov et al., 2020; Qian et al., 2021; Lu et al., 2022; Condat et al., 2022) local training (Wang & Joshi, 2019; Lin et al., 2019; Stich, 2018; Haddadpour et al., 2019; Mishchenko et al., 2022), and asynchronous updates (Peng et al., 2017; Zheng et al., 2017; Zhou et al., 2018; Simsekli et al., 2018; Nguyen et al., 2018).

We focus on foundation model training over slow networks, centering around a key observation: *different communication-efficient training methods bring complementary benefits and disadvantages*, as illustrated in Figure 1:

1. **Top-K sparsification** improves gradient sparsity, which decrease the number of non-zero values to store, but leaves significant overhead to encode sparsity pattern;
2. **Quantization** decreases the overhead for storing each value, but cannot go below 1 bit and thus can only support limited compression ratio;
3. **Random sparsification** decreases the number of non-zero values to store and provides an efficient way to encode the sparsity pattern, but might miss important values compared with Top-K sparsification.

What would happen if we combine these methods? As we will show in this paper, combining these three methods together allows us to outperform each individual method. In fact, *only with all these techniques together, we can reach an aggressive communication compression ratio*. The key challenge in combining these methods is to balance the communication compression ratio with the convergence of the algorithm. Specifically, **(Contribution 1)** we propose CocktailSGD, a novel framework that allows us to integrate various communication compression techniques together. CocktailSGD is an asynchronous training framework that overlaps communication with the local gradient computation of the gradients. During the communication step, we use a combination of random sparsification, top-K sparsification, quantization, and careful error compensation. This allows us to *amortize* the end-to-end aggressive communication ratio to each of these dimensions.

Theoretically, **(Contribution 2)** We provably show that CocktailSGD ensures local convergence with linear speed-up, at rate $O(1/\sqrt{NT})$ on smooth non-convex objectives, where N is the number of parallel workers and T is the total number of iterations. The convergence bound holds under weaker assumptions on the compression given in previous work (Stich et al., 2018). Moreover, we also theoretically justify the effectiveness and benefits of the hybrid approach taken by CocktailSGD.

Empirically, **(Contribution 3)** we conduct large-scale experiments fine-tuning open models up to 20B parameters. We show that CocktailSGD converges to comparable training loss with a comparable number of iterations, but communicates up to 117× less data. This allows us to significantly decrease our requirements on bandwidth. Over 1Gbps network, we achieve the *same* training throughput compared with data center network. Over 500Mbps networks, we are only 1.2× slower compared with data center networks. We also conduct a careful ablation study to verify that CocktailSGD's ability to combine different techniques together is crucial to achieving such an aggressive compression ratio.

Moreover, **(Contribution 4)** we show that CocktailSGD can be used to improve open models: fine-tuned models achieve significantly higher HELM (Liang et al., 2022) core scenario scores by 7.0% for GPT-NeoX-20B.

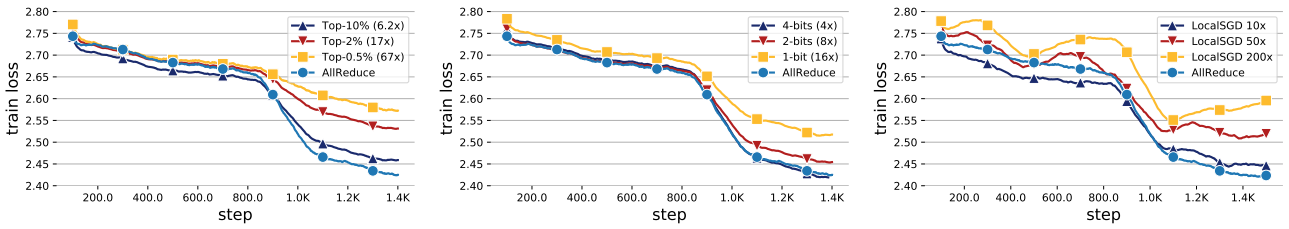
2. The CocktailSGD Framework

Problem Formulation. In this paper, we focus on the standard data-parallel setup with N parallel workers. Each worker i maintains a local data source $\mathcal{D}_i$ over which a local loss function f_i is defined. All the parallel workers collaborate to minimize the objective function $f:\mathbb{R}^d\to\mathbb{R}$, that is, find target model parameters $\hat{\boldsymbol{x}}\in\mathbb{R}^d$ such that,

$$\hat{\boldsymbol{x}}=\arg\min_{\boldsymbol{x}\in\mathbb{R}^d}\left[f(\boldsymbol{x})=\frac{1}{N}\sum_{i=1}^{N}\underbrace{\mathbb{E}_{\boldsymbol{\zeta}\sim\mathcal{D}_i}f_i(\boldsymbol{x};\boldsymbol{\zeta})}_{=f_i(\boldsymbol{x})}\right],$$

where $\boldsymbol{\zeta}$ is the data sampled from each local data source.

Figure 3(a) illustrates the computation and communication pattern of the standard data-parallel SGD (DP-SGD) method. At iteration t, each worker i holds a local model replica $\boldsymbol{x}_t^{(i)}$ and computes its local gradient over data sample $\boldsymbol{\zeta}_t^{(i)}$. All workers then communicate and compute the average



(a) Top-K. (b) Quantization. (c) Local Training.

Figure 2: Existing methods can significantly decrease the communication overhead of distributed training; however, they often converge sub-optimally under aggressive compression ratios. Top-K: (Alistarh et al., 2018) with 10%, 2%, 0.5% sparsity; Quantization: (Alistarh et al., 2016) with 4-bits, 2-bits, and 1-bit; Local Training: (Stich, 2018) with 10, 50, 200 local steps. We fine-tune OPT-1.3B model over on the WIKITEXT-103 dataset.

gradient. Worker i then updates its local model:

$$\boldsymbol{x}_{t+1}^{(i)} = \boldsymbol{x}_t^{(i)} - \gamma \sum_{i=1}^{N} \nabla f_i(\boldsymbol{x}_t^{(i)}; \boldsymbol{\zeta}_t^{(i)})$$

where γ is the learning rate. When running DP-SGD over slow networks, the communication time starts to dominate and thus slows down the training process.

2.1. Existing Methods of Communication Compression

Communication compression can speed up training over slow networks by decreasing the communication overhead, as illustrated in Figure 3(b). There are three major categories of compression techniques, sparsification (Strom, 2015; Wangni et al., 2018; Alistarh et al., 2018; Wang et al., 2018; Wangni et al., 2018; Wang et al., 2017; Lin et al., 2018), quantization (Alistarh et al., 2016; Wen et al., 2017; Bernstein et al., 2018; Wu et al., 2018; Karimireddy et al., 2019; Tang et al., 2021), and local training (Wang & Joshi, 2019; Lin et al., 2019; Stich, 2018; Haddadpour et al., 2019; Mishchenko et al., 2022), which tackle the same problem from a different perspective. Local training aims to reduce the total number of times the gradient needs to be sent at all. Method such as lo-fi (Wortsman et al., 2022) is an extreme of local training where all workers train independently and communicate once in the end. Sparsification reduces the *number of elements* that needed to be communicated in each iteration. Usually, it focuses on finding the subset of parameter updates that are more important in each iteration and only communicates partial gradients on those parameters. Quantization, on the other hand, communicates gradients for every parameter but with lower precision. It tries to optimize the necessary bits or precision for the gradient update of every parameter.

These methods can significantly decrease the communication overhead of distributed training. In Figure 2, we show one such example by fine-tuning OPT-1.3B and use three representative methods: top-K (Alistarh et al., 2018) as the representative sparsification method with the top-K percentage of 10%, 2%, and 0.5%; QSGD (Alistarh et al., 2016) as the representative quantization method and quantize the value to 4-bits, 2-bits, and 1-bit; and LocalSGD (Stich, 2018) as the representative method that introduces staleness and varies local update iterations as 10, 50, and 200. As we can see, all the algorithms can significantly lower the communication overhead — with top-K compression, we only need to communicate 10% of a total number of gradients and update the corresponding parameters; with quantization, we only need 4-bit representation of the data; and with local training we only need to communicate every 10 steps.

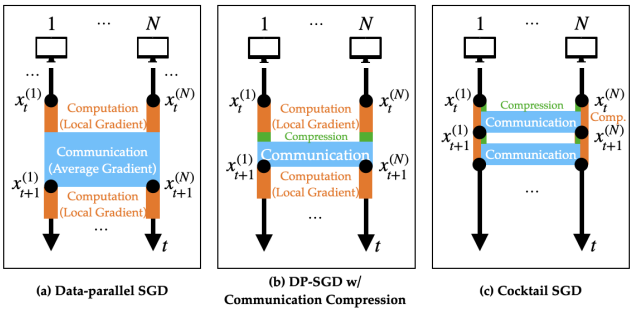


Figure 3: Communication and computation patterns for DP-SGD, DP-SGD with compression, and COCKTAILSGD.

Nevertheless, none of these methods achieve compression ratio larger than 10× without hurting the convergence. With *sparsity* of 2%, top-K gradient starts to converge slower compared with ALLREDUCE; With *precision* of 2 bits and *staleness* of 50 steps, quantization and local training methods suffer similarly. *How can we reach 100× compression?* In this paper, we combine all these methods together, to balance sparsity, staleness, and precision all in a single framework.

2.2. The COCKTAILSGD Framework

COCKTAILSGD compresses communication aggressively using a combination of top-K sparsification, quantization, and random sparsification. Such aggressive compression

Algorithm 1 CocktailSGD

1: **Initialize:** given the number of GPU nodes N, the learning rate γ, the global initialized model $\boldsymbol{x}_{t=0}$, compression function $\mathcal{C}$, global error buffer $\boldsymbol{e}_t$;
Each worker indexed by $^{(i)}$ maintains:
i) a local model copy notated by $\boldsymbol{x}_t^{(i)} \leftarrow \boldsymbol{x}_{t=0}$;
ii) a global model copy notated by $\boldsymbol{x}_t'^{(i)} \leftarrow \boldsymbol{x}_{t=0}$.
2: **for** $t = 0,\ldots,T-1$ **do**
3: // Each GPU node $^{(i)}$ runs a compute thread and a communication thread in parallel:
4: (1) Compute thread:
5: // Compute gradients on the local model:
6: Worker i: $\boldsymbol{g}_t^{(i)} \leftarrow \nabla f_i\left(\boldsymbol{x}_t^{(i)}; \boldsymbol{\zeta}_t^{(i)}\right),\ \boldsymbol{\zeta}_t^{(i)} \sim \mathcal{D}_i$
7: (2) Communication thread:
8: // Compute local model difference:
9: Worker i: $\boldsymbol{\delta}_t^{(i)} \leftarrow \left(\boldsymbol{x}_t^{(i)} - \boldsymbol{x}_t'^{(i)}\right)$
10: All workers Communicate ($\boldsymbol{\Delta}_t$ and $\boldsymbol{e}_t$ are sharded among workers or stored in parameter servers): $\boldsymbol{\Delta}_t \leftarrow \frac{1}{N}\sum_i \mathcal{C}[\boldsymbol{\delta}_t^{(i)}] + \boldsymbol{e}_t$
11: Update global error: $\boldsymbol{e}_{t+1} \leftarrow \boldsymbol{\Delta}_t - \mathcal{C}[\boldsymbol{\Delta}_t]$
12: // Worker i: wait both threads to finish
13: (3) Apply updates to the local and global models:
14: Worker i: $\boldsymbol{x}_{t+1}^{(i)} \leftarrow \boldsymbol{x}_t^{(i)} - \gamma \boldsymbol{g}_t^{(i)} + \mathcal{C}[\boldsymbol{\Delta}_t] - \mathcal{C}[\boldsymbol{\delta}_t^{(i)}]$
15: Worker i: $\boldsymbol{x}_{t+1}'^{(i)} \leftarrow \boldsymbol{x}_t'^{(i)} + \mathcal{C}[\boldsymbol{\Delta}_t]$
16: **end for**
17: **Output:** $\boldsymbol{x}_{t=T}'^{(1)} = \ldots = \boldsymbol{x}_{t=T}'^{(N)}$.

enables new system optimizations: we can often fully hide communication behind computation. As illustrated in Figure 3(c), CocktailSGD is an asynchronous framework that overlaps communication with the local gradient computation.

Algorithm 1 illustrates the CocktailSGD framework. One key design choice is the compressor $\mathcal{C}$ that we will discuss in the next section. The structure of CocktailSGD follows a natural asynchronous training paradigm with error compensation. On each worker i, we maintain two copies of the model: one copy $\boldsymbol{x}(i)_t$ is used for local forward and backward computation, and the other copy $\boldsymbol{x}_t'^{(i)}$ is used for global synchronization. This is necessary because of the concurrent nature of local computation and global synchronization. During one training iteration the worker uses *two parallel threads* — one for local computation (Line 4) and another for exchanging their last step's *model difference* between the local and global model (Line 7). Once both threads have finished, the worker updates the model parameters and proceeds to the next iteration (Line 13).

This framework introduces a 1-step staleness; *if* communication overhead is comparable to the computation, it can fully hide the communication behind computation. However, in scenarios where the network is slow, e.g. geo-distributed device connected with <1Gbps networks, the communication overhead can be significantly larger than the local computation. To overcome this issue, CocktailSGD utilizes bidirectional compression (Lines 10 and 11) to balance execution time between communication and computation. To minimize the side-effect due to compression, we absorb compression error into the local model and leverage error compensation (Gruntkowska et al., 2022; Tang et al., 2019).

Next, we focus on the core part of the paper: designing an effective compressor to enable efficient communication.

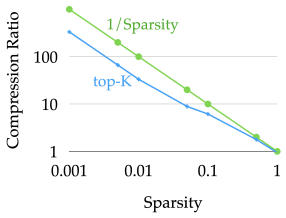
2.3. Design the Compressor

One key design in CocktailSGD is the compressor $\mathcal{C}$. Note that with a single method, it is hard to reach an aggressive end-to-end compression ratio such as 100×. We first analyze this behavior and then propose a simple, novel compressor.

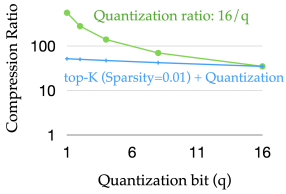
2.3.1. Challenges of Aggressive Compression

Achieving an aggressive compression ratio with a single technique is challenging. In the following, we illustrate this by assuming that the original data to be compressed consists of d numbers stored in `fp16`.

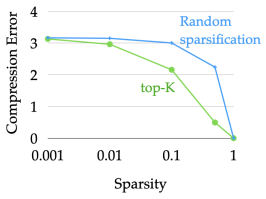
Top-K Compression One challenge of using top-K compression to achieve aggressive compression ratios is that storing the sparsity pattern introduces significant overhead. In fact, with d numbers as input, such sparsity pattern requires $K\log_2 d$ bits (as index list) or d bits (as bitmap). This often dominates what is required to store the values — if these were stored in `fp16`, we would need $16K$ bits. The right Figure illustrates this behavior — to achieve 100× compression ratio, one would need 0.3% sparsity.



Quantization Value-wise quantization alone cannot achieve more than 16× compression ratio if the original values are stored in `fp16`. Moreover, a linear decrease of # bits for quantization often lead to an exponential increase of the error. It is possible to apply quantization together with top-K compression, but this yields diminishing returns, as illustrated in the right Figure — compressing the value by 16× introduces only up to 2× additional compression because we are dominated by storing the sparsity patterns.



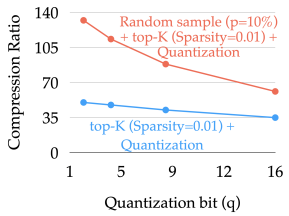
Random Sparsification Quantization sparsification eliminates the communication cost of the sparsity pattern — by sending only a random seed, the sparsity pattern can be fully recovered. However, it also



comes with its problem —
as illustrated on the right figure, given the same sparisty, ransom sparsification introduces more errors (in l_2 norm) compared with top-K sparsification, since it does not necessarily keep values of the largest norm.

2.3.2. CocktailSGD Compressor

The goal of the CocktailSGD compressor is to balance three dimensions: *sparsity*, *precision*, and *staleness* — our goal is to be mild at each dimension, but achieve significant end-to-end compression ratio altogether. Intuitively, since each of these dimensions introduces errors of different kinds, their combination does not compound significantly. Algorithm 2 illustrates the compressor that we used. Given an input vector $\boldsymbol{\delta}$ of size d to compress, we first sample a subset of parameters randomly with probability p. We then perform top-K compression over the selected pd (in expectation) parameters, followed by a q-bit quantization of the values.



Each of these three steps introduce compression in different ways. *Random sample with probability p* makes it easier to communicate sparsity patterns — if all workers agree on a random seed, these workers will agree on the selected subset of parameters without any additional communication; *Top-K* compression introduces additional sparsity over random sample and maintains important values; and *quantization* decreases the number of bits we need to store these values. Putting these together alleviates the diminishing returns. In fact, with $p=10\%$, we can reach $100\times$ compression with 1% sparsity from top-K and 4-bit quantization.

What about Convergence? The benefit provided by random sampling does not come as free — compared with top-K compression, random sampling does not necessarily choose the most important values to communicate. This can potentially hurt convergence. Empirically, we observe that the benefits in compression outweigh the degradation in convergence. In the next section, we will analyze this formally and justify this hybrid compression method.

3. Theoretical Analysis

Given the problem setup defined in Section 2, we now proceed to analyze the convergence bound of CocktailSGD. We start with a few assumptions.

Assumption 3.1. (Smoothness) For any $i\in\{0,\dots,N-1\}$, the loss function f_i is L-smooth such that there exists a constant $L>0$, for any $\boldsymbol{x},\boldsymbol{y}\in\mathbb{R}^d$, it holds that

$$\|\nabla f_i(\boldsymbol{x})-\nabla f_i(\boldsymbol{y})\|\le L\|\boldsymbol{x}-\boldsymbol{y}\|.$$

Algorithm 2 Compressor $\mathcal{C}[\boldsymbol{\delta}]$.

Input: model difference $\boldsymbol{\delta}$, sampling ratio p, top-K sparsification K, quantization q-bits.
(1) Randomly pick a subset of numbers from $(\boldsymbol{\delta})$, with probability p to include each number:
  $\boldsymbol{\delta_1}\leftarrow$ RandomSelect$(\boldsymbol{\delta},p)$
(2) Select top-K element:
  $\boldsymbol{\delta_2}\leftarrow$ top-K$(\boldsymbol{\delta_1})$
(3) Quantize values to q bits:
  $\boldsymbol{\delta_3}\leftarrow$ Quantize$(\boldsymbol{\delta_2})$
Output: $\boldsymbol{\delta_3}$.

Assumption 3.2. (Data Sampling) The stochastic gradient computed is an unbiased estimation for the full gradient with bounded variance, i.e, there exists a constant $\sigma>0$ such that for any local dataset $\mathcal{D}_i$, it holds that for any $\boldsymbol{x}\in\mathbb{R}^d$:

$$\mathbb{E}_{\boldsymbol{\zeta}\sim\mathcal{D}_i}[\nabla f_i(\boldsymbol{x};\boldsymbol{\zeta})]=\nabla f_i(\boldsymbol{x}),$$

and

$$\mathbb{E}_{\boldsymbol{\zeta}\sim\mathcal{D}_i}\|\nabla f_i(\boldsymbol{x};\boldsymbol{\zeta})-\nabla f_i(\boldsymbol{x})\|^2\le\sigma^2.$$

Assumption 3.3. (Data Heterogeneity) There exists $\varsigma>0$ such that for any $x\in\mathbb{R}^d$, it holds that

$$\frac{1}{N}\sum_{i=0}^{N-1}\|\nabla f_i(\boldsymbol{x})-f(\boldsymbol{x})\|^2\le\varsigma^2.$$

Assumption 3.1, 3.2 and 3.3 are standard assumptions in distributed training literature and are widely adopted (Koloskova et al., 2019; Tang et al., 2019; Lu & De Sa, 2020; 2021). We now introduce our assumption on the compression. Note that, CocktailSGD applies multiple approaches in the communication thread, which can be seen as an end-to-end compression procedure $\mathcal{C}:\mathbb{R}^d\to\mathbb{R}^d$ that takes $\boldsymbol{x}_t^{(i)}-\boldsymbol{x}_t$ as input and outputs $\mathcal{C}[\boldsymbol{x}_t^{(i)}-\boldsymbol{x}_t]$, for any $t\ge0$. More specifically, denote $\mathcal{C}_R,\mathcal{C}_K,\mathcal{C}_Q$ to be RandomSelect, top-K, and Quantize in CocktailSGD, respectively, then $\mathcal{C}=\mathcal{C}_Q\circ\mathcal{C}_K\circ\mathcal{C}_R$ can be seen as the composition of three individual compression procedures. Without the loss of generality, we first make an end-to-end assumption on $\mathcal{C}$:

Assumption 3.4. (Compression Error) The end-to-end compression procedure $\mathcal{C}$ in Algorithm 1 has bounded error such that for any $\boldsymbol{x}\in\mathbb{R}^d$, there exists a constant $0\le\omega<1$,

$$\mathbb{E}\|\mathcal{C}(\boldsymbol{x})-\boldsymbol{x}\|^2\le\omega^2\|\boldsymbol{x}\|^2.$$

We now give the convergence bound of CocktailSGD in the following theorem,

Theorem 3.5. *In Algorithm 1, under Assumption 3.1, 3.2, 3.3 and 3.4, if we set the learning rate γ to be*

$$\gamma=\left(\sigma\sqrt{\frac{T}{N}}+\frac{(\sigma^2+3\varsigma^2)^{\frac{1}{3}}L^{\frac{2}{3}}T^{\frac{1}{3}}}{F^{\frac{1}{3}}(1-\omega)^{\frac{4}{3}}}+\frac{35L}{(1-\omega)^2}\right)^{-1}$$

, Algorithm 1 converges at the following rate,

$$\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\|\nabla f(\boldsymbol{x}_t)\|^2\le O\left(\frac{1}{\sqrt{NT}}+\frac{1}{(1-\omega)^{\frac{4}{3}}T^{\frac{2}{3}}}+\frac{1}{(1-\omega)^2T^2}\right),$$

where $F=f(\boldsymbol{x}_0)-\inf_{\boldsymbol{x}}f(\boldsymbol{x})$.

Remark I: Linear Speed Up. Observing the bound $O(1/\sqrt{NT} + 1/(1-\omega)^{4/3}T^{2/3})$ given in Theorem 3.5, CocktailSGD guarantees linear speed up over number of workers N and number of iterations T, at rate $O(1/\sqrt{NT})$. Notice that the compression-related variable ω only appears in the non-leading terms. Intuitively, this allows aggressive compression without significantly hurting the convergence. The bound is tight in terms of N and T compared to the lower bound in the same setting (Lu & De Sa, 2021).

Remark II: Comparison to Previous Bounds. Comparing Theorem 3.5 and the bound given in Tang et al. (2019), CocktailSGD converges under weaker assumption on the compression scheme. Specifically, as pointed out by Tang et al. (2019), previous work like (Stich et al., 2018) obtains the rate $O(1/\sqrt{NT})$ under both Assumption 3.4 and bounded gradients. In contrast, Theorem 3.5 ensures the same convergence rate without the bounded gradients assumption.

Remark III: A Closer Look at ω. We now discuss how the variable ω connects to different compression components in CocktailSGD. Note that each compression approach, including RandomSelect, top-K, and Quantize, fulfills Assumption 3.4 with different ω. We now give the error bound where all the compression components are composed.

Lemma 3.6. *Denote $\mathcal{C}_R, \mathcal{C}_K, \mathcal{C}_Q$ to be* RandomSelect, top-K, *and* Quantize *in* CocktailSGD, *respectively. Then it holds that the end-to-end compression in* CocktailSGD*: $\mathcal{C} = \mathcal{C}_Q \circ \mathcal{C}_K \circ \mathcal{C}_R$ fulfills the Assumption 3.4 with*

$$\omega^2 = 1 - \frac{\min\{K, pd\}}{d + d \cdot \min\{d/s^2, \sqrt{d}/s\}},$$

where s denotes the precision level given in the Quantize *function (Alistarh et al., 2016).*

Given Lemma 3.6, we can obtain a finer-grained convergence bound of CocktailSGD as given in the following corollary.

Corollary 3.7. *In Algorithm 1, under Assumption 3.1, 3.2, 3.3 and 3.4, if we use Algorithm 2 as $\mathcal{C}$, set the learning rate γ to be the same as in Theorem 3.5, then Algorithm 1 converges at the following rate,*

$$\frac{1}{T}\sum_{t=0}^{T-1} \mathbb{E}\|\nabla f(\boldsymbol{x}_t)\|^2 \leq O\left(\frac{1}{\sqrt{NT}} + \frac{d^{\frac{4}{3}} + d^{\frac{4}{3}} \min\{d/s^2, \sqrt{d}/s\}^{\frac{4}{3}}}{\min\{K, pd\}^{\frac{2}{3}} T^{\frac{2}{3}}}\right).$$

Note that since Assumption 3.4 explicitly constraints that $\omega < 1$, so that Corollary 3.7 implicitly requires $\min\{K, pd\} > 0$, which holds in our case. Observing the bound given in Corollary 3.7, the compression-related term is not dominant in terms of T. However, when compression becomes aggressive (this could either mean $K \to 0$, $p \to 0$ or $s \to 0$), then the second term will gradually become the leading term in the convergence bound. On the other hand, if p is not too small (specifically $p \geq K/d$), then the RandomSelect-p component will have no impact on the final convergence rate as $\min\{K, pd\}$ will be dominated by K. *This justifies our hybrid compressor* — the convergence is relatively robust to mild random sparsification, but on the other hand, random sparsification improves the end-to-end compression ratio as we showed in Section 2.3.2.

4. Experiments

We demonstrate that CocktailSGD can significantly speed up the training of large language models in slow networks. Specifically, we show that:

(1) CocktailSGD can tolerate up to 117× aggressive compression without hurting the converging rate and final loss, whereas several strong baselines including top-K (Alistarh et al., 2018) and ProxSkip (Mishchenko et al., 2022), converge slower with comparable compression ratio.

(2) Under slow networks (500Mbps), CocktailSGD provides a 20-50× end-to-end speedup comparing with AllReduce based data parallel implementation. CocktailSGD under 500Mbps network is only 1.2× slower compared with AllReduce under data center network.

(3) With CocktailSGD, we further fine-tune GPT-NeoX-20B using instruction-tuning data, improving the average HELM core scenario score by 7%.

4.1. Experimental Setup

Datasets and Models. We consider *instruction tuning* on standard *language modeling* datasets. For instruction tuning, we use a collection of Natural-Instruction (NI) (Mishra et al., 2022; Wang et al., 2022), Public Pool of Prompts (P3) (Bach et al., 2022), Chain-of-Thought (Wei et al., 2022) data, and The Pile (Gao et al., 2020) to prevent catastrophic forgetting previously learned knowledge. We refer to this data collection as InstructionData. For language modeling, we train on WikiText-103 data (Merity et al., 2016). We fine-tune OPT-1.3B, GPT-J-6B, and GPT-NeoX-20B on these datasets.

Infrastructure. We use A100-80GB GPUs to train large language models. To simulate slower network conditions, we utilize Linux traffic control to control the communication bandwidth between instances. To effectively utilize the resources of each server, we employ pipeline parallelism and utilize the fast intra-machine connection to communicate activations and their corresponding gradients. For the models OPT-1.3B and GPT-J-6B, we utilize 4 data parallel workers with 2 A100 GPUs each. For the GPT-NeoX-20B model, we use 4 data parallel workers with 8 A100 GPUs each, for a total of 32 A100 GPUs.

Baselines. We compare with three strong baselines:

AllReduce. We use an efficient AllReduce based data

parallel implementation as the baseline for statistical efficiency, where all communications are in `fp16`.

PROXSKIP. We choose PROXSKIP (Mishchenko et al., 2022) as the state-of-the-art baseline for communication delay methods with staleness.

TOP-K. We select top-K sparsification (Alistarh et al., 2018) on the gradient with two-pass error-compensated compression (Tang et al., 2019) as the strong baseline for methods optimized for communication compression.

We run all methods with a targeted compression ratio of 100×. We use a compression ratio of 100 for PROXSKIP. For TOP-K, we set the sparsity ratio to 0.5%. COCKTAILSGD uses 10% random sampling p and 20% top-K, and perform quantization with 4 bits. We also apply lossless compression with zlib[1] on all methods, achieving an average end-to-end compression ratios of up to 109× for PROXSKIP, 110× for TOP-K, and 117× for COCKTAILSGD.

Hyperparameter Tuning. We train in mixed precision (FP16) training and conduct careful tuning for all methods on all datasets. We use the default Adam optimizer. The optimal learning rate is determined through a grid search, ranging from 1e-6 to 1e-3, to ensure optimal convergence for each method. We use a batch size of 64, 128, 128 for OPT-1.3B, GPT-J-6B, GPT-NeoX-20B, respectively, and a sequence length of 2048.

4.2. Results

Convergence. We first compare the convergence behavior of different methods in Figure 4. Figure 4 shows the convergence curves on instruct-tuning and language modeling data. ALLREDUCE converges fastest in terms of iteration as it exchanges the full gradients in `fp16` without compression and relaxation. TOP-K and PROXSKIP under aggressive compression significantly compromises the convergence due to aggressive compression or staleness from the same technique, which matches our finding illustrated in Section 2. On the other hand, COCKTAILSGD converges almost as fast as ALLREDUCE in terms of the number of training iterations, despite requiring up to 117× communication.

End-to-End Runtime. Figure 5 shows the throughput of ALLREDUCE and COCKTAILSGD under different network configurations. Benefiting from the asynchronous communication mode, COCKTAILSGD can hide the communication in the local computation, thus maintaining nearly constant training throughput if the communication time is less than the computation time. Under a slower 500Mbps network, COCKTAILSGD introduces 47.2× improvement of the training throughput for GPT-NeoX-20B, when compared with ALLREDUCE. Under 1Gbps network, COCKTAILSGD also achieves substantial improvements, with 30.2× improvement for the respective models. In fact, this throughput is almost the same as the throughput of ALLREDUCE with "infinite bandwidth" (when all communications are disabled), with only 7% slower. And the throughput of 500Mbps is only 1.2× slower than data center networks. This unlocks the use of geographically distributed machines connected with slow networks to effectively train large-scale language models while maintaining a reasonable hardware efficiency and convergence speed.

Improving Foundation Models. We are also excited to share that COCKTAILSGD managed to accomplish a set of real-world foundation model training tasks. Specifically, we used the fine-tuned GPT-J-6B and GPT-NeoX-20B models, which were trained on the INSTRUCTIONDATA dataset, and evaluated them using the HELM protocol. Our results, as depicted in Figure 6, showed an improvement of 7.4% and 7.0% for GPT-J-6B and GPT-NeoX-20B respectively on the core scenario tasks, spanning a diverse range of task types.

4.3. Ablation Study

In this subsection, we conduct ablation study to understand the importance of top-K, random sampling, and quantization.

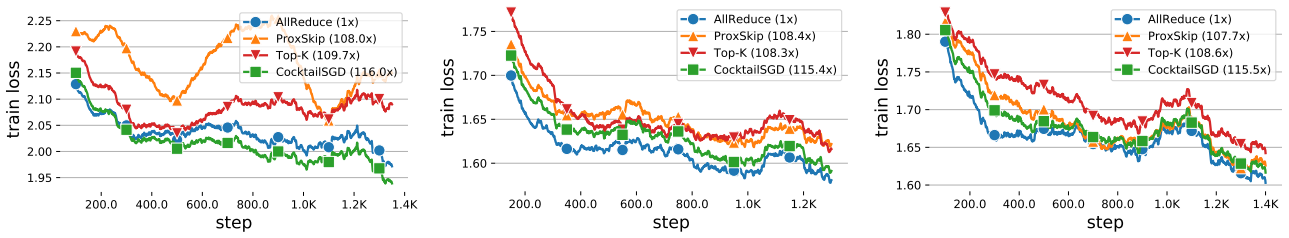
We found that using top-K alone for aggressive sparsification can lead to poor convergence, as shown in Figure 7a (Top0.5%). When combined with quantization, it compresses the values from `fp16` to 4-bits. However, the sparsity patterns then dominate the communication overhead. In fact, for large-scale models, e.g. >1 billion parameters, the indices are larger than the quantized top-K values themselves. To tackle this issue, we perform random sampling first and use bitmap to represent the position of top-K values. Figure 7a shows that performing random sampling before top-K does not hurt the convergence under the same sparsity. Meanwhile, as shown in Figure 7b, this greatly reduces the communication volume under the same sparsity.

Furthermore, we also try removing top-K or quantization respectively. However, as presented in Figure 7a, the removal of these methods resulted in slower convergence, which highlights the importance of utilizing various types of compression algorithms in order to maintain a fast convergence rate.

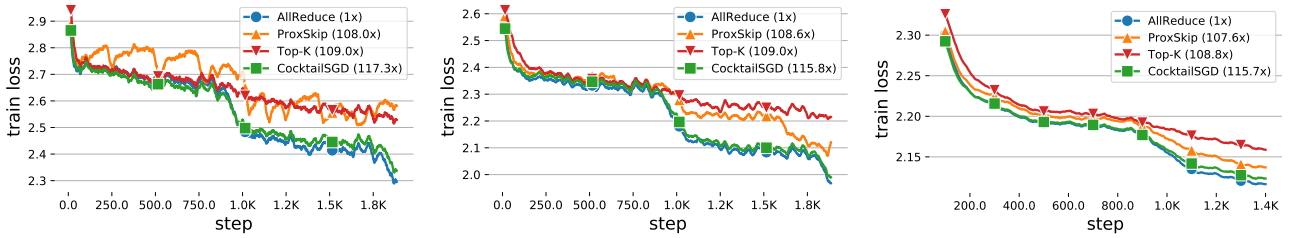
5. Related Work

Communication-efficient learning algorithms have been studied for decades since model/gradient synchronization can be the bottleneck of distributed data-parallel training (Xu et al., 2020). *Quantization* and *sparsification* are two popular compression methods for communication volume reduction methods. On the other hand, there are some communication delay methods such as *Local SGD* and *asynchronous SGD* that introduce some staleness of gradient updates in return for

[1] https://www.zlib.net/



(a) INSTRUCTIONDATA, OPT-1.3B (b) INSTRUCTIONDATA, GPT-J-6B (c) INSTRUCTIONDATA, GPT-NeoX-20B



(d) WIKITEXT-103, OPT-1.3B (e) WIKITEXT-103, GPT-J-6B (f) WIKITEXT-103, GPT-NeoX-20B

Figure 4: Convergence of OPT-1.3B, GPT-J-6B and GPT-NeoX-20B on INSTRUCTIONDATA and WIKITEXT-103. COCKTAILSGD achieves comparble convergence as ALLREDUCE despite using up to 117× less communication.

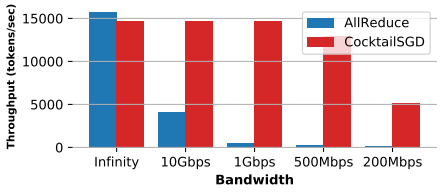


Figure 5: Throughput (GPT-NeoX-20B) under different network configurations. OPT-1.3B and GPT-J-6B incur smaller slowdown (shown in Appendix). 'Infinity' is when all communications are disabled, which is a throughput upper bound.

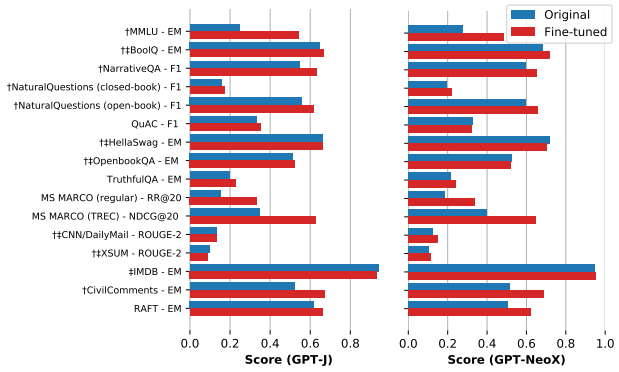
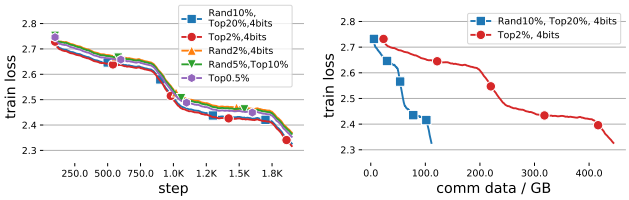


Figure 6: Improving open models on HELM core scenarios. † indicates NI data overlaps with the corresponding dataset; ‡ indicates overlaps with P3.



(a) Loss vs. training steps. (b) Loss vs communication data.

Figure 7: Ablation study of compression methods.

a reduction of communication frequency or synchronization.

Quantization based methods (Alistarh et al., 2016; Wen et al., 2017; Bernstein et al., 2018; Wu et al., 2018; Karimireddy et al., 2019; Tang et al., 2021) first quantize the gradient/model into a lower precision representation before communication instead of directly communicating the `fp32` numbers. The quantized gradient can be a biased (Bernstein et al., 2018; Wu et al., 2018; Karimireddy et al., 2019) or unbiased estimation (Alistarh et al., 2016; Wen et al., 2017) of the original value. To further improve the compression ratio and overcome the 1-bit-per-value limit of conventional quantization methods, recent studies have proposed leveraging the data structure in the quantization strategy (Stock et al., 2019; Fan et al., 2020). **Sparsification** based methods (Strom, 2015; Wangni et al., 2018; Alistarh et al., 2018; Wang et al., 2018; Wangni et al., 2018; Wang et al., 2017; Lin et al., 2018) can support a more aggressive compression ratio than quantization—quantization can only compress the message up to 32×, where at least one bit is required to repre-

sent a value. The gradients of neural networks can be very sparse, which provides great opportunities for sparsification-based compression. Similarly, the sparsify function can be biased (Strom, 2015; Lin et al., 2018) or unbiased (Wangni et al., 2018) as well. **Error Compensation** (Seide et al., 2014; Gruntkowska et al., 2022; Condat et al., 2022) compensates for compression errors and further enables more aggressive compression ratios. Xie et al. (2020) proposes *error reset* producing bifurcated local models with periodic reset of resulting local residual errors.

Local Training based methods (Wang & Joshi, 2019; Lin et al., 2019; Stich, 2018; Haddadpour et al., 2019; Mishchenko et al., 2022) are introduced to optimize for the number of communication rounds during training (Wang & Joshi, 2019; Lin et al., 2019; Stich, 2018; Haddadpour et al., 2019). **Asynchronous update** based methods (Peng et al., 2017; Zheng et al., 2017; Zhou et al., 2018; Simsekli et al., 2018; Nguyen et al., 2018; Nadiradze et al., 2021) try to remove the synchronization barrier, which is an obstacle in communication for clusters with stragglers.

Combination of Multiple Methods. Since quantization and sparsification are two orthogonal methods, there is also some active research discussing how to effectively combine them (Basu et al., 2019). For example, Jiang & Agrawal (2018) leverages quantization and local computation, and Basu et al. (2019) combines aggressive sparsification with quantization and local computation along with error compensation. However, they can only compress one direction and thus cannot achieve aggressive compression. Our paper is inspired by these efforts but significantly improve their compression ratio and applies to training large-scale LLMs.

6. Conclusion

In this paper, we examine the limitations of using a single category of compression methods for distributed training and propose a new framework called CocktailSGD. It combines three distinct compression techniques, namely random sparsification, top-K sparsificaiton, and quantizaiton, to achieve better compression ratios than each technique alone. Our theoretical analysis suggests that CocktailSGD ensures local convergence with linear speed up at $O(1/\sqrt{NT})$ on smooth non-convex objectives. Empirically, we show that CocktailSGD can achieve up to 117× compression without hurting the convergence.

Acknowledgments

CZ and the DS3Lab gratefully acknowledge the support from the Swiss State Secretariat for Education, Research and Innovation (SERI) under contract number MB22.00036 (for European Research Council (ERC) Starting Grant TRIDENT 101042665), the Swiss National Science Foundation (Project Number 200021_184628, and 197485), Innosuisse/SNF BRIDGE Discovery (Project Number 40B2-0_187132), European Union Horizon 2020 Research and Innovation Programme (DAPHNE, 957407), Botnar Research Centre for Child Health, Swiss Data Science Center, Alibaba, Cisco, eBay, Google Focused Research Awards, Kuaishou Inc., Oracle Labs, Zurich Insurance, and the Department of Computer Science at ETH Zurich. CR gratefully acknowledges the support of NIH under No. U54EB020405 (Mobilize), NSF under Nos. CCF1763315 (Beyond Sparsity), CCF1563078 (Volume to Velocity), and 1937301 (RTML); ARL under No. W911NF-21-2-0251 (Interactive Human-AI Teaming); ONR under No. N000141712266 (Unifying Weak Supervision); ONR N00014-20-1-2480: Understanding and Applying Non-Euclidean Geometry in Machine Learning; N000142012275 (NEPTUNE); NXP, Xilinx, LETI-CEA, Intel, IBM, Microsoft, NEC, Toshiba, TSMC, ARM, Hitachi, BASF, Accenture, Ericsson, Qualcomm, Analog Devices, Google Cloud, Salesforce, Total, the HAI-GCP Cloud Credits for Research program, the Stanford Data Science Initiative (SDSI), and members of the Stanford DAWN project: Facebook, Google, and VMWare. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views, policies, or endorsements, either expressed or implied, of NIH, ONR, or the U.S. CDS acknowledges the support from NSF-2046760 CAREER. YL acknowledges the support of the Meta PhD Fellowship. The computation required in this work was provided by Together Computer (`https://together.xyz/`).

References

Alistarh, D., Grubic, D., Li, J., Tomioka, R., and Vojnovic, M. Qsgd: Communication-efficient sgd via gradient quantization and encoding. *arXiv preprint arXiv:1610.02132*, 2016.

Alistarh, D., Hoefler, T., Johansson, M., Khirirat, S., Konstantinov, N., and Renggli, C. The convergence of sparsified gradient methods. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pp. 5977–5987, 2018.

Austin, J., Odena, A., Nye, M., Bosma, M., Michalewski, H., Dohan, D., Jiang, E., Cai, C. J., Terry, M., Le, Q. V., et al. Program synthesis with large language models. 2021.

Bach, S. H., Sanh, V., Yong, Z.-X., Webson, A., Raffel, C., Nayak, N. V., Sharma, A., Kim, T., Bari, M. S., Fevry, T., Alyafeai, Z., Dey, M., Santilli, A., Sun, Z., Ben-David, S., Xu, C., Chhablani, G., Wang, H., Fries, J. A., Al-shaibani, M. S., Sharma, S., Thakker, U., Almubarak, K., Tang, X.,

Tang, X., Jiang, M. T.-J., and Rush, A. M. Promptsource: An integrated development environment and repository for natural language prompts, 2022.

Basu, D., Data, D., Karakus, C., and Diggavi, S. Qsparse-local-sgd: Distributed sgd with quantization, sparsification and local computations. *Advances in Neural Information Processing Systems*, 32, 2019.

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

Bommasani, R., Hudson, D. A., Adeli, E., Altman, R., Arora, S., von Arx, S., Bernstein, M. S., Bohg, J., Bosselut, A., Brunskill, E., et al. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*, 2021.

Borzunov, A., Baranchuk, D., Dettmers, T., Ryabinin, M., Belkada, Y., Chumachenko, A., Samygin, P., and Raffel, C. Petals: Collaborative inference and fine-tuning of large models. *arXiv preprint arXiv:2209.01188*, 2022a.

Borzunov, A., Ryabinin, M., Dettmers, T., Lhoest, Q., Saulnier, L., Diskin, M., and Jernite, Y. Training transformers together. In *NeurIPS 2021 Competitions and Demonstrations Track*, pp. 335–342. PMLR, 2022b.

Brants, T., Popat, A. C., Xu, P., Och, F. J., and Dean, J. Large language models in machine translation. 2007.

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 2020.

Chowdhery, A., Narang, S., Devlin, J., Bosma, M., Mishra, G., Roberts, A., Barham, P., Chung, H. W., Sutton, C., Gehrmann, S., et al. Palm: Scaling language modeling with pathways. *arXiv preprint arXiv:2204.02311*, 2022.

Condat, L., Agarsky, I., and Richtárik, P. Provably doubly accelerated federated learning: The first theoretically successful combination of local training and compressed communication. *arXiv preprint arXiv:2210.13277*, 2022.

Diskin, M., Bukhtiyarov, A., Ryabinin, M., Saulnier, L., Sinitsin, A., Popov, D., Pyrkin, D. V., Kashirin, M., Borzunov, A., Villanova del Moral, A., et al. Distributed deep learning in open collaborations. *Advances in Neural Information Processing Systems*, 34:7879–7897, 2021.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*.

Fan, A., Stock, P., Graham, B., Grave, E., Gribonval, R., Jegou, H., and Joulin, A. Training with quantization noise for extreme model compression. *arXiv preprint arXiv:2004.07320*, 2020.

Gao, L., Biderman, S., Black, S., Golding, L., Hoppe, T., Foster, C., Phang, J., He, H., Thite, A., Nabeshima, N., et al. The pile: An 800gb dataset of diverse text for language modeling. *arXiv preprint arXiv:2101.00027*, 2020.

Gorbunov, E., Kovalev, D., Makarenko, D., and Richtárik, P. Linearly converging error compensated sgd. *Advances in Neural Information Processing Systems*, 33:20889–20900, 2020.

Gruntkowska, K., Tyurin, A., and Richtárik, P. Ef21-p and friends: Improved theoretical communication complexity for distributed optimization with bidirectional compression. *arXiv preprint arXiv:2209.15218*, 2022.

Haddadpour, F., Kamani, M. M., Mahdavi, M., and Cadambe, V. R. Local sgd with periodic averaging: Tighter analysis and adaptive synchronization. *arXiv preprint arXiv:1910.13598*, 2019.

Ivkin, N., Rothchild, D., Ullah, E., Stoica, I., Arora, R., et al. Communication-efficient distributed sgd with sketching. In *Advances in Neural Information Processing Systems*, pp. 13144–13154, 2019.

Jiang, J., Fu, F., Yang, T., and Cui, B. Sketchml: Accelerating distributed machine learning with data sketches. In *Proceedings of the 2018 ACM SIGMOD International Conference on Management of Data*, pp. 1269–1284, 2018.

Jiang, P. and Agrawal, G. A linear speedup analysis of distributed deep learning with sparse and quantized communication. *Advances in Neural Information Processing Systems*, 31, 2018.

Jue, W., Yuan, B., Rimanic, L., He, Y., Dao, T., Chen, B., Re, C., and Zhang, C. Fine-tuning language models over slow networks using activation quantization with guarantees. In *Advances in Neural Information Processing Systems*.

Kaplan, J., McCandlish, S., Henighan, T., Brown, T. B., Chess, B., Child, R., Gray, S., Radford, A., Wu, J., and Amodei, D. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

Karimireddy, S. P., Rebjock, Q., Stich, S., and Jaggi, M. Error feedback fixes signsgd and other gradient compression schemes. In *International Conference on Machine Learning*, pp. 3252–3261. PMLR, 2019.

Koloskova, A., Stich, S., and Jaggi, M. Decentralized stochastic optimization and gossip algorithms with compressed communication. In *International Conference on Machine Learning*, pp. 3478–3487. PMLR, 2019.

Larson, S. M., Snow, C. D., Shirts, M., and Pande, V. S. Folding@ home and genome@ home: Using distributed computing to tackle previously intractable problems in computational biology. *arXiv preprint arXiv:0901.0866*, 2009.

Lian, X., Zhang, C., Zhang, H., Hsieh, C.-J., Zhang, W., and Liu, J. Can decentralized algorithms outperform centralized algorithms? a case study for decentralized parallel stochastic gradient descent. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pp. 5336–5346, 2017.

Liang, P., Bommasani, R., Lee, T., Tsipras, D., Soylu, D., Yasunaga, M., Zhang, Y., Narayanan, D., Wu, Y., Kumar, A., et al. Holistic evaluation of language models. *arXiv preprint arXiv:2211.09110*, 2022.

Lin, T., Stich, S. U., Patel, K. K., and Jaggi, M. Don't use large mini-batches, use local sgd. In *International Conference on Learning Representations*, 2019.

Lin, Y., Han, S., Mao, H., Wang, Y., and Dally, B. Deep gradient compression: Reducing the communication bandwidth for distributed training. In *International Conference on Learning Representations*, 2018.

Lu, Y. and De Sa, C. Moniqua: Modulo quantized communication in decentralized sgd. In *International Conference on Machine Learning*, pp. 6415–6425. PMLR, 2020.

Lu, Y. and De Sa, C. Optimal complexity in decentralized training. In *International Conference on Machine Learning*, pp. 7111–7123. PMLR, 2021.

Lu, Y., Li, C., Zhang, M., De Sa, C., and He, Y. Maximizing communication efficiency for large-scale training via 0/1 adam. *arXiv preprint arXiv:2202.06009*, 2022.

Merity, S., Xiong, C., Bradbury, J., and Socher, R. Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843*, 2016.

Microsoft. Microsoft announces new supercomputer, lays out vision for future ai work. https://news.microsoft.com/source/features/ai/openai-azure-supercomputer/, 2020. May 19, 2020.

Mishchenko, K., Malinovsky, G., Stich, S., and Richtárik, P. Proxskip: Yes! local gradient steps provably lead to communication acceleration! finally! *arXiv preprint arXiv:2202.09357*, 2022.

Mishra, S., Khashabi, D., Baral, C., and Hajishirzi, H. Cross-task generalization via natural language crowdsourcing instructions. In *ACL*, 2022.

Nadiradze, G., Sabour, A., Davies, P., Li, S., and Alistarh, D. Asynchronous decentralized sgd with quantized and local updates. *Advances in Neural Information Processing Systems*, 34:6829–6842, 2021.

Nguyen, L., Nguyen, P. H., Dijk, M., Richtárik, P., Scheinberg, K., and Takác, M. Sgd and hogwild! convergence without the bounded gradients assumption. In *International Conference on Machine Learning*, pp. 3750–3758. PMLR, 2018.

Peng, H., Zhe, S., Zhang, X., and Qi, Y. Asynchronous distributed variational gaussian process for regression. In *International Conference on Machine Learning*, pp. 2788–2797. PMLR, 2017.

Qian, X., Richtárik, P., and Zhang, T. Error compensated distributed sgd can be accelerated. *Advances in Neural Information Processing Systems*, 34:30401–30413, 2021.

Rothchild, D., Panda, A., Ullah, E., Ivkin, N., Stoica, I., Braverman, V., Gonzalez, J., and Arora, R. Fetchsgd: Communication-efficient federated learning with sketching. In *International Conference on Machine Learning*, pp. 8253–8265. PMLR, 2020.

Ryabinin, M. and Gusev, A. Towards crowdsourced training of large neural networks using decentralized mixture-of-experts. *Advances in Neural Information Processing Systems*, 33:3659–3672, 2020.

Ryabinin, M., Gorbunov, E., Plokhotnyuk, V., and Pekhimenko, G. Moshpit sgd: Communication-efficient decentralized training on heterogeneous unreliable devices. *Advances in Neural Information Processing Systems*, 34: 18195–18211, 2021.

Scao, T. L., Fan, A., Akiki, C., Pavlick, E., Ilić, S., Hesslow, D., Castagné, R., Luccioni, A. S., Yvon, F., Gallé, M., et al. Bloom: A 176b-parameter open-access multilingual language model. *arXiv preprint arXiv:2211.05100*, 2022.

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

Simsekli, U., Yildiz, C., Nguyen, T. H., Cemgil, T., and Richard, G. Asynchronous stochastic quasi-newton mcmc for non-convex optimization. In *International Conference on Machine Learning*, pp. 4674–4683. PMLR, 2018.

Stich, S. U. Local sgd converges fast and communicates little. In *International Conference on Learning Representations*, 2018.

Stich, S. U., Cordonnier, J.-B., and Jaggi, M. Sparsified sgd with memory. *Advances in Neural Information Processing Systems*, 31:4447–4458, 2018.

Stock, P., Joulin, A., Gribonval, R., Graham, B., and Jégou, H. And the bit goes down: Revisiting the quantization of neural networks. *arXiv preprint arXiv:1907.05686*, 2019.

Strom, N. Scalable distributed dnn training using commodity gpu cloud computing. In *Sixteenth annual conference of the international speech communication association*, 2015.

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

Tang, H., Gan, S., Awan, A. A., Rajbhandari, S., Li, C., Lian, X., Liu, J., Zhang, C., and He, Y. 1-bit adam: Communication efficient large-scale training with adam's convergence speed. *arXiv preprint arXiv:2102.02888*, 2021.

Vogels, T., Karimireddy, S. P., and Jaggi, M. Powersgd: Practical low-rank gradient compression for distributed optimization. *Advances in Neural Information Processing Systems*, 32, 2019.

Wang, H., Sievert, S., Charles, Z., Liu, S., Wright, S., and Papailiopoulos, D. Atomo: communication-efficient learning via atomic sparsification. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pp. 9872–9883, 2018.

Wang, J. and Joshi, G. Adaptive communication strategies to achieve the best error-runtime trade-off in local-update sgd. In *Systems and Machine Learning (SysML) Conference*, 2019.

Wang, J., Kolar, M., Srebro, N., and Zhang, T. Efficient distributed learning with sparsity. In *International Conference on Machine Learning*, pp. 3636–3645. PMLR, 2017.

Wang, Y., Mishra, S., Alipoormolabashi, P., Kordi, Y., Mirzaei, A., Arunkumar, A., Ashok, A., Dhanasekaran, A. S., Naik, A., Stap, D., et al. Super-naturalinstructions:generalization via declarative instructions on 1600+ tasks. In *EMNLP*, 2022.

Wangni, J., Liu, J., Wang, J., and Zhang, T. Gradient sparsification for communication-efficient distributed optimization. *Advances in Neural Information Processing Systems*, 31:1299, 2018.

Wei, J., Wang, X., Schuurmans, D., Bosma, M., Chi, E., Le, Q., and Zhou, D. Chain of thought prompting elicits reasoning in large language models. *arXiv preprint arXiv:2201.11903*, 2022.

Wen, W., Xu, C., Yan, F., Wu, C., Wang, Y., Chen, Y., and Li, H. Terngrad: ternary gradients to reduce communication in distributed deep learning. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pp. 1508–1518, 2017.

Wortsman, M., Gururangan, S., Li, S., Farhadi, A., Schmidt, L., Rabbat, M., and Morcos, A. S. lo-fi: distributed fine-tuning without communication. *arXiv preprint arXiv:2210.11948*, 2022.

Wu, J., Huang, W., Huang, J., and Zhang, T. Error compensated quantized sgd and its applications to large-scale distributed optimization. In *International Conference on Machine Learning*, pp. 5325–5333. PMLR, 2018.

Xie, C., Zheng, S., Koyejo, S., Gupta, I., Li, M., and Lin, H. Cser: Communication-efficient sgd with error reset. *Advances in Neural Information Processing Systems*, 33: 12593–12603, 2020.

Xu, H., Ho, C.-Y., Abdelmoniem, A. M., Dutta, A., Bergou, E. H., Karatsenidis, K., Canini, M., and Kalnis, P. Compressed communication for distributed deep learning: Survey and quantitative evaluation. Technical report, 2020.

Yuan, B., He, Y., Davis, J. Q., Zhang, T., Dao, T., Chen, B., Liang, P., Re, C., and Zhang, C. Decentralized training of foundation models in heterogeneous environments. *arXiv preprint arXiv:2206.01288*, 2022.

Zhang, H., Li, J., Kara, K., Alistarh, D., Liu, J., and Zhang, C. Zipml: Training linear models with end-to-end low precision, and a little bit of deep learning. In *International Conference on Machine Learning*, pp. 4035–4043. PMLR, 2017.

Zhang, S., Roller, S., Goyal, N., Artetxe, M., Chen, M., Chen, S., Dewan, C., Diab, M., Li, X., Lin, X. V., et al. Opt: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068*, 2022.

Zheng, S., Meng, Q., Wang, T., Chen, W., Yu, N., Ma, Z.-M., and Liu, T.-Y. Asynchronous stochastic gradient descent with delay compensation. In *International Conference on Machine Learning*, pp. 4120–4129. PMLR, 2017.

Zhou, Z., Mertikopoulos, P., Bambos, N., Glynn, P., Ye, Y., Li, L.-J., and Fei-Fei, L. Distributed asynchronous optimization with unbounded delays: How slow can you go? In *International Conference on Machine Learning*, pp. 5970–5979. PMLR, 2018.

A. Theoretical Analysis

A.1. Proof to Theorem 3.5

Proof. For simplicity, we start from defining the following variables (we denote $\hat{\boldsymbol{\delta}}_t^{(i)}$ to be the output of Algorithm 2):

$$\begin{aligned}
\boldsymbol{\delta}_t^{(i)} &= \boldsymbol{x}_t^{(i)} - \boldsymbol{x}_t \\
\bar{\boldsymbol{x}}_t &= \frac{1}{N}\sum_{i=0}^{N-1} \boldsymbol{x}_t^{(i)} \\
\bar{\boldsymbol{g}}_t &= \frac{1}{N}\sum_{i=0}^{N-1} \boldsymbol{g}_t^{(i)} \\
\bar{\boldsymbol{\delta}}_t &= \frac{1}{N}\sum_{i=0}^{N-1} \boldsymbol{\delta}_t^{(i)} \\
\bar{\hat{\boldsymbol{\delta}}}_t &= \frac{1}{N}\sum_{i=0}^{N-1} \hat{\boldsymbol{\delta}}_t^{(i)}.
\end{aligned}$$

Additionally, we denote

$$\hat{\boldsymbol{g}}_t = -\frac{\gamma}{N}\sum_{i=0}^{N-1} f_i\left(\boldsymbol{x}_t^{(i)}\right).$$

Now we take average on all the local model update formula (i.e., taking average over $i=0$ to $N-1$ on the update of $\boldsymbol{x}_t^{(i)}$), it'll give us

$$\bar{\boldsymbol{x}}_{t+1} = \bar{\boldsymbol{x}}_t + \bar{\boldsymbol{g}}_t + \boldsymbol{\Delta}_t - \bar{\hat{\boldsymbol{\delta}}}_t.$$

Note that

$$\begin{aligned}
&\boldsymbol{\Delta}_t - \bar{\hat{\boldsymbol{\delta}}}_t \\
=& C_\omega\left(\frac{1}{N}\sum_{i=0}^{N-1}\hat{\boldsymbol{\delta}}_t^{(i)} + \boldsymbol{e}_t\right) - \frac{1}{N}\sum_{i=0}^{N-1}\hat{\boldsymbol{\delta}}_t^{(i)} \\
=& \frac{1}{N}\sum_{i=0}^{N-1}\hat{\boldsymbol{\delta}}_t^{(i)} + \boldsymbol{e}_t + \left[C_\omega\left(\frac{1}{N}\sum_{i=0}^{N-1}\hat{\boldsymbol{\delta}}_t^{(i)} + \boldsymbol{e}_t\right) - \left(\frac{1}{N}\sum_{i=0}^{N-1}\hat{\boldsymbol{\delta}}_t^{(i)} + \boldsymbol{e}_t\right)\right] - \frac{1}{N}\sum_{i=0}^{N-1}\hat{\boldsymbol{\delta}}_t^{(i)} \\
=& \boldsymbol{e}_t - \boldsymbol{e}_{t+1},
\end{aligned}$$

this will give us

$$\bar{\boldsymbol{x}}_{t+1} + \boldsymbol{e}_{t+1} = \bar{\boldsymbol{x}}_t + \boldsymbol{e}_t + \bar{\boldsymbol{g}}_t.$$

Now given Assumption 3.1,

$$\begin{aligned}
f(\bar{\boldsymbol{x}}_{t+1} + \boldsymbol{e}_{t+1}) &= f(\bar{\boldsymbol{x}}_t + \boldsymbol{e}_t + \bar{\boldsymbol{g}}_t) \\
&\le f(\bar{\boldsymbol{x}}_t + \boldsymbol{e}_t) + \gamma\left\langle \nabla f(\bar{\boldsymbol{x}}_t + \boldsymbol{e}_t), \gamma^{-1}\bar{\boldsymbol{g}}_t\right\rangle + \frac{L}{2}\|\bar{\boldsymbol{g}}_t\|^2.
\end{aligned}$$

Take total expectation on both sides,

$$\begin{aligned}
\mathbb{E} f(\bar{\boldsymbol{x}}_{t+1} + \boldsymbol{e}_{t+1}) &\le \mathbb{E} f(\bar{\boldsymbol{x}}_t + \boldsymbol{e}_t) + \gamma\mathbb{E}\left\langle \nabla f(\bar{\boldsymbol{x}}_t + \boldsymbol{e}_t), \gamma^{-1}\hat{\boldsymbol{g}}_t\right\rangle + \frac{L}{2}\mathbb{E}\|\bar{\boldsymbol{g}}_t\|^2 \\
&= \mathbb{E} f(\bar{\boldsymbol{x}}_t + \boldsymbol{e}_t) - \frac{\gamma}{2}\mathbb{E}\|\nabla f(\bar{\boldsymbol{x}}_t + \boldsymbol{e}_t)\|^2 - \frac{\gamma}{2}\mathbb{E}\left\|\gamma^{-1}\hat{\boldsymbol{g}}_t\right\|^2 + \frac{\gamma}{2}\mathbb{E}\left\|\nabla f(\bar{\boldsymbol{x}}_t + \boldsymbol{e}_t) + \gamma^{-1}\hat{\boldsymbol{g}}_t\right\|^2 + \frac{L}{2}\mathbb{E}\|\bar{\boldsymbol{g}}_t\|^2,
\end{aligned}$$

where in the second step we apply the fact that for any $\boldsymbol{a}, \boldsymbol{b}$, it holds that $\langle \boldsymbol{a}, \boldsymbol{b}\rangle = -\frac{1}{2}\|\boldsymbol{a}\|^2 - \frac{1}{2}\|\boldsymbol{b}\|^2 + \frac{1}{2}\|\boldsymbol{a}+\boldsymbol{b}\|^2$.

Next, notice that

$$\begin{aligned}
&-\frac{\gamma}{2}\mathbb{E}\left\|\gamma^{-1}\hat{\boldsymbol{g}}_t\right\|^2 + L\mathbb{E}\|\bar{\boldsymbol{g}}_t\|^2 \\
=& -\frac{\gamma}{2}\mathbb{E}\left\|\frac{1}{N}\sum_{i=0}^{N-1} f_i\left(\boldsymbol{x}_t^{(i)}\right)\right\|^2 + \gamma^2 L\mathbb{E}\left\|\frac{1}{N}\sum_{i=0}^{N-1} f_i\left(\boldsymbol{x}_t^{(i)}; \boldsymbol{\zeta}_t^{(i)}\right)\right\|^2 \\
=& -\frac{\gamma}{2}\mathbb{E}\left\|\frac{1}{N}\sum_{i=0}^{N-1} f_i\left(\boldsymbol{x}_t^{(i)}\right)\right\|^2 + \gamma^2 L\mathbb{E}\left\|\frac{1}{N}\sum_{i=0}^{N-1} f_i\left(\boldsymbol{x}_t^{(i)}\right)\right\|^2 + \frac{\gamma^2\sigma^2 L}{N} \\
\le& \frac{\gamma^2\sigma^2 L}{N},
\end{aligned}$$

where in the last step we apply Assumption 3.2 and $\gamma \le 1/2L$. Move the gradient norm to the LHS, we get

$$\mathbb{E}\|\nabla f(\bar{\boldsymbol{x}}_t+\boldsymbol{e}_t)\|^2 \le \frac{2(f(\bar{\boldsymbol{x}}_t+\boldsymbol{e}_t)-f(\bar{\boldsymbol{x}}_{t+1}+\boldsymbol{e}_{t+1}))}{\gamma}+\frac{2\gamma\sigma^2 L}{N}+\mathbb{E}\left\|\nabla f(\bar{\boldsymbol{x}}_t+\boldsymbol{e}_t)+\gamma^{-1}\hat{\boldsymbol{g}}_t\right\|^2.$$

Now summing from $t=0$ to $T-1$ on both sides, we get

$$\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\|\nabla f(\bar{\boldsymbol{x}}_t+\boldsymbol{e}_t)\|^2 \le \frac{2(f(\boldsymbol{x}_0)-f^*)}{\gamma T}+\frac{2\gamma\sigma^2 L}{N}+\frac{1}{\gamma^2 T}\sum_{t=0}^{T-1}\mathbb{E}\|\hat{\boldsymbol{g}}_t+\gamma\nabla f(\bar{\boldsymbol{x}}_t+\boldsymbol{e}_t)\|^2,$$

where $f^*=\inf_{\boldsymbol{x}} f(\boldsymbol{x})$. Note that since $f=\frac{1}{N}\sum_{i=0}^{N-1} f_i$,

$$\begin{aligned}
\mathbb{E}\|\hat{\boldsymbol{g}}_t+\gamma\nabla f(\bar{\boldsymbol{x}}_t+\boldsymbol{e}_t)\|^2 &\le 2\gamma^2\mathbb{E}\left\|\frac{1}{N}\sum_{i=0}^{N-1}\nabla f_i\left(\boldsymbol{x}_t^{(i)}\right)-\frac{1}{N}\sum_{i=0}^{N-1}\nabla f_i(\bar{\boldsymbol{x}}_t)\right\|^2+2\gamma^2\mathbb{E}\left\|\frac{1}{N}\sum_{i=0}^{N-1}\nabla f_i(\bar{\boldsymbol{x}}_t)-\frac{1}{N}\sum_{i=0}^{N-1}\nabla f_i(\bar{\boldsymbol{x}}_t+\boldsymbol{e}_t)\right\|^2\\
&\le \frac{2\gamma^2}{N}\sum_{i=0}^{N-1}\mathbb{E}\left\|\nabla f_i\left(\boldsymbol{x}_t^{(i)}\right)-\nabla f_i(\bar{\boldsymbol{x}}_t)\right\|^2+\frac{2\gamma^2 L^2}{N}\sum_{i=0}^{N-1}\mathbb{E}\|\boldsymbol{e}_t\|^2\\
&\le \frac{2\gamma^2 L^2}{N}\sum_{i=0}^{N-1}\mathbb{E}\left\|\boldsymbol{x}_t^{(i)}-\bar{\boldsymbol{x}}_t\right\|^2+2\gamma^2 L^2\mathbb{E}\|\boldsymbol{e}_t\|^2,
\end{aligned}$$

where we apply Assumption 3.1 and Jensen Inequality. Push it back we obtain

$$\begin{aligned}
&\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\|\nabla f(\bar{\boldsymbol{x}}_t+\boldsymbol{e}_t)\|^2\\
\le&\frac{2(f(\boldsymbol{x}_0)-f^*)}{\gamma T}+\frac{2\gamma\sigma^2 L}{N}+\frac{2L^2}{NT}\sum_{t=0}^{T-1}\sum_{i=0}^{N-1}\mathbb{E}\left\|\boldsymbol{x}_t^{(i)}-\bar{\boldsymbol{x}}_t\right\|^2+\frac{2L^2}{T}\sum_{t=0}^{T-1}\mathbb{E}\|\boldsymbol{e}_t\|^2\\
\le&\frac{2(f(\boldsymbol{x}_0)-f^*)}{\gamma T}+\frac{2\gamma\sigma^2 L}{N}+\frac{16L^2}{(1-\omega)^2 NT}\sum_{t=0}^{T-1}\sum_{i=0}^{N-1}\mathbb{E}\left\|\boldsymbol{x}_t^{(i)}-\boldsymbol{x}_t\right\|^2,
\end{aligned}$$

where in the last step we apply Lemma A.2 and the fact that

$$\begin{aligned}
\sum_{t=0}^{T-1}\sum_{i=0}^{N-1}\mathbb{E}\left\|\bar{\boldsymbol{x}}_t-\boldsymbol{x}_t^{(i)}\right\|^2 &\le 2\sum_{t=0}^{T-1}\sum_{i=0}^{N-1}\mathbb{E}\|\bar{\boldsymbol{x}}_t-\boldsymbol{x}_t\|^2+2\sum_{t=0}^{T-1}\sum_{i=0}^{N-1}\mathbb{E}\left\|\boldsymbol{x}_t-\boldsymbol{x}_t^{(i)}\right\|^2\\
&=2\sum_{t=0}^{T-1}\sum_{i=0}^{N-1}\mathbb{E}\left\|\frac{1}{N}\sum_{i=0}^{N-1}\left(\boldsymbol{x}_t-\boldsymbol{x}_t^{(i)}\right)\right\|^2+2\sum_{t=0}^{T-1}\sum_{i=0}^{N-1}\mathbb{E}\left\|\boldsymbol{x}_t-\boldsymbol{x}_t^{(i)}\right\|^2\\
&\le 4\sum_{t=0}^{T-1}\sum_{i=0}^{N-1}\mathbb{E}\left\|\boldsymbol{x}_t-\boldsymbol{x}_t^{(i)}\right\|^2.
\end{aligned}$$

For notational simplicity, we denote

$$M=\frac{1}{NT}\sum_{t=0}^{T-1}\sum_{i=0}^{N-1}\left\|\boldsymbol{x}_t^{(i)}-\boldsymbol{x}_t\right\|^2.$$

Next, considering that

$$\begin{aligned}
\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\|\nabla f(\boldsymbol{x}_t)\|^2 &\le \frac{2}{T}\sum_{t=0}^{T-1}\mathbb{E}\|\nabla f(\boldsymbol{x}_t)-\nabla f(\bar{\boldsymbol{x}}_t+\boldsymbol{e}_t)\|^2+\frac{2}{T}\sum_{t=0}^{T-1}\mathbb{E}\|\nabla f(\bar{\boldsymbol{x}}_t+\boldsymbol{e}_t)\|^2\\
&\le \frac{2L^2}{T}\sum_{t=0}^{T-1}\mathbb{E}\|\boldsymbol{x}_t-\bar{\boldsymbol{x}}_t-\boldsymbol{e}_t\|^2+\frac{2}{T}\sum_{t=0}^{T-1}\mathbb{E}\|\nabla f(\bar{\boldsymbol{x}}_t+\boldsymbol{e}_t)\|^2\\
&\le \frac{4L^2}{T}\sum_{t=0}^{T-1}\mathbb{E}\|\boldsymbol{x}_t-\bar{\boldsymbol{x}}_t\|^2+\frac{4L^2}{T}\sum_{t=0}^{T-1}\mathbb{E}\|\boldsymbol{e}_t\|^2+\frac{2}{T}\sum_{t=0}^{T-1}\mathbb{E}\|\nabla f(\bar{\boldsymbol{x}}_t+\boldsymbol{e}_t)\|^2\\
&\le 4L^2 M+\frac{64\omega^2 L^2}{(1-\omega)^2}M+\frac{4(f(\boldsymbol{x}_0)-f^*)}{\gamma T}+\frac{4\gamma\sigma^2 L}{N}+\frac{32L^2}{(1-\omega)^2}M\\
&\le \frac{4(f(\boldsymbol{x}_0)-f^*)}{\gamma T}+\frac{4\gamma\sigma^2 L}{N}+\frac{100L^2 M}{(1-\omega)^2},
\end{aligned}$$

where we applied Assumption 3.1 and Lemma A.2. Finally apply Lemma A.1, we get

$$\begin{aligned}
\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\|\nabla f(\boldsymbol{x}_t)\|^2 &\le \frac{4(f(\boldsymbol{x}_0)-f^*)}{\gamma T}+\frac{4\gamma\sigma^2 L}{N}+\frac{100L^2 M}{(1-\omega)^2}\\
&\le \frac{4(f(\boldsymbol{x}_0)-f^*)}{\gamma T}+\frac{4\gamma\sigma^2 L}{N}+\frac{200\gamma^2(\sigma^2+3\varsigma^2)L^2}{(1-\omega)^4}+\frac{600\gamma^2 L^2}{(1-\omega)^4 T}\sum_{t=0}^{T-1}\mathbb{E}\|\nabla f(\boldsymbol{x}_t)\|^2.
\end{aligned}$$

Apply the condition that

$$\gamma \leq \frac{(1-\omega)^2}{35L},$$

we finally get

$$\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\|\nabla f(\boldsymbol{x}_t)\|^2 \leq \frac{8(f(\boldsymbol{x}_0)-f^*)}{\gamma T}+\frac{8\gamma\sigma^2 L}{N}+\frac{400\gamma^2(\sigma^2+3\varsigma^2)L^2}{(1-\omega)^4}.$$

Set the learning rate γ to be

$$\gamma=\left(\sigma\sqrt{\frac{T}{N}}+\frac{(\sigma^2+3\varsigma^2)^{\frac{1}{3}}L^{\frac{2}{3}}T^{\frac{1}{3}}}{(f(\boldsymbol{x}_0)-f^*)^{\frac{1}{3}}(1-\omega)^{\frac{4}{3}}}+\frac{35L}{(1-\omega)^2}\right)^{-1}$$

, this gives us

$$\begin{aligned}\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\|\nabla f(\boldsymbol{x}_t)\|^2 &\leq \frac{8\sigma(f(\boldsymbol{x}_0)-f^*+L)}{\sqrt{NT}}+\frac{408(f(\boldsymbol{x}_0)-f^*)^{\frac{2}{3}}(\sigma^2+3\varsigma^2)^{\frac{2}{3}}L^{\frac{2}{3}}}{(1-\omega)^{\frac{4}{3}}T^{\frac{2}{3}}}+\frac{280(f(\boldsymbol{x}_0)-f^*)L}{(1-\omega)^2T}\\ &\leq O\left(\frac{1}{\sqrt{NT}}+\frac{1}{(1-\omega)^{\frac{4}{3}}T^{\frac{2}{3}}}+\frac{1}{(1-\omega)^2T^2}\right),\end{aligned}$$

that completes the proof. ■

A.2. Technical Lemma

Lemma A.1. *In* CocktailSGD *algorithm, if*

$$\gamma \leq \frac{1-\omega}{3L},$$

it holds that

$$\frac{1}{NT}\sum_{i=0}^{N-1}\sum_{t=0}^{T-1}\mathbb{E}\left\|\boldsymbol{x}_t-\boldsymbol{x}_t^{(i)}\right\|^2 \leq \frac{2\gamma^2(\sigma^2+3\varsigma^2)}{(1-\omega)^2}+\frac{6\gamma^2}{(1-\omega)^2T}\sum_{t=0}^{T-1}\mathbb{E}\|\nabla f(\boldsymbol{x}_t)\|^2.$$

Proof. For any $t\geq 1$ and any $i\in\{0,\cdots,N-1\}$, we have

$$\begin{aligned}\mathbb{E}\left\|\boldsymbol{x}_t-\boldsymbol{x}_t^{(i)}\right\| &= \mathbb{E}\left\|\boldsymbol{g}_{t-1}^{(i)}+\left(\boldsymbol{\delta}_{t-1}^{(i)}-\hat{\boldsymbol{\delta}}_{t-1}^{(i)}\right)\right\|\\ &= \mathbb{E}\left\|\gamma\nabla f_i\left(\boldsymbol{x}_{t-1}^{(i)};\boldsymbol{\zeta}_{t-1}^{(i)}\right)-\left(\boldsymbol{\delta}_{t-1}^{(i)}-\hat{\boldsymbol{\delta}}_{t-1}^{(i)}\right)\right\|\\ &\leq \gamma\mathbb{E}\left\|\nabla f_i\left(\boldsymbol{x}_{t-1}^{(i)};\boldsymbol{\zeta}_{t-1}^{(i)}\right)\right\|+\mathbb{E}\left[\mathbb{E}_{\mathcal{C}}\left\|\boldsymbol{\delta}_{t-1}^{(i)}-\hat{\boldsymbol{\delta}}_{t-1}^{(i)}\right\|\right]\\ &\leq \gamma\mathbb{E}\left\|\nabla f_i\left(\boldsymbol{x}_{t-1}^{(i)};\boldsymbol{\zeta}_{t-1}^{(i)}\right)\right\|+\mathbb{E}\left[\sqrt{\mathbb{E}_{\mathcal{C}}\left\|\boldsymbol{\delta}_{t-1}^{(i)}-\hat{\boldsymbol{\delta}}_{t-1}^{(i)}\right\|^2}\right]\\ &\leq \gamma\mathbb{E}\left\|\nabla f_i\left(\boldsymbol{x}_{t-1}^{(i)};\boldsymbol{\zeta}_{t-1}^{(i)}\right)\right\|+\omega\mathbb{E}\left\|\boldsymbol{x}_{t-1}-\boldsymbol{x}_{t-1}^{(i)}\right\|\\ &\leq \gamma\sum_{k=0}^{t-1}\omega^{t-k-1}\mathbb{E}\left\|\nabla f_i\left(\boldsymbol{x}_k^{(i)};\boldsymbol{\zeta}_k^{(i)}\right)\right\|\end{aligned}$$

Now square on both sides, summing from $t=0$ to $T-1$, also summing from $i=0$ to $N-1$, we obtain

$$\begin{aligned}\sum_{i=0}^{N-1}\sum_{t=0}^{T-1}\mathbb{E}\left\|\boldsymbol{x}_t-\boldsymbol{x}_t^{(i)}\right\|^2 &\leq \gamma^2\sum_{i=0}^{N-1}\sum_{t=0}^{T-1}\left(\sum_{k=0}^{t-1}\omega^{t-k-1}\mathbb{E}\left\|\nabla f_i\left(\boldsymbol{x}_k^{(i)};\boldsymbol{\zeta}_k^{(i)}\right)\right\|\right)^2\\ &\leq \frac{\gamma^2}{(1-\omega)^2}\sum_{i=0}^{N-1}\sum_{t=0}^{T-1}\mathbb{E}\left\|\nabla f_i\left(\boldsymbol{x}_t^{(i)};\boldsymbol{\zeta}_t^{(i)}\right)\right\|^2.\end{aligned}$$

Note that

$$\begin{aligned}
&\sum_{i=0}^{N-1}\mathbb{E}\left\|\nabla f_i\left(\boldsymbol{x}_t^{(i)};\boldsymbol{\zeta}_t^{(i)}\right)\right\|^2\\
=&\sum_{i=0}^{N-1}\mathbb{E}\left\|\nabla f_i\left(\boldsymbol{x}_t^{(i)};\boldsymbol{\zeta}_t^{(i)}\right)-\nabla f_i\left(\boldsymbol{x}_t^{(i)}\right)\right\|^2+\sum_{i=0}^{N-1}\mathbb{E}\left\|\nabla f_i\left(\boldsymbol{x}_t^{(i)}\right)\right\|^2\\
\le&\sigma^2 N+3\sum_{i=0}^{N-1}\mathbb{E}\left\|\nabla f_i(\boldsymbol{x}_t)-\nabla f_i\left(\boldsymbol{x}_t^{(i)}\right)\right\|^2+3\sum_{i=0}^{N-1}\mathbb{E}\|\nabla f_i(\boldsymbol{x}_t)-\nabla f(\boldsymbol{x}_t)\|^2+3\sum_{i=0}^{N-1}\mathbb{E}\|\nabla f(\boldsymbol{x}_t)\|^2\\
\le&(\sigma^2+3\varsigma^2)N+3L^2\sum_{i=0}^{N-1}\mathbb{E}\left\|\boldsymbol{x}_t-\boldsymbol{x}_t^{(i)}\right\|^2+3N\mathbb{E}\|\nabla f(\boldsymbol{x}_t)\|^2,
\end{aligned}$$

where in the second and third step we apply Assumption 3.2, and in the final step we apply Assumption 3.1. We obtain

$$\sum_{i=0}^{N-1}\sum_{t=0}^{T-1}\mathbb{E}\left\|\boldsymbol{x}_t-\boldsymbol{x}_t^{(i)}\right\|^2\le\frac{\gamma^2(\sigma^2+3\varsigma^2)NT}{(1-\omega)^2}+\frac{3\gamma^2N}{(1-\omega)^2}\sum_{t=0}^{T-1}\mathbb{E}\|\nabla f(\boldsymbol{x}_t)\|^2+\frac{3\gamma^2L^2}{(1-\omega)^2}\sum_{i=0}^{N-1}\sum_{t=0}^{T-1}\mathbb{E}\left\|\boldsymbol{x}_t-\boldsymbol{x}_t^{(i)}\right\|^2.$$

In the derivation above, we apply the Assumption 3.1, 3.2 and 3.3. Now, applying the condition that

$$\gamma\le\frac{1-\omega}{3L},$$

we obtain

$$\sum_{i=0}^{N-1}\sum_{t=0}^{T-1}\mathbb{E}\left\|\boldsymbol{x}_t-\boldsymbol{x}_t^{(i)}\right\|^2\le\frac{2\gamma^2(\sigma^2+3\varsigma^2)NT}{(1-\omega)^2}+\frac{6\gamma^2N}{(1-\omega)^2}\sum_{t=0}^{T-1}\mathbb{E}\|\nabla f(\boldsymbol{x}_t)\|^2.$$

That completes the proof. ∎

Lemma A.2. *In Algorithm 1, the averaged compression error is bounded below by,*

$$\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\|\boldsymbol{e}_t\|^2\le\frac{4\omega^2}{(1-\omega)^2NT}\sum_{i=0}^{N-1}\sum_{t=0}^{T-1}\mathbb{E}\left\|\boldsymbol{x}_t-\boldsymbol{x}_t^{(i)}\right\|^2,$$

Proof. For any $t\ge0$, by the definition of $\boldsymbol{e}_{t+1}$, we can get for any $\eta>0$,

$$\begin{aligned}
\mathbb{E}\|\boldsymbol{e}_{t+1}\|^2=&\mathbb{E}\left[\mathbb{E}_{\mathcal{C}}\left\|\frac{1}{N}\sum_{i=0}^{N-1}\hat{\boldsymbol{\delta}}_t^{(i)}+\boldsymbol{e}_t-C_\omega\left(\frac{1}{N}\sum_{i=0}^{N-1}\hat{\boldsymbol{\delta}}_t^{(i)}+\boldsymbol{e}_t\right)\right\|^2\right]\\
\le&\omega^2\mathbb{E}\left\|\frac{1}{N}\sum_{i=0}^{N-1}\hat{\boldsymbol{\delta}}_t^{(i)}+\boldsymbol{e}_t\right\|^2\\
\le&\omega^2(1+\eta)\mathbb{E}\|\boldsymbol{e}_t\|^2+\omega^2(1+1/\eta)\mathbb{E}\left\|\frac{1}{N}\sum_{i=0}^{N-1}\hat{\boldsymbol{\delta}}_t^{(i)}\right\|^2,
\end{aligned}$$

where in the second step we apply Assumption 3.4. Summing from $t=0$ to $T-1$ and apply the fact that $\boldsymbol{e}_0=0$, we get

$$\sum_{t=0}^{T-1}\mathbb{E}\|\boldsymbol{e}_t\|^2\le\omega^2(1+\eta)\sum_{t=0}^{T-1}\mathbb{E}\|\boldsymbol{e}_t\|^2+\omega^2(1+1/\eta)\sum_{t=0}^{T-1}\mathbb{E}\left\|\frac{1}{N}\sum_{i=0}^{N-1}\hat{\boldsymbol{\delta}}_t^{(i)}\right\|^2.$$

Take $\eta=\frac{1-\omega^2}{2\omega^2}$, we get

$$\sum_{t=0}^{T-1}\mathbb{E}\|\boldsymbol{e}_t\|^2\le\frac{2\omega^2(1+\omega^2)}{(1-\omega^2)^2}\sum_{t=0}^{T-1}\mathbb{E}\left\|\frac{1}{N}\sum_{i=0}^{N-1}\hat{\boldsymbol{\delta}}_t^{(i)}\right\|^2.$$

For the last term,

$$\begin{aligned}
\mathbb{E}\left\|\frac{1}{N}\sum_{i=0}^{N-1}\hat{\boldsymbol{\delta}}_t^{(i)}\right\|^2\le&\frac{1}{N}\sum_{i=0}^{N-1}\mathbb{E}\left\|\hat{\boldsymbol{\delta}}_t^{(i)}\right\|^2\\
\le&\frac{2}{N}\sum_{i=0}^{N-1}\mathbb{E}\left[\mathbb{E}_{\mathcal{C}}\left\|\hat{\boldsymbol{\delta}}_t^{(i)}-\boldsymbol{\delta}_t^{(i)}\right\|^2+\frac{2}{N}\sum_{i=0}^{N-1}\mathbb{E}\left\|\boldsymbol{\delta}_t^{(i)}\right\|^2\right]\\
\le&\frac{2(1+\omega^2)}{N}\sum_{i=0}^{N-1}\mathbb{E}\left\|\boldsymbol{x}_t-\boldsymbol{x}_t^{(i)}\right\|^2.
\end{aligned}$$

Combine them together we obatin

$$\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\|\boldsymbol{e}_t\|^2\le\frac{4\omega^2}{(1-\omega)^2NT}\sum_{i=0}^{N-1}\sum_{t=0}^{T-1}\mathbb{E}\left\|\boldsymbol{x}_t-\boldsymbol{x}_t^{(i)}\right\|^2,$$

where we apply the fact that $\omega<1$. That completes the proof. ∎

A.3. Proof to Lemma 3.6

Proof. For any $\boldsymbol{x} \in \mathbb{R}^d$, it trivially holds that

$$\|\mathcal{C}_K(\boldsymbol{x}) - \boldsymbol{x}\|^2 \leq \|\mathcal{C}_R(\boldsymbol{x}) - \boldsymbol{x}\|^2.$$

On the other hand, following (Stich et al., 2018) Lemma A.1, we can show that for $\mathcal{C}_R$, if eventually q coordinates are kept, then

$$\mathbb{E}_R\|\mathcal{C}_R(\boldsymbol{x}) - \boldsymbol{x}\|^2 = \left(1 - \frac{q}{d}\right)\|\boldsymbol{x}\|^2.$$

Notice that after $\mathcal{C}_K \circ \mathcal{C}_R$, only $\min\{K, pd\}$ coordinates are kept. Then,

$$\begin{aligned}
\mathbb{E}_{QR}\|\mathcal{C}_Q \circ \mathcal{C}_K \circ \mathcal{C}_R(\boldsymbol{x}) - \boldsymbol{x}\|^2 &= \mathbb{E}_R\mathbb{E}_Q\|\mathcal{C}_Q \circ \mathcal{C}_K \circ \mathcal{C}_R(\boldsymbol{x}) - \mathcal{C}_K \circ \mathcal{C}_R(\boldsymbol{x})\|^2 + \mathbb{E}_R\|\mathcal{C}_K \circ \mathcal{C}_R(\boldsymbol{x}) - \boldsymbol{x}\|^2 \\
&\leq \left(1 - \frac{1}{1 + \min(d/s^2, \sqrt{d}/s)}\right)\mathbb{E}_R\|\mathcal{C}_K \circ \mathcal{C}_R(\boldsymbol{x})\|^2 + \left(1 - \frac{\min\{K, pd\}}{d}\right)\|\boldsymbol{x}\|^2 \\
&= \left(1 - \frac{1}{1 + \min(d/s^2, \sqrt{d}/s)}\right) \cdot \frac{\min\{K, pd\}}{d}\|\boldsymbol{x}\|^2 + \left(1 - \frac{\min\{K, pd\}}{d}\right)\|\boldsymbol{x}\|^2 \\
&= \left(1 - \frac{\min\{K, pd\}}{d + d \cdot \min\{d/s^2, \sqrt{d}/s\}}\right)\|\boldsymbol{x}\|^2.
\end{aligned}$$

Comparing it with Assumption 3.4, it gives

$$\omega = \sqrt{1 - \frac{\min\{K, pd\}}{d + d \cdot \min\{d/s^2, \sqrt{d}/s\}}}.$$

That completes the proof. ∎

A.4. Avoid Sending Indices with Random Sampling

We now demonstrate the effect of random selection before compression on a toy example. Consider the following two settings: (1) Compressing vector with density q and length n with b bits on average per nonzero value. (2) Random selecting the parameters of vector at rate p, and then run (1).

The number of bits for the first one is

$$bqn - n(q\log_2(q) + (1-q)\log_2(1-q))$$

for the second one, it's

$$bqn - np\left(\frac{q}{p}\log_2\left(\frac{q}{p}\right) + \left(1 - \frac{q}{p}\right)\log_2\left(1 - \frac{q}{p}\right)\right).$$

This is n times

$$b - \left(\log_2\left(\frac{q}{p}\right) + \left(\frac{p}{q} - 1\right)\log_2\left(1 - \frac{q}{p}\right)\right).$$

Let $x = p/q$. Then

$$b + \left(\log_2(x) + (x-1)\log_2\left(\frac{x}{x-1}\right)\right).$$

This is

$$b + (x\log_2(x) - (x-1)\log_2(x-1)).$$

Heuristically, set

$$b = (x\log_2(x) - (x-1)\log_2(x-1)).$$

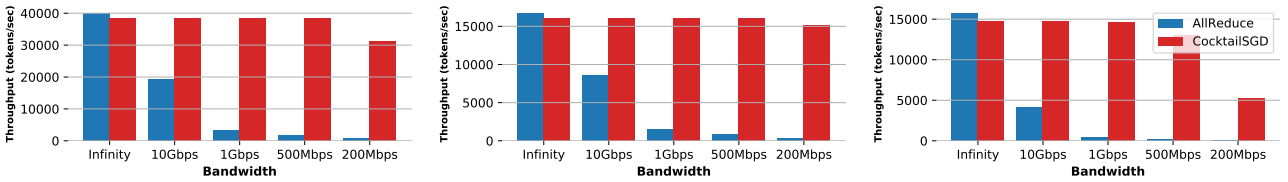
$$b\log(2) = (x\log(x) - (x-1)\log(x-1)) = \log(y) + 1,$$

for some y between $x-1$ and x. So,

$$y = \exp(b\log(2) - 1) = \frac{1}{e}2^b,$$

and so since $x - 1 \leq y \leq x$,

$$\frac{1}{e}2^b \leq x \leq \frac{1}{e}2^b + 1.$$



(a) OPT-1.3B (b) GPT-J-6B (c) GPT-NeoX-20B

Figure 8: Throughput under different network configurations. 'Infinity' is when all communications are disabled, which is an upper bound on the throughput.

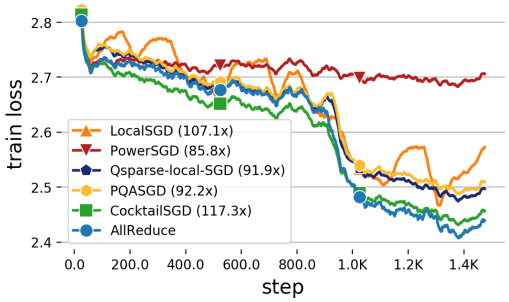


Figure 9: Comparison with other communication-efficient schemes on WIKITEXT-103 dataset.

B. Throughput under Different Network Configurations

We simulate slower network conditions and utilize Linux traffic control to control the communication bandwidth between instances. For the models OPT-1.3B and GPT-J-6B, we utilize 4 data parallel workers with 2 A100 GPUs each. For the GPT-NeoX-20B model, we use 4 data parallel workers with 8 A100 GPUs each, for a total of 32 A100 GPUs.

Figure 8 presents the training throughput under different network configuration of OPT-1.3B, GPT-J-6B, and GPT-NeoX-20B. COCKTAILSGD can hide the communication in the local computation, thus maintaining nearly constant training throughput if the communication time is less than the computation time. Under a slower 500Mbps network, COCKTAILSGD introduces 22.0×, 19.7×, and 47.2× improvement of the training throughput for OPT-1.3B, GPT-J-6B, and GPT-NeoX-20B, respectively, when comparing with ALLREDUCE. Under 1Gbps network, COCKTAILSGD also achieve substantial improvements, with 11.4×, 10.3×, and 30.2× improvement for the respective models. In fact, this throughput is almost the same as the throughput of ALLREDUCE with "infinite bandwidth" (when all communications are disabled), with only 5-7% slower. This unlocks the use of geographically distributed machines connected with slow networks to effectively train large-scale language models while maintaining a reasonable hardware efficiency and convergence speed.

C. Comparison with Other Communication-Efficient Schemes

In this section, we present a comparative evaluation of several more communication-efficient schemes, including LocalSGD (Stich, 2018), PowerSGD (Vogels et al., 2019), and algorithms that incorporate multiple compression methods, such as Qsparse-local-SGD (Basu et al., 2019) and PQASGD (Jiang & Agrawal, 2018). To evaluate the performance of these methods, we conducted experiments on the WIKITEXT-103 dataset and compared the achieved loss values under a comparable compression ratio. As depicted in Figure 9, our results demonstrate that both ALLREDUCE and COCKTAILSGD exhibit comparable convergence behavior, while LocalSGD, PowerSGD, Qsparse-local-SGD, and PQASGD show clear gaps in convergence. The results of LocalSGD and PowerSGD suggest that it is challenging for a single technique to achieve an aggressive compression ratio. On the other hand, the results of Qsparse-local-SGD and PQASGD show that it is crucial to take into account communication overhead in both directions. Our findings suggest that COCKTAILSGD is a promising approach for optimizing LLMs and other large-scale models.

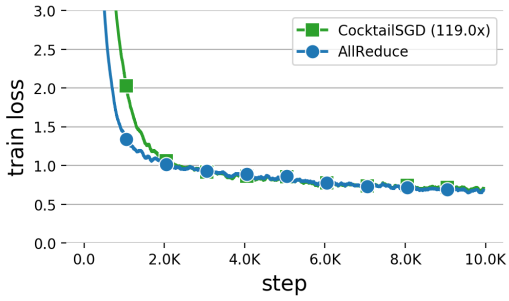


Figure 10: Fine-tuning ViT-huge on ImageNet-1k.

D. Training Image Models

We further carried out experiments by fine-tuning ViT-huge (Dosovitskiy et al.)[2] on ImageNet-1k and observed comparable convergence between ALLREDUCE and COCKTAILSGD. Figure 10 presents the result. Both approaches started with a loss value of 6.91 and were able to reduce it to 0.683 and 0.685 for ALLREDUCE and COCKTAILSGD, respectively, after processing 1.28 million images. We used the same compression configuration as in our current paper for this experiment. Therefore, while this work mainly focused on LLMs, the proposed approach can also be applied to other types of models, including large image models such as ViT.

[2] https://huggingface.co/google/vit-huge-patch14-224-in21k