

---

# Compressed SGD with Memory

---

Anonymous Author(s)

Affiliation

Address

email

## Abstract

1 Nowadays machine learning applications require stochastic optimization algorithms  
2 that can be implemented on distributed systems. The communication overhead  
3 of the algorithms is a key bottleneck that hinders perfect scalability. Various  
4 recent work proposed to use quantization or sparsification techniques to reduce the  
5 amount of data that needs to be communicated, for instance by only sending the  
6 most significant entries of the stochastic gradient (top- $k$  sparsification). Whilst this  
7 scheme shows good performance in practice it eluded theoretical analysis so far.  
8 In this work we analyze a variant of Stochastic Gradient Descent (SGD) with  
9  $k$ -sparsification (for instance top- $k$  or random- $k$ ) and show that this scheme con-  
10 verges at the same rate as vanilla SGD. That is, the communication can be reduced  
11 by a factor of the dimension of the problem (sometimes even more) whilst still  
12 converging at the same rate. We present numerical experiments to illustrate the  
13 theoretical findings and especially the better scalability for distributed applications.

## 14 1 Introduction

15 Stochastic Gradient Descent (SGD) [25] and variants thereof (e.g. [7, 12]) are among the most popular  
16 optimization algorithms in machine- and deep-learning [4]. SGD consists of iterations of the form

$$\mathbf{x}_{t+1} := \mathbf{x}_t - \eta_t \mathbf{g}_t, \quad (1)$$

17 for iterates  $\mathbf{x}_t, \mathbf{x}_{t+1} \in \mathbb{R}^d$ , stepsize (or learning rate)  $\eta_t > 0$ , and stochastic gradient  $\mathbf{g}_t$  with  
18 the property  $\mathbb{E} \mathbf{g}_t = \nabla f(\mathbf{x}_t)$ , for a loss function  $f: \mathbb{R}^d \rightarrow \mathbb{R}$ . SGD addresses the computational  
19 bottleneck of full gradient descent, as the stochastic gradients can in general be computed much  
20 more efficiently than a full gradient  $\nabla f(\mathbf{x}_t)$ . However, note that in general both  $\mathbf{g}_t$  and  $\nabla f(\mathbf{x}_t)$   
21 are *dense* vectors<sup>1</sup> of size  $d$ , i.e. SGD does not address the communication bottleneck of gradient  
22 descent, which occurs as a roadblock both in distributed as well as parallel training. In the setting of  
23 distributed training, communicating the stochastic gradients to the other worker has been reported as  
24 a major limiting factor for many large scale deep learning applications, see e.g. [3, 17, 30, 38]. The  
25 same bottleneck can also appear for parallel training, e.g. even in the increasingly common setting of  
26 a single multicore machine or device, where locking and bandwidth of memory write operations for  
27 the common shared parameter  $\mathbf{x}_t$  often form the main bottleneck, see e.g. [11, 14, 21].

28 A possible remedy to address these issues is to *enforce* sparsity of the updates by just applying the  
29 update  $\text{sparse}(\mathbf{g}_t)$ , where  $\text{sparse}: \mathbb{R}^d \rightarrow \mathbb{R}^d$  generates a lossy quantization of the gradient. We  
30 discuss different schemes below. It has been reported that too aggressive sparsification can hurt  
31 the performance, unless it is implemented in a clever way: 1Bit-SGD [30, 33] combines gradient  
32 quantization with an error accumulation technique. Roughly speaking, the method keeps track of a  
33 memory vector  $\mathbf{m}$  which contains the sum of the information that has been suppressed thus far, i.e.

---

<sup>1</sup>Note that the stochastic gradients  $\mathbf{g}_t$  are dense vectors for the setting of training neural networks. The  $\mathbf{g}_t$  themselves can be sparse for generalized linear models under the additional assumption that the data is sparse.

34  $\mathbf{m}_{t+1} := \mathbf{m}_t + \mathbf{g}_t - \text{sparse}(\mathbf{g}_t)$ , and injects this information back in the next iteration, by transmitting  
 35  $\text{sparse}(\mathbf{m}_{t+1} + \mathbf{g}_{t+1})$  instead of only  $\text{sparse}(\mathbf{g}_{t+1})$ . Updates of this kind are not unbiased and there  
 36 is also no control over the delay after which the single coordinates are applied. These are reasons  
 37 why there exists no theoretical analysis of this scheme up to now.

38 In this paper we analyze SGD with memory and  $k$ -sparsifications operators, such as top- $k$ . The  
 39 analysis also supports ultra-sparsification operators for which  $k < 1$ , i.e. where *less than one*  
 40 coordinate of the stochastic gradient is applied on average in (1). We not only provide the first  
 41 convergence result of this method, but the result also shows that the method converges *at the same*  
 42 *rate* as vanilla SGD.

## 43 1.1 Related Work

44 There are several ways to reduce the communication in SGD. For instance by simply increasing the  
 45 amount of computation before communication, i.e. by using large mini-batches (see e.g. [9, 37]), or  
 46 by designing communication-efficient schemes [39]. These approaches are a bit orthogonal to the  
 47 methods we consider in this paper, which focus on quantization or sparsification of the gradient.

48 Several papers consider approaches that limit the number of bits to represent floating point num-  
 49 bers [10, 20, 27]. Recent work proposes adaptive tuning of the compression ratio [5]. Unbiased  
 50 quantization operators not only limit the number of bits, but quantize the stochastic gradients in such  
 51 a way that they are still unbiased estimators of the gradient [3, 36]. The ZipML framework applies  
 52 this technique also to the data [38]. Sparsification methods reduce the number of non-zero entries in  
 53 the stochastic gradient [3, 35].

54 A very aggressive sparsification method is to keep only very few coordinates of the stochastic gradient  
 55 by considering only the coordinates with the largest magnitudes [1, 6]. In contrast to the unbiased  
 56 schemes it is clear that such methods can only work by using some kind of error accumulation or  
 57 feedback procedure, similar to the one we have already discussed [30, 33], as otherwise certain  
 58 coordinates could simply never be updated. However, in certain applications no feedback mechanism  
 59 is needed [34]. Also more elaborated sparsification schemes have been introduced [17].

60 Asynchronous updates provide an alternative solution to disguise the communication overhead  
 61 to a certain amount [15]. However, those methods usually rely on a sparsity assumption on the  
 62 updates [21, 27], which is not realistic e.g. in deep learning. We like to advocate that combining  
 63 gradient sparsification with those asynchronous schemes seems to be a promising approach, as  
 64 it combines the best of both worlds. Other scenarios that could profit from sparsification are  
 65 heterogeneous systems or specialized hardware, e.g. accelerators [8, 38].

66 Convergence proofs for SGD [25] typically rely on averaging of the iterates [19, 23, 26], though also  
 67 convergence of the last iterate can be proven [31]. For our convergence proof we rely on averaging  
 68 techniques that give more weight to more recent iterates [13, 24, 31], as well as the perturbed iterate  
 69 framework from Mania et al. [18] and techniques from [14].

## 70 1.2 Contributions

71 We consider finite-sum convex optimization problems  $f: \mathbb{R}^d \rightarrow \mathbb{R}$  of the form

$$f(\mathbf{x}) = \frac{1}{n} \sum_{i=1}^n f_i(\mathbf{x}), \quad \mathbf{x}^* := \arg \min_{\mathbf{x} \in \mathbb{R}^d} f(\mathbf{x}), \quad f^* := f(\mathbf{x}^*), \quad (2)$$

72 where each  $f_i$  is  $L$ -smooth<sup>2</sup> and  $f$  is  $\mu$ -strongly convex<sup>3</sup>. We consider a sequential sparsified SGD  
 73 algorithm with error accumulation technique and prove convergence for  $k$ -sparsification operators,  
 74  $0 < k \leq d$  (for instance the operators top- $k$  or random- $k$ ). For appropriately chosen stepsizes and an  
 75 averaged iterate  $\bar{\mathbf{x}}_T$  after  $T$  steps we show convergence

$$\mathbb{E} f(\bar{\mathbf{x}}_T) - f^* = \mathcal{O}\left(\frac{G^2}{\mu T}\right) + \mathcal{O}\left(\frac{\frac{d^2}{k^2} G^2 \kappa}{\mu T^2}\right) + \mathcal{O}\left(\frac{\frac{d^3}{k^3} G^2}{\mu T^3}\right), \quad (3)$$

<sup>2</sup>  $f_i(\mathbf{y}) \leq f_i(\mathbf{x}) + \langle \nabla f_i(\mathbf{x}), \mathbf{y} - \mathbf{x} \rangle + \frac{L}{2} \|\mathbf{y} - \mathbf{x}\|^2, \forall \mathbf{x}, \mathbf{y} \in \mathbb{R}^d, i \in [n]$ .

<sup>3</sup>  $f(\mathbf{y}) \geq f(\mathbf{x}) + \langle \nabla f(\mathbf{x}), \mathbf{y} - \mathbf{x} \rangle + \frac{\mu}{2} \|\mathbf{y} - \mathbf{x}\|^2, \forall \mathbf{x}, \mathbf{y} \in \mathbb{R}^d$ .

for  $\kappa = \frac{L}{\mu}$  and  $G^2 \geq \mathbb{E} \|\nabla f_i(\mathbf{x}_t)\|^2$ . Not only is this, to the best of our knowledge, the first convergence result for sparsified SGD with memory, but the result also shows that for  $T = \Omega(\frac{d}{k}\sqrt{\kappa})$  the first term is dominating and the convergence rate is the same as for vanilla SGD.

We introduce the method formally in Section 2 and show a sketch of the convergence proof in Section 3. In Section 4 we include a few numerical experiments for illustrative purposes. The experiments highlight that top- $k$  sparsification yields a very effective compression method and does not hurt convergence. Our multicore simulations demonstrate that SGD with memory scales better than asynchronous SGD thanks to the enforced sparsity of the updates. It also drastically decreases the communication cost without sacrificing the rate of convergence. We like to stress that the effectiveness of the scheme has already been demonstrated in practice [1, 6, 17, 30, 33].

Although we do not yet provide convergence guarantees for parallel and asynchronous variants of the scheme, this is the main application of this method. For instance, we like to highlight that asynchronous SGD schemes [2, 21] could profit from the gradient sparsification. To demonstrate this use-case, we include in Section 4 a set of experiments for a multicore implementation.

## 2 SGD with Memory

In this section we present the sparse SGD algorithm with memory. First we introduce the sparsification operators that we use to drastically reduce the communication cost in comparison with vanilla SGD.

### 2.1 Sparsification Operators

We consider  $k$ -sparsification operators, defined as follows:

**Definition 2.1** ( $k$ -sparsification operator). *For a parameter  $0 < k \leq d$ , a (random) operator  $\text{sparse}_k: \mathbb{R}^d \rightarrow \mathbb{R}^d$  that satisfies the contraction property*

$$\mathbb{E} \|\mathbf{x} - \text{sparse}_k(\mathbf{x})\|^2 \leq \left(1 - \frac{k}{d}\right) \|\mathbf{x}\|^2, \quad (4)$$

for all  $\mathbf{x} \in \mathbb{R}^d$  is a  $k$ -sparsification operator.

The contraction property does not require  $\text{sparse}_k(\mathbf{x})$  to be actually sparse, also dense vectors can satisfy (4), but we will focus on sparse operators in this contribution, such as these two examples:

**Definition 2.2.** *For a parameter  $1 \leq k \leq d$ , the operators  $\text{top}_k: \mathbb{R}^d \rightarrow \mathbb{R}^d$  and  $\text{rand}_k: \mathbb{R}^d \times \Omega_k \rightarrow \mathbb{R}^d$ , where  $\Omega_k = \binom{[d]}{k}$  denotes the set of all  $k$  element subsets of  $[d]$ , are defined for  $\mathbf{x} \in \mathbb{R}^d$  as*

$$(\text{top}_k(\mathbf{x}))_i := \begin{cases} (\mathbf{x})_{\pi(i)}, & \text{if } i \leq k, \\ 0 & \text{otherwise,} \end{cases} \quad (\text{rand}_k(\mathbf{x}, \omega))_i := \begin{cases} (\mathbf{x})_i, & \text{if } i \in \omega, \\ 0 & \text{otherwise,} \end{cases} \quad (5)$$

where  $\pi$  is a permutation of  $[d]$  such that  $(|\mathbf{x}|)_{\pi(i)} \geq (|\mathbf{x}|)_{\pi(i+1)}$  for  $i = 1, \dots, d-1$ . We abbreviate  $\text{rand}_k(\mathbf{x})$  whenever the second argument is chosen uniformly at random,  $\omega \sim_{\text{u.a.r.}} \Omega_k$ .

It is easy to see that both operators satisfy (4). For completeness the proof is included in Appendix A.1.

**Remark 2.3** (Ultra-sparsification). *We like to highlight that not only those two operators satisfy (4), but many others. As a notable variant we like to point out that by picking a random coordinate of a vector with probability  $\frac{k}{d}$ , for  $0 < k \leq 1$ , property (4) holds even if  $k < 1$ . I.e. it suffices to transmit on average less than one coordinate per iteration (this would then correspond to a mini-batch update).*

### 2.2 Variance Blow-up for Unbiased Updates

Before introducing SGD with memory we first discuss a motivating example. Consider the following variant of SGD, where  $d - k$  random coordinates of the stochastic gradient are dropped:

$$\mathbf{x}_{t+1} := \mathbf{x}_t - \eta_t \mathbf{g}_t, \quad \mathbf{g}_t := \frac{d}{k} \cdot \text{rand}_k(\nabla f_i(\mathbf{x}_t)), \quad (6)$$

where  $i \sim_{\text{u.a.r.}} [n]$ . It is important to note that the update is unbiased, i.e.  $\mathbb{E} \mathbf{g}_t = \nabla f(\mathbf{x})$ . For carefully chosen stepsizes  $\eta_t$  this algorithm converges at rate  $\mathcal{O}(\frac{\sigma^2}{t})$  on strongly convex and smooth functions  $f$ , where  $\sigma^2$  is an upper bound on the variance, see for instance [40]. We have

$$\sigma^2 = \mathbb{E} \left\| \frac{d}{k} \text{rand}_k(\nabla f_i(\mathbf{x})) - \nabla f(\mathbf{x}) \right\|^2 \leq \mathbb{E} \left\| \frac{d}{k} \text{rand}_k(\nabla f_i(\mathbf{x})) \right\|^2 \leq \frac{d}{k} \mathbb{E}_i \|\nabla f_i(\mathbf{x})\|^2 \leq \frac{d}{k} G^2$$

**Algorithm 1** MEM-SGD

---

```

1: Initialize variables  $\mathbf{x}_0$  and  $\mathbf{m}_0 = \mathbf{0}$ 
2: for  $t$  in  $0 \dots T - 1$  do
3:   Sample  $i_t$  uniformly in  $[n]$ 
4:    $\mathbf{g}_t \leftarrow \text{sparse}_k(\mathbf{m}_t + \eta_t \nabla f_{i_t}(\mathbf{x}_t))$ 
5:    $\mathbf{x}_{t+1} \leftarrow \mathbf{x}_t - \mathbf{g}_t$ 
6:    $\mathbf{m}_{t+1} \leftarrow \mathbf{m}_t + \eta_t \nabla f_{i_t}(\mathbf{x}_t) - \mathbf{g}_t$ 
7: end for

```

---

**Algorithm 2** PARALLEL-MEM-SGD

---

```

1: Initialize shared variable  $\mathbf{x}$ 
   and  $\mathbf{m}_0^w = \mathbf{0}, \forall w \in [W]$ 
2: parallel for  $w$  in  $1 \dots W$  do
3:   for  $t$  in  $0 \dots T - 1$  do
4:     Sample  $i_t^w$  uniformly in  $[n]$ 
5:      $\mathbf{g}_t^w \leftarrow \text{sparse}_k(\mathbf{m}_t^w + \eta_t \nabla f_{i_t^w}(\mathbf{x}))$ 
6:      $\mathbf{x} = \mathbf{x} - \mathbf{g}_t^w$   $\triangleright$  shared memory
7:      $\mathbf{m}_{t+1}^w \leftarrow \mathbf{m}_t^w + \eta_t \nabla f_{i_t^w}(\mathbf{x}) - \mathbf{g}_t^w$ 
8:   end for
9: end parallel for

```

---

Figure 1: *Left*: The MEM-SGD algorithm. *Right*: Implementation for multicore experiments.

115 where we used the variance decomposition  $\mathbb{E} \|X - \mathbb{E} X\|^2 = \mathbb{E} \|X\|^2 - \|\mathbb{E} X\|^2$  and the standard  
116 assumption  $\mathbb{E}_i \|\nabla f_i(\mathbf{x})\|^2 \leq G^2$ . Hence, when  $k$  is small this algorithm requires  $d$  times more  
117 iterations to achieve the same error guarantee as vanilla SGD with  $k = d$ .

118 It is well known, that by using mini-batches the variance of the gradient estimator can be reduced. If  
119 we consider in (6) the estimator  $\mathbf{g}_t := \frac{d}{k} \cdot \text{rand}_k(\frac{1}{\tau} \sum_{i \in \mathcal{I}_\tau} \nabla f_i(\mathbf{x}_t))$  for  $\tau = \lceil \frac{k}{d} \rceil$ , and  $\mathcal{I}_\tau \sim_{\text{u.a.r.}} \binom{[n]}{k}$   
120 instead, we have

$$\sigma^2 = \mathbb{E} \|\mathbf{g}_t - \nabla f(\mathbf{x}_t)\|^2 \leq \mathbb{E} \left\| \frac{d}{k} \cdot \text{rand}_k\left(\frac{1}{\tau} \sum_{i \in \mathcal{I}_\tau} \nabla f_i(\mathbf{x}_t)\right) \right\|^2 \leq \frac{d}{k\tau} \mathbb{E}_i \|\nabla f_i(\mathbf{x}_t)\|^2 \leq G^2. \quad (7)$$

121 This shows, that when using mini-batches of appropriate size, the sparsification of the gradient does  
122 not hurt the convergence. However, by increasing the mini-batch size, we increase the computation  
123 by a factor of  $\frac{d}{k}$ .

124 These two observation seem to indicate that the factor  $\frac{d}{k}$  is inevitably lost, either by increased number  
125 of iterations or increased computation. However, this is no longer true when the information in (6)  
126 is not dropped, but kept in memory. To illustrate this, assume  $k = 1$  and that index  $i$  has not been  
127 selected by the  $\text{rand}_1$  operator in iterations  $t = t_0, \dots, t_{s-1}$ , but is selected in iteration  $t_s$ . Then  
128 the memory  $\mathbf{m}_{t_s} \in \mathbb{R}^d$  contain this past information  $(\mathbf{m}_{t_s})_i = \sum_{t=t_0}^{t_s-1} (\nabla f_{i_t}(\mathbf{x}_t))_i$ . Intuitively, we  
129 would expect that the variance of this estimator is now reduced by a factor of  $s$  compared to the naïve  
130 estimator in (6), similar to the mini-batch update in (7). Indeed, SGD with memory converges at the  
131 same rate as vanilla SGD, as we will demonstrate below.

**2.3 SGD with Memory: Algorithm and Convergence Results**

132 We consider the following algorithm for parameter  $0 < k \leq d$ , and  $k$ -sparsification operator  
133  $\text{sparse}_k: \mathbb{R}^d \rightarrow \mathbb{R}^d$  (cf. Definition 2.1):

$$\mathbf{x}_{t+1} := \mathbf{x}_t - \mathbf{g}_t, \quad \mathbf{g}_t := \text{sparse}_k(\mathbf{m}_t + \eta_t \nabla f_{i_t}(\mathbf{x}_t)), \quad \mathbf{m}_{t+1} := \mathbf{m}_t + \eta_t \nabla f_{i_t}(\mathbf{x}_t) - \mathbf{g}_t, \quad (8)$$

135 where  $i_t \sim_{\text{u.a.r.}} [n]$ ,  $\mathbf{m}_0 = \mathbf{0}$  and  $\{\eta_t\}_{t \geq 0}$  denotes a sequence of stepsizes. The pseudocode is given  
136 in Algorithm 1. Note that the gradients get multiplied with the stepsize  $\eta_t$  at the timestep  $t$  when they  
137 put into memory, and not when they are (partially) retrieved from the memory.

138 We state the precise convergence result for Algorithm 1 in Theorem 2.4 below. In Remark 2.6 we  
139 give a simplified statement in big- $O$  notation for a specific choice of the stepsizes  $\eta_t$ .

140 **Theorem 2.4.** *Let  $f_i$  be  $L$ -smooth,  $f$  be  $\mu$ -strongly convex,  $0 < k \leq d$ ,  $\mathbb{E}_i \|\nabla f_i(\mathbf{x}_t)\|^2 \leq G^2$  for  
141  $t = 0, \dots, T - 1$ , where  $\{\mathbf{x}_t\}_{t \geq 0}$  are generated according to (8) for stepsizes  $\eta_t = \frac{8}{\mu(a+t)}$  and shift  
142 parameter  $a > 1$ . Then for  $\alpha > 4$  such that  $\frac{(\alpha+1)\frac{d}{k} + \rho}{\rho+1} \leq a$ , with  $\rho := \frac{4\alpha}{(\alpha-4)(\alpha+1)^2}$ , it holds*

$$\mathbb{E} f(\bar{\mathbf{x}}_T) - f^* \leq \frac{4T(T+2a)}{\mu S_T} G^2 + \frac{\mu a^3}{8S_T} \|\mathbf{x}_0 - \mathbf{x}^*\|^2 + \frac{64T(1+2\frac{L}{\mu})}{\mu S_T} \left( \frac{4\alpha}{\alpha-4} \right) \frac{d^2}{k^2} G^2, \quad (9)$$

143 where  $\bar{\mathbf{x}}_T = \frac{1}{S_T} \sum_{t=0}^{T-1} w_t \mathbf{x}_t$ , for  $w_t = (a+t)^2$ , and  $S_T = \sum_{t=0}^{T-1} w_t \geq \frac{1}{3} T^3$ .

**Remark 2.5** (Choice of the shift  $a$ ). *Theorem 2.4 says that for any shift  $a > 1$  there is a parameter  $\alpha(a) > 4$  such that (9) holds. However, for the choice  $a = O(1)$  one has to set  $\alpha$  such that  $\frac{\alpha}{\alpha-4} = \Omega(\frac{d}{k})$  and the last term in (9) will be of order  $O(\frac{d^3}{k^3 T^2})$ , thus requiring  $T = \Omega(\frac{d^{1.5}}{k^{1.5}})$  steps to yield convergence. For  $\alpha \geq 5$  we have  $\frac{\alpha}{\alpha-4} = O(1)$  and the last term is only of order  $O(\frac{d^2}{k^2 T^2})$  instead. However, this requires typically a large shift. Observe  $\frac{(\alpha+1)\frac{d}{k} + \rho}{\rho+1} \leq 1 + (\alpha+1)\frac{d}{k} \leq (\alpha+2)\frac{d}{k}$ , i.e. setting  $a = (\alpha+2)\frac{d}{k}$  is enough. We like to stress that in general it is not advisable to set  $a \gg (\alpha+2)\frac{d}{k}$  as the first two terms in (9) depend on  $a$ . In practice, it often suffices to set  $a = \frac{d}{k}$ , as we will discuss in Section 4.*

**Remark 2.6.** *As discussed in Remark 2.5 above, setting  $\alpha = 5$  and  $a = (\alpha+2)\frac{d}{k}$  is feasible. With this choice, equation (9) simplifies to*

$$\mathbb{E} f(\bar{\mathbf{x}}_T) - f^* \leq \mathcal{O}\left(\frac{G^2}{\mu T}\right) + \mathcal{O}\left(\frac{\frac{d^2}{k^2} G^2 \kappa}{\mu T^2}\right) + \mathcal{O}\left(\frac{\frac{d^3}{k^3} G^2}{\mu T^3}\right), \quad (10)$$

for  $\kappa = \frac{L}{\mu}$ . To estimate the second term in (9) we used the property  $\mathbb{E} \mu \|\mathbf{x}_0 - \mathbf{x}^*\| \leq 2G$  for  $\mu$ -strongly convex  $f$ , as derived in [24, Lemma 2]. We observe that for  $T = \Omega(\frac{d}{k} \kappa^{1/2})$  the first term is dominating, and Algorithm 1 converges at rate  $O(\frac{G^2}{\mu T})$ , the same rate as vanilla SGD [13].

### 3 Proof Outline

We now give the outline of the proof. The proofs of the lemmas are given in Appendix A.2.

**Perturbed iterate analysis.** Inspired by the perturbed iterate framework in [18] and [14] we first define a virtual sequence  $\{\tilde{\mathbf{x}}_t\}_{t \geq 0}$  in the following way:

$$\tilde{\mathbf{x}}_0 = \mathbf{x}_0, \quad \tilde{\mathbf{x}}_{t+1} = \tilde{\mathbf{x}}_t - \eta_t \nabla f_{i_t}(\mathbf{x}_t), \quad (11)$$

where the sequences  $\{\mathbf{x}_t\}_{t \geq 0}$ ,  $\{\eta_t\}_{t \geq 0}$  and  $\{i_t\}_{t \geq 0}$  are the same as in (8). Notice that

$$\tilde{\mathbf{x}}_t - \mathbf{x}_t = \left(\mathbf{x}_0 - \sum_{j=0}^{t-1} \eta_j \nabla f_{i_j}(\mathbf{x}_j)\right) - \left(\mathbf{x}_0 - \sum_{j=0}^{t-1} \mathbf{g}_j\right) = \mathbf{m}_t. \quad (12)$$

**Lemma 3.1.** *Let  $\{\mathbf{x}_t\}_{t \geq 0}$  and  $\{\tilde{\mathbf{x}}_t\}_{t \geq 0}$  be defined as in (8) and (11) and let  $f_i$  be  $L$ -smooth and  $f$  be  $\mu$ -strongly convex with  $\mathbb{E}_i \|\nabla f_i(\mathbf{x}_t)\|^2 \leq G^2$ . Then*

$$\mathbb{E} \|\tilde{\mathbf{x}}_{t+1} - \mathbf{x}^*\|^2 \leq \left(1 - \frac{\eta_t \mu}{2}\right) \mathbb{E} \|\tilde{\mathbf{x}}_t - \mathbf{x}^*\|^2 + \eta_t^2 G^2 - \eta_t e_t + \eta_t (\mu + 2L) \mathbb{E} \|\mathbf{m}_t\|^2, \quad (13)$$

where  $e_t := \mathbb{E} f(\mathbf{x}_t) - f^*$ .

**Bounding the memory.** From equation (13) it becomes clear that we should derive an upper bound on  $\mathbb{E} \|\mathbf{m}_t\|^2$ . For this we will use the contraction property (4) of the sparsity operators.

**Lemma 3.2.** *Let  $\{\mathbf{x}_t\}_{t \geq 0}$  as defined in (8) for  $0 < k \leq d$ ,  $\mathbb{E}_i \|\nabla f_i(\mathbf{x}_t)\|^2 \leq G^2$  and stepsizes  $\eta_t = \frac{8}{\mu(a+t)}$  with  $a, \alpha > 4$ , as in Theorem 2.4. Then*

$$\mathbb{E} \|\mathbf{m}_t\|^2 \leq \eta_t^2 \frac{4\alpha}{\alpha-4} \frac{d^2}{k^2} G^2. \quad (14)$$

**Optimal averaging.** Similar as discussed in [13, 24, 31] we have to define a suitable averaging scheme for the iterates  $\{\mathbf{x}_t\}_{t \geq 0}$  to get the optimal convergence rate. In contrast to [13] that use linearly increasing weights, we use quadratically increasing weights, as for instance [31].

**Lemma 3.3.** *Let  $\{a_t\}_{t \geq 0}$ ,  $a_t \geq 0$ ,  $\{e_t\}_{t \geq 0}$ ,  $e_t \geq 0$ , be sequences satisfying*

$$a_{t+1} \leq \left(1 - \frac{\mu \eta_t}{2}\right) a_t + \eta_t^2 A + \eta_t^3 B - \eta_t e_t, \quad (15)$$

for  $\eta_t = \frac{8}{\mu(a+t)}$  and constants  $A, B \geq 0$ ,  $\mu > 0$ ,  $a > 1$ . Then

$$\frac{1}{S_T} \sum_{t=0}^{T-1} w_t e_t \leq \frac{\mu a^3}{8 S_T} a_0 + \frac{4T(T+2a)}{\mu S_T} A + \frac{64T}{\mu^2 S_T} B, \quad (16)$$

for  $w_t = (a+t)^2$  and  $S_T := \sum_{t=0}^{T-1} w_t = \frac{T}{6} (2T^2 + 6aT - 3T + 6a^2 - 6a + 1) \geq \frac{1}{3} T^3$ .

175 **Proof of Theorem 2.4.** The proof of the theorem immediately follows from the three lemmas that  
 176 we have presented in this section and convexity of  $f$ , i.e. we have  $\mathbb{E} f(\bar{\mathbf{x}}_T) - f^* \leq \frac{1}{S_T} \sum_{t=0}^{T-1} w_t e_t$   
 177 in (16), for constants  $A = G^2$  and  $B = (\mu + 2L)_{\alpha=4} \frac{d^2}{k^2} G^2$ .  $\square$

## 178 4 Experiments

179 We present numerical experiments to illustrate the excellent convergence properties and commu-  
 180 nication efficiency of MEM-SGD. As the usefulness of the scheme has already been proven in  
 181 practical applications[1, 6, 17, 30, 33] we focus here on a few particular aspects. First, we verify  
 182 the impact of the initial learning rate that did drop up in the statement of Theorem 2.4. We then  
 183 compare our method with QSGD [3] which decreases the communication cost in SGD by using  
 184 random quantization operators, but without memory. Finally, we show the performance of the parallel  
 185 local SGD depicted in Algorithm 2 in a multicore setting and compare the speed-up to asynchronous  
 186 Hogwild! [21].

### 187 4.1 Experimental Setup

188 **Models.** Our experiments focus on the performance of MEM-SGD applied to logistic regression.  
 189 The associated objective function is  $\frac{1}{n} \sum_{i=1}^n \log(1 + \exp(-b_i \mathbf{a}_i^\top \mathbf{x})) + \frac{\lambda}{2} \|\mathbf{x}\|^2$ , where  $\mathbf{a}_i \in \mathbb{R}^d$  and  
 190  $b_i \in \{-1, +1\}$  are the data samples, and we employ a standard  $L_2$ -regularizer. The regularization  
 191 parameter is set to  $\lambda = 1/n$  for both datasets following [29].

192 **Datasets.** We consider a dense dataset, *epsilon* [32], as well as a sparse dataset, *RCV1* [16] where  
 we train on the larger test set. Statistics on the datasets are listed in Table 1 below:

	$n$	$d$	density
<b>epsilon</b>	400'000	2'000	100%
<b>RCV1-test</b>	677'399	47'236	0.15%

193 Table 1: Datasets statistics.

	parameter	value
<b>epsilon</b>	$\gamma$	2
	$a$	$d/k$
<b>RCV1-test</b>	$\gamma$	2
	$a$	$10d/k$

Table 2: Learning rate  $\eta_t = \gamma/(\lambda(t + a))$ .

194 **Implementation.** Our experiments are run using Python3 and the numpy library. Code will be  
 195 released with the publication for reproducibility. We emphasize that our high level implementation is  
 196 not optimized for performance but for readability and simplicity. We only report convergence per  
 197 iteration and relative speedups, but not wall-clock time because unequal efforts have been made to  
 198 speed up the different implementations. Plots additionally show the baseline computed with the  
 199 standard optimizer `LogisticSGD` of `scikit-learn` [22]. Experiments were run on an Ubuntu 16.04  
 200 machine with a 24 cores processor Intel® Xeon® CPU E5-2680 v3 @ 2.50GHz. The kernel is Linux  
 201 4.4.0-116.

JB: is it a better disclaimer?

### 202 4.2 Verifying the Theory

203 We study the convergence of the method using the stepsizes  $\eta_t = \gamma/(\lambda(t + a))$  and hyperparameters  
 204  $\gamma$  and  $a$  set as in Table 2. We compute the final estimate  $\bar{\mathbf{x}}$  as a weighted average of all iterates  $\mathbf{x}_t$   
 205 with weights  $w_t = (t + a)^2$  as indicated by Theorem 2.4. The results are depicted in Figure 2. We  
 206 use  $k \in \{1, 2, 3\}$  for *epsilon* and  $k \in \{10, 20, 30\}$  for *RCV1* due to the large number of features.  
 207 The  $\text{top}_k$  variant consistently outperforms  $\text{rand}_k$ . We also evaluate the impact of the delay  $a$  in the  
 208 learning rate: setting it to 1 instead of order  $\mathcal{O}(d/k)$  dramatically hurts the memory and requires time  
 209 to recover from the high initial learning rate (labeled “without delay” on the plot).

210 We experimentally verified the convergence properties of MEM-SGD for different sparsification  
 211 operators and stepsizes but we want to further evaluate its fundamental benefits in terms of sparsity  
 212 enforcement and reduction of the communication bottleneck. The gain in communication cost  
 213 of SGD with memory is very high for dense datasets—using the  $\text{top}_1$  strategy on *epsilon* dataset  
 214 improves the amount of communication by  $10^3$  compared to SGD. For the sparse dataset, SGD can  
 215 readily use the given sparsity of the gradients. The effective dimension of the gradients on *RCV1*

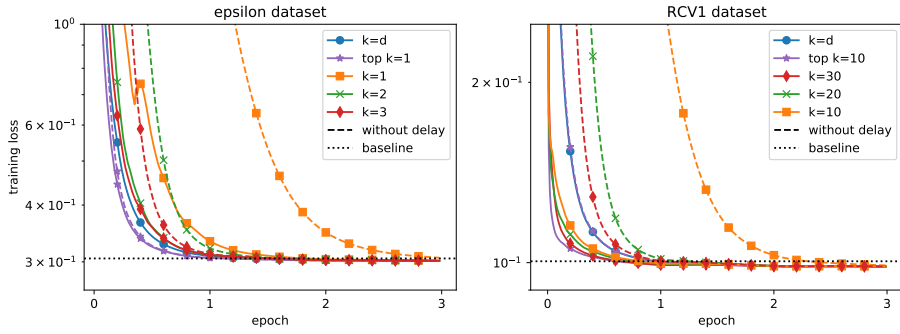


Figure 2: Convergence of MEM-SGD using different sparsification operators compared to full SGD with theoretical learning rates (parameters in Table 2).

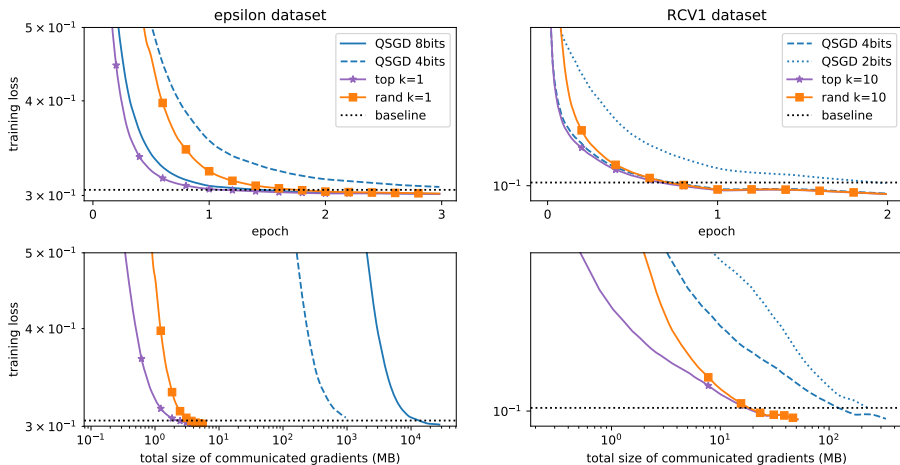


Figure 3: MEM-SGD and QSGD convergence comparison. *Top row*: convergence in number of iterations. *Bottom row*: cumulated size of the communicated gradients during training.

is around  $47'236 \times 0.15\% \approx 71$ , which let SGD use sparse gradients and in expectation share only 142 floating-point numbers per update. Nevertheless, the improvement for  $\text{top}_{10}$  on *RCV1* is of approximately an order of magnitude.

### 4.3 Comparison with QSGD

Now we compare MEM-SGD with the QSGD compression scheme [3] which reduces communication cost by random quantization. The accuracy (and the compression ratio) in QSGD is controlled by a parameter  $s$ , corresponding to the number of quantization levels. Ideally, we would like to set the quantization precision in QSGD such that the number of bits transmitted by QSGD and MEM-SGD are identical. However, even for the lowest precision, QSGD needs to send the sign and index of  $\sqrt{d}$  coordinates. It is therefore not possible to reach the compression level of sparsification operators that only transmit a constant number of bits per iteration. Hence, we did not enforce this condition and resorted to pick reasonable levels of quantization in QSGD ( $s = 2^b$  with  $b = 2, 4, 8$ ). Figure 3 shows that MEM-SGD with  $\text{top}_1$  and  $\text{top}_{10}$  on *epsilon* and *RCV1* converges as fast as QSGD in term of iterations for 8 and 4-bits respectively. Note that  $b$ -bits stands for the number of bits used to encode  $s = 2^b$  levels but the actual number of bits transmitted in QSGD can be reduced using Elias coding. According to [3, Theorem 3.2] QSGD with 8-bits levels needs to share  $10^5$  bits per update on *epsilon*, and 4-bits levels on *RCV1* needs  $10^3$  bits per update. As shown in the bottom of Figure 3, we are transmitting two orders of magnitude fewer bits with the  $\text{top}_1$  sparsifier for *epsilon* and one order of magnitude for *RCV1*, concluding that sparsification offers a much more aggressive and performant strategy than quantization.

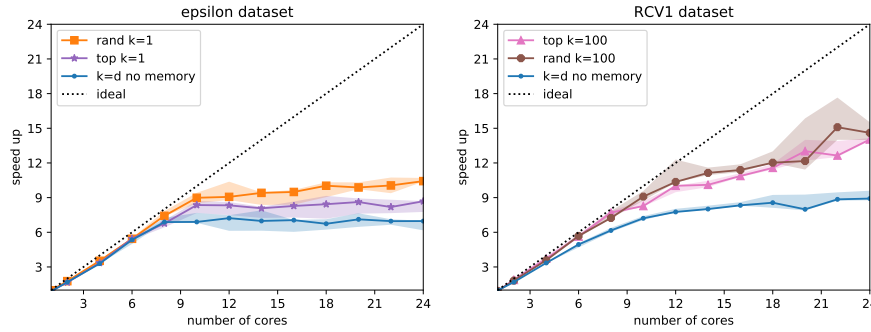


Figure 4: Multicore CPU time speed up comparison between MEM-SGD and lock-free SGD.

#### 4.4 Multicore experiment

We implement a parallelized version of MEM-SGD, as depicted in Algorithm 2. The enforced sparsity allows us to do the update in shared memory using a lock-free mechanism as in [21]. For this experiment we evaluate the final iterate  $\mathbf{x}_T$  instead of the weighted average  $\bar{\mathbf{x}}_T$  above, and we also investigated constant learning rate, which turns out to work well in practice for *epsilon*. We used constant  $\eta_t \equiv 0.05$  for *epsilon* and reused the parameters from Table 2 for *RCV1*.

Figure 4 shows the speed-up obtained when increasing the number of cores processing the dataset. We see that the speed-up is almost linear up to 10 cores. This is especially remarkable for the dense *epsilon* dataset, as the dense gradient updates would usually imply many update conflicts for classic SGD and Hogwild! [21] (i.e. hurting convergence because of gradients computed on stale iterates) or require the use of locking (i.e. hurting speed). We did not use atomic updates of the parameter in the shared memory, allowing some workers to overwrite the progress of others which might contribute to the slowdown for higher number of workers. The experiment is run on a single machine, hence no inter-node communication is used. The colored area depicts the best and worst results of 3 independent runs for each dataset.

In this asynchronous setting, SGD with memory computes gradients on stale iterates that differs only by a few coordinates. It encounters fewer inconsistent read/write operation than lock free asynchronous SGD and exhibit better scaling properties. The  $\text{top}_k$  operator performs better than  $\text{rand}_k$  in the sequential setup, but this is not the case in the parallel setup. A reason for this could be that due to the deterministic nature of the  $\text{top}_k$  operator the cores are more prone to update the same set of coordinates, and more collisions appear.

M: MJ: not completely ideal/precise here yet; S: agree, do we need to talk about collisions? JB: moved here and removed collision

S: need to add in this paragraph a sentence about  $k = d$  is similar to Hogwild

## 5 Conclusion

We studied the convergence properties of heavily sparsified SGD using memory, a variant of SGD that reduces drastically the size of the gradients, overcoming the communication bottleneck, while keeping the SGD convergence rate. This new method enforces sparse updates which opens the way to applying lock free asynchronous methods (i.e. Hogwild [21]) to a wide variety of dense problems, e.g. neural nets, logistic regression on dense datasets.

It has been shown in practice that our approach can be efficiently applied to bandwidth memory limited systems such as multi GPU training. Our proof gives a novel theoretical explanation for the sequential setup but does not yet encompass the asynchronous parallel setting.

Hogwild with mini-batches: [28], we could add a comment (here or main text), that  $k < 1$  is like close to mini-batch (I think it would fit below sparsification remark)

Try to add:

- short summary, We propose .... useful because, ... many applications.
- we don't analyze distributed setting (yet)
- ?

We can keep it short, but needs to sell again the main points, where could this have impact, etc.?

273 Multi GPU training.

## 274 References

- 275 [1] Alham Fikri Aji and Kenneth Heafield. Sparse communication for distributed gradient descent. In  
276 *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages  
277 440–445. Association for Computational Linguistics, 2017.
- 278 [2] Dan Alistarh, Christopher De Sa, and Nikola Konstantinov. The Convergence of Stochastic Gradient  
279 Descent in Asynchronous Shared Memory. *arXiv*, March 2018.
- 280 [3] Dan Alistarh, Demjan Grubic, Jerry Li, Ryota Tomioka, and Milan Vojnovic. QSGD: Communication-  
281 efficient SGD via gradient quantization and encoding. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach,  
282 R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*  
283 *30*, pages 1709–1720. Curran Associates, Inc., 2017.
- 284 [4] Léon Bottou. Large-scale machine learning with stochastic gradient descent. In Yves Lechevallier and  
285 Gilbert Saporta, editors, *Proceedings of COMPSTAT’2010*, pages 177–186, Heidelberg, 2010. Physica-  
286 Verlag HD.
- 287 [5] Chia-Yu Chen, Jungwook Choi, Daniel Brand, Ankur Agrawal, Wei Zhang, and Kailash Gopalakrishnan.  
288 Adacomp : Adaptive residual gradient compression for data-parallel distributed training, 2018.
- 289 [6] N. Dryden, T. Moon, S. A. Jacobs, and B. V. Essen. Communication quantization for data-parallel training  
290 of deep neural networks. In *2016 2nd Workshop on Machine Learning in HPC Environments (MLHPC)*,  
291 pages 1–8, Nov 2016.
- 292 [7] John Duchi, Elad Hazan, and Yoram Singer. Adaptive Subgradient Methods for Online Learning and  
293 Stochastic Optimization. *JMLR*, 12:2121–2159, August 2011.
- 294 [8] Celestine Dünnér, Thomas Parnell, and Martin Jaggi. Efficient use of limited-memory accelerators for  
295 linear learning on heterogeneous systems. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus,  
296 S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages  
297 4258–4267. Curran Associates, Inc., 2017.
- 298 [9] Priya Goyal, Piotr Dollár, Ross B. Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew  
299 Tulloch, Yangqing Jia, and Kaiming He. Accurate, large minibatch SGD: training imagenet in 1 hour.  
300 *CoRR*, abs/1706.02677, 2017.
- 301 [10] Suyog Gupta, Ankur Agrawal, Kailash Gopalakrishnan, and Pritish Narayanan. Deep learning with limited  
302 numerical precision. In *Proceedings of the 32Nd International Conference on International Conference on*  
303 *Machine Learning - Volume 37*, ICML’15, pages 1737–1746. JMLR.org, 2015.
- 304 [11] Cho-Jui Hsieh, Hsiang-Fu Yu, and Inderjit Dhillon. Passcode: Parallel asynchronous stochastic dual  
305 co-ordinate descent. In *International Conference on Machine Learning*, pages 2370–2379, 2015.
- 306 [12] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980,  
307 2014.
- 308 [13] Simon Lacoste-Julien, Mark W. Schmidt, and Francis R. Bach. A simpler approach to obtaining an  $o(1/t)$   
309 convergence rate for the projected stochastic subgradient method. *CoRR*, abs/1212.2002, 2012.
- 310 [14] Rémi Leblond, Fabian Pedregosa, and Simon Lacoste-Julien. ASAGA: Asynchronous parallel SAGA.  
311 In Aarti Singh and Jerry Zhu, editors, *Proceedings of the 20th International Conference on Artificial*  
312 *Intelligence and Statistics*, volume 54 of *Proceedings of Machine Learning Research*, pages 46–54, Fort  
313 Lauderdale, FL, USA, 20–22 Apr 2017. PMLR.
- 314 [15] Rémi Leblond, Fabian Pedregosa, and Simon Lacoste-Julien. Improved asynchronous parallel optimization  
315 analysis for stochastic incremental methods. *arXiv.org*, January 2018.
- 316 [16] David D. Lewis, Yiming Yang, Tony G. Rose, and Fan Li. RCV1: A new benchmark collection for text  
317 categorization research. *Journal of Machine Learning Research*, 5:361–397, 2004.
- 318 [17] Yujun Lin, Song Han, Huizi Mao, Yu Wang, and Bill Dally. Deep gradient compression: Reducing the  
319 communication bandwidth for distributed training. In *ICLR 2018 - International Conference on Learning*  
320 *Representations*, 2018.

- [18] Horia Mania, Xinghao Pan, Dimitris Papailiopoulos, Benjamin Recht, Kannan Ramchandran, and Michael I. Jordan. Perturbed iterate analysis for asynchronous stochastic optimization. *SIAM Journal on Optimization*, 27(4):2202–2229, 2017.
- [19] Eric Moulines and Francis R. Bach. Non-asymptotic analysis of stochastic approximation algorithms for machine learning. In J. Shawe-Taylor, R. S. Zemel, P. L. Bartlett, F. Pereira, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 24*, pages 451–459. Curran Associates, Inc., 2011.
- [20] T. Na, J. H. Ko, J. Kung, and S. Mukhopadhyay. On-chip training of recurrent neural networks with limited numerical precision. In *2017 International Joint Conference on Neural Networks (IJCNN)*, pages 3716–3723, May 2017.
- [21] Feng Niu, Benjamin Recht, Christopher Re, and Stephen J. Wright. Hogwild!: A lock-free approach to parallelizing stochastic gradient descent. In *Proceedings of the 24th International Conference on Neural Information Processing Systems, NIPS’11*, pages 693–701, USA, 2011. Curran Associates Inc.
- [22] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.
- [23] B. T. Polyak and A. B. Juditsky. Acceleration of stochastic approximation by averaging. *SIAM Journal on Control and Optimization*, 30(4):838–855, 1992.
- [24] Alexander Rakhlin, Ohad Shamir, and Karthik Sridharan. Making gradient descent optimal for strongly convex stochastic optimization. In *Proceedings of the 29th International Conference on Machine Learning, ICML’12*, pages 1571–1578, USA, 2012. Omnipress.
- [25] Herbert Robbins and Sutton Monro. A Stochastic Approximation Method. *The Annals of Mathematical Statistics*, 22(3):400–407, September 1951.
- [26] David Ruppert. Efficient estimations from a slowly convergent robbins-monro process. Technical report, Cornell University Operations Research and Industrial Engineering, 1988.
- [27] Christopher De Sa, Ce Zhang, Kunle Olukotun, and Christopher Ré. Taming the wild: A unified analysis of hog wild! -style algorithms. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 2, NIPS’15*, pages 2674–2682, Cambridge, MA, USA, 2015. MIT Press.
- [28] S. Sallinen, N. Satish, M. Smelyanskiy, S. S. Sury, and C. Ré. High performance parallel stochastic gradient descent in shared memory. In *2016 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, pages 873–882, May 2016.
- [29] Mark Schmidt, Nicolas Le Roux, and Francis Bach. Minimizing finite sums with the stochastic average gradient. *Math. Program.*, 162(1-2):83–112, March 2017.
- [30] 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 Haizhou Li, Helen M. Meng, Bin Ma, Engsiong Chng, and Lei Xie, editors, *INTERSPEECH*, pages 1058–1062. ISCA, 2014.
- [31] Ohad Shamir and Tong Zhang. Stochastic gradient descent for non-smooth optimization: Convergence results and optimal averaging schemes. In Sanjoy Dasgupta and David McAllester, editors, *Proceedings of the 30th International Conference on Machine Learning*, volume 28 of *Proceedings of Machine Learning Research*, pages 71–79, Atlanta, Georgia, USA, 17–19 Jun 2013. PMLR.
- [32] Soren Sonnenburg, Vojtech Franc, E Yom-Tov, and M Sebag. Pascal large scale learning challenge. 10:1937–1953, 01 2008.
- [33] Nikko Strom. Scalable distributed dnn training using commodity gpu cloud computing. In *INTERSPEECH*, pages 1488–1492. ISCA, 2015.
- [34] Xu Sun, Xuancheng Ren, Shuming Ma, and Houfeng Wang. meProp: Sparsified back propagation for accelerated deep learning with reduced overfitting. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 3299–3308, International Convention Centre, Sydney, Australia, 06–11 Aug 2017. PMLR.
- [35] Jianqiao Wangni, Jiale Wang, Ji Liu, and Tong Zhang. Gradient sparsification for communication-efficient distributed optimization. *CoRR*, abs/1710.09854, 2017.

- 371 [36] Wei Wen, Cong Xu, Feng Yan, Chunpeng Wu, Yandan Wang, Yiran Chen, and Hai Li. Terngrad:  
372 Ternary gradients to reduce communication in distributed deep learning. In I. Guyon, U. V. Luxburg,  
373 S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information*  
374 *Processing Systems 30*, pages 1509–1519. Curran Associates, Inc., 2017.
- 375 [37] Yang You, Igor Gitman, and Boris Ginsburg. Scaling sgd batch size to 32k for imagenet training. *CoRR*,  
376 abs/1708.03888, 2017.
- 377 [38] Hantian Zhang, Jerry Li, Kaan Kara, Dan Alistarh, Ji Liu, and Ce Zhang. ZipML: Training linear models  
378 with end-to-end low precision, and a little bit of deep learning. In Doina Precup and Yee Whye Teh, editors,  
379 *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of*  
380 *Machine Learning Research*, pages 4035–4043, International Convention Centre, Sydney, Australia, 06–11  
381 Aug 2017. PMLR.
- 382 [39] Yuchen Zhang, Martin J Wainwright, and John C Duchi. Communication-efficient algorithms for statistical  
383 optimization. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural*  
384 *Information Processing Systems 25*, pages 1502–1510. Curran Associates, Inc., 2012.
- 385 [40] Peilin Zhao and Tong Zhang. Stochastic optimization with importance sampling for regularized loss  
386 minimization. In Francis Bach and David Blei, editors, *Proceedings of the 32nd International Conference*  
387 *on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 1–9, Lille, France,  
388 07–09 Jul 2015. PMLR.

# Appendix

## A Proofs

### A.1 Useful facts

**Lemma A.1.** For  $\mathbf{x} \in \mathbb{R}^d$ ,  $1 \leq k \leq d$ , and operator  $\text{sparse}_k \in \{\text{top}_k, \text{rand}_k\}$  it holds

$$\mathbb{E} \|\text{sparse}_k(\mathbf{x}) - \mathbf{x}\|^2 \leq \left(1 - \frac{k}{d}\right) \|\mathbf{x}\|^2. \quad (17)$$

*Proof.* From the definition of the operators, for all  $\mathbf{x}$  in  $\mathbb{R}^d$  we have

$$\|\mathbf{x} - \text{top}_k(\mathbf{x})\|^2 \leq \|\mathbf{x} - \text{rand}_k(\mathbf{x})\|^2 \quad (18)$$

and we apply the expectation

$$\mathbb{E}_\omega \|\mathbf{x} - \text{rand}_k(\mathbf{x})\|^2 = \frac{1}{|\Omega_k|} \sum_{\omega \in \Omega_k} \sum_{i=1}^d \mathbf{x}_i^2 \mathbb{I}\{i \notin \omega\} = \sum_{i=1}^d x_i^2 \sum_{\omega \in \Omega_k} \frac{\mathbb{I}\{i \notin \omega\}}{|\Omega_k|} = \left(1 - \frac{k}{d}\right) \|\mathbf{x}\|^2 \quad (19)$$

which concludes the proof.  $\square$

**Lemma A.2.** Let  $\eta_t = \frac{1}{c+t}$ , for  $c \geq 1$ . Then  $\eta_t^2 \left(1 - \frac{2}{c}\right) \leq \eta_{t+1}^2$ .

*Proof.* Observe

$$\eta_t^2 \left(1 - \frac{2}{c}\right) = \frac{c-2}{c(c+t)^2} \leq \frac{c-2}{(c+t+1)^2(c-2)} = \eta_{t+1}^2. \quad (20)$$

where the inequality follows from

$$(c+t+1)^2(c-2) = c(c+t)^2 + \underbrace{(c-2)(1+2(t+c)) - 2(c+t)^2}_{=-2t^2-2ct-4t-3c-2 \leq 0} \quad (21)$$

$\square$

### A.2 Proof of the Main Theorem

*Proof of Lemma 3.1.* Using the update equation (11) we have

$$\|\tilde{\mathbf{x}}_{t+1} - \mathbf{x}^*\|^2 = \|\tilde{\mathbf{x}}_t - \mathbf{x}^*\|^2 + \eta_t^2 \|\nabla f_{i_t}(\mathbf{x}_t)\|^2 - 2\eta_t \langle \mathbf{x}_t - \mathbf{x}^*, \nabla f_{i_t}(\mathbf{x}_t) \rangle + 2\eta_t \langle \mathbf{x}_t - \tilde{\mathbf{x}}_t, \nabla f_{i_t}(\mathbf{x}_t) \rangle. \quad (22)$$

And by applying expectation

$$\mathbb{E}_{i_t} \|\tilde{\mathbf{x}}_{t+1} - \mathbf{x}^*\|^2 \leq \|\tilde{\mathbf{x}}_t - \mathbf{x}^*\|^2 + \eta_t^2 G^2 - 2\eta_t \langle \mathbf{x}_t - \mathbf{x}^*, \nabla f(\mathbf{x}_t) \rangle + 2\eta_t \langle \mathbf{x}_t - \tilde{\mathbf{x}}_t, \nabla f(\mathbf{x}_t) \rangle. \quad (23)$$

To upper bound the third term, we use the same estimates as in [14, Appendix C.3]: By strong convexity,  $f(\mathbf{y}) \geq f(\mathbf{x}) + \langle \nabla f(\mathbf{x}), \mathbf{y} - \mathbf{x} \rangle + \frac{\mu}{2} \|\mathbf{y} - \mathbf{x}\|^2$  for  $\mathbf{x}, \mathbf{y} \in \mathbb{R}^d$ , hence

$$-\langle \mathbf{x}_t - \mathbf{x}^*, \nabla f(\mathbf{x}_t) \rangle \leq -(f(\mathbf{x}_t) - f^*) - \frac{\mu}{2} \|\mathbf{x}_t - \mathbf{x}^*\|^2 \quad (24)$$

and with  $\|\mathbf{a} + \mathbf{b}\|^2 \leq 2\|\mathbf{a}\|^2 + 2\|\mathbf{b}\|^2$  we further have

$$-\|\mathbf{x}_t - \mathbf{x}^*\|^2 \leq \|\mathbf{x}_t - \tilde{\mathbf{x}}_t\|^2 - \frac{1}{2} \|\tilde{\mathbf{x}}_t - \mathbf{x}^*\|^2. \quad (25)$$

Putting these two estimates together, we can bound (23) as follows:

$$\mathbb{E}_{i_t} \|\tilde{\mathbf{x}}_{t+1} - \mathbf{x}^*\|^2 \leq \left(1 - \frac{\eta_t \mu}{2}\right) \|\tilde{\mathbf{x}}_t - \mathbf{x}^*\|^2 + \eta_t^2 G^2 - 2\eta_t e_t + \eta_t \mu \|\mathbf{x}_t - \tilde{\mathbf{x}}_t\|^2 + 2\eta_t \langle \mathbf{x}_t - \tilde{\mathbf{x}}_t, \nabla f(\mathbf{x}_t) \rangle, \quad (26)$$

407 where  $e_t = \mathbb{E} f(\mathbf{x}_t) - f^*$ . We now estimate the last term. As each  $f_i$  is  $L$ -smooth also  $f$  is  $L$ -smooth, i.e. satisfies  
 408  $f(\mathbf{x}) - f(\mathbf{y}) - \langle \nabla f(\mathbf{y}), \mathbf{x} - \mathbf{y} \rangle \geq \frac{1}{2L} \|\nabla f(\mathbf{y}) - \nabla f(\mathbf{x})\|^2$ . Together with  $2 \langle a, b \rangle \leq \gamma \|a\|^2 + \gamma^{-1} \|b\|^2$  we  
 409 have

$$\langle \mathbf{x}_t - \tilde{\mathbf{x}}_t, \nabla f(\mathbf{x}_t) \rangle \leq \frac{1}{2} \left( 2L \|\mathbf{x}_t - \tilde{\mathbf{x}}_t\|^2 + \frac{1}{2L} \|\nabla f(\mathbf{x}_t)\|^2 \right) \quad (27)$$

$$= L \|\mathbf{x}_t - \tilde{\mathbf{x}}_t\|^2 + \frac{1}{4L} \|\nabla f(\mathbf{x}_t) - \nabla f(\mathbf{x}^*)\|^2 \quad (28)$$

$$\leq L \|\mathbf{x}_t - \tilde{\mathbf{x}}_t\|^2 + \frac{1}{2} (f(\mathbf{x}_t) - f^*) . \quad (29)$$

410 Combining with (26) we have

$$\mathbb{E}_{i_t} \|\tilde{\mathbf{x}}_{t+1} - \mathbf{x}^*\|^2 \leq \left( 1 - \frac{\eta_t \mu}{2} \right) \|\tilde{\mathbf{x}}_t - \mathbf{x}^*\|^2 + \eta_t^2 G^2 - \eta_t e_t + \eta_t (\mu + 2L) \|\mathbf{x}_t - \tilde{\mathbf{x}}_t\|^2 , \quad (30)$$

411 and the claim follows with (12).  $\square$

412 *Proof of Lemma 3.2.* First, observe that by Lemma A.1 and  $\|\mathbf{a} + \mathbf{b}\|^2 \leq (1 + \gamma) \|\mathbf{a}\|^2 + (1 + \gamma^{-1}) \|\mathbf{b}\|^2$  for  
 413  $\gamma > 0$  we have

$$\mathbb{E} \|\mathbf{m}_{t+1}\|^2 \leq \left( 1 - \frac{k}{d} \right) \|\mathbf{m}_t + \eta_t \nabla f_{i_t}(\mathbf{x}_t)\|^2 \quad (31)$$

$$\leq \left( 1 - \frac{k}{d} \right) \left( \left( 1 + \frac{k}{2d} \right) \mathbb{E} \|\mathbf{m}_t\|^2 + \left( 1 + \frac{2d}{k} \right) \eta_t^2 \mathbb{E} \|\nabla f_{i_t}(\mathbf{x}_t)\|^2 \right) \quad (32)$$

$$\leq \left( 1 - \frac{k}{2d} \right) \mathbb{E} \|\mathbf{m}_t\|^2 + \frac{2d}{k} \eta_t^2 G^2 . \quad (33)$$

414 On the other hand, from  $\|\sum_{i=1}^s \mathbf{a}_i\|^2 \leq s \sum_{i=1}^s \|\mathbf{a}_i\|^2$  we also have

$$\mathbb{E} \|\mathbf{m}_{t+1}\|^2 \leq (t+1) \sum_{i=0}^t \eta_i^2 G^2 . \quad (34)$$

415 Now the claim follows from Lemma A.3 just below with  $A = \frac{8G^2}{\mu}$ .  $\square$

416 **Lemma A.3.** Let  $A \geq 0$ ,  $d \geq k \geq 1$ ,  $\{h_t\}_{t \geq 0}$ ,  $h_t \geq 0$  be a sequence satisfying

$$h_0 = 0 , \quad h_{t+1} \leq \min \left\{ \left( 1 - \frac{k}{2d} \right) h_t + \frac{2d}{k} \eta_t^2 A, (t+1) \sum_{i=0}^t \eta_i^2 A \right\} , \quad (35)$$

417 for a sequence  $\eta_t = \frac{1}{a+t}$  with  $a \geq \frac{(\alpha+1)\frac{d}{k} + \rho + 1}{\rho + 1} > 1$ , for  $\alpha > 4$ ,  $\rho := \frac{4\alpha}{(\alpha-4)(\alpha+1)^2}$ . Then

$$h_t \leq \frac{4\alpha}{\alpha-4} \eta_t^2 \frac{d^2}{k^2} A , \quad (36)$$

418 for  $t \geq 0$ .

419 *Proof.* The claim holds for  $t = 0$ .

420 **Large  $t$ .** Let  $t_0 = \max\{\lceil \alpha \frac{d}{k} - a \rceil, 0\}$ , i.e.  $\eta_{t_0} \leq \frac{k}{\alpha d}$ . (Note that for any  $a \geq \alpha \frac{k}{d}$  it holds  $t_0 = 0$ .) Suppose  
 421 the claim holds for  $t \leq t_0$ . Observe,

$$\eta_t^2 \left( 1 - \frac{2k}{\alpha d} \right) \leq \eta_{t+1}^2 , \quad (37)$$

422 for  $t \geq t_0$ . This follows from Lemma A.2 with  $c = \frac{\alpha d}{k}$ . By induction,

$$h_{t+1} \leq \left( 1 - \frac{k}{2d} \right) \frac{4\alpha}{\alpha-4} \eta_t^2 \frac{d^2}{k^2} A + \frac{2d}{k} \eta_t^2 A \quad (38)$$

$$= \underbrace{\eta_t^2 \left( 1 - \frac{2k}{\alpha d} \right) \frac{4\alpha}{\alpha-4} \frac{d^2}{k^2} A}_{\leq \eta_{t+1}^2} , \quad (39)$$

423 where we used  $t \geq t_0$  (and the observation just above) for the last inequality.

424 **Small  $t$ .** Assume  $t_0 \geq 1$ , otherwise the claim follows from the part above. We have

$$h_t \leq t \sum_{i=0}^{t-1} \eta_i^2 A \leq \frac{t}{a-1} A, \quad (40)$$

425 where we used

$$\sum_{t=0}^{t-1} \eta_t^2 \leq \sum_{t=0}^{\infty} \frac{1}{(a+t)^2} \leq \int_{a-1}^{\infty} \frac{1}{x^2} dx = \frac{1}{a-1}, \quad (41)$$

426 for  $a > 1$ . For  $t \leq t_0$  we have

$$\eta_t^2 \frac{d^2}{k^2} \geq \eta_{t_0}^2 \frac{d^2}{k^2} = \frac{1}{(a+t_0)^2} \frac{d^2}{k^2} \geq \frac{1}{\left(\frac{\alpha d}{k} + 1\right)^2} \frac{d^2}{k^2} \geq \frac{1}{\left(\frac{(\alpha+1)d}{k}\right)^2} \frac{d^2}{k^2} = \frac{1}{(\alpha+1)^2}, \quad (42)$$

427 using  $\frac{d}{k} \geq 1$ . Observe  $t_0 \leq \alpha \frac{d}{k} - a + 1 \leq (\alpha+1) \frac{d}{k} - a$ . For  $t \leq (\alpha+1) \frac{d}{k} - a$  we have

$$h_t \leq \frac{t}{a-1} A \leq \frac{(\alpha+1) \frac{d}{k} - a}{a-1} A \leq \rho A, \quad (43)$$

428 by the condition on  $a$ . Hence, by combining these observations,

$$h_t \leq \frac{t}{a-1} A \leq \rho A = \frac{4\alpha}{\alpha-4} \frac{1}{(\alpha+1)^2} A \leq \frac{4\alpha}{\alpha-4} \eta_{t_0}^2 \frac{d^2}{k^2} A \leq \frac{4\alpha}{\alpha-4} \eta_t^2 \frac{d^2}{k^2} A, \quad (44)$$

429 and the proof follows.  $\square$

430 *Proof of Lemma 3.3.* Observe

$$\left(1 - \frac{\mu\eta_t}{2}\right) \frac{w_t}{\eta_t} = \left(\frac{a+t-4}{a+t}\right) \frac{\mu(a+t)^3}{8} = \frac{\mu(a+t-4)(a+t)^2}{8} \leq \frac{\mu(a+t-1)^3}{8} = \frac{w_{t-1}}{\eta_{t-1}}, \quad (45)$$

431 where the inequality is due to

$$(a+t-4)(a+t)^2 = (a+t-1)^3 + \underbrace{1-3a-a^2-3t-2at-t^2}_{\leq 0} \leq (a+t-1)^3, \quad (46)$$

432 for  $a \geq 1, t \geq 0$ .

433 We now multiply equation (15) with  $\frac{w_t}{\eta_t}$ , which yields

$$a_{t+1} \frac{w_t}{\eta_t} \leq \underbrace{\left(1 - \frac{\mu\eta_t}{2}\right) \frac{w_t}{\eta_t}}_{\leq \frac{w_{t-1}}{\eta_{t-1}}} a_t + w_t \eta_t A + w_t \eta_t^2 B - w_t e_t. \quad (47)$$

434 and by recursively substituting  $a_t \frac{w_{t-1}}{\eta_{t-1}}$  we get

$$a_T \frac{w_{T-1}}{\eta_{T-1}} \leq \left(1 - \frac{\mu\eta_0}{2}\right) \frac{w_0}{\eta_0} a_0 + \sum_{t=0}^{T-1} w_t \eta_t A + \sum_{t=0}^{T-1} w_t \eta_t^2 B - \sum_{t=0}^{T-1} w_t e_t, \quad (48)$$

435 i.e.

$$\sum_{t=0}^{T-1} w_t e_t \leq \frac{w_0}{\eta_0} a_0 + \sum_{t=0}^{T-1} w_t \eta_t A + \sum_{t=0}^{T-1} w_t \eta_t^2 B. \quad (49)$$

436 We will now derive upper bounds for the terms on the right hand side. We have

$$\frac{w_0}{\eta_0} = \frac{\mu a^3}{8}, \quad (50)$$

$$\sum_{t=0}^{T-1} w_t \eta_t = \sum_{t=0}^{T-1} \frac{8(a+t)}{\mu} = \frac{4T^2 + 8aT - 4T}{\mu} \leq \frac{4T(T+2a)}{\mu}, \quad (51)$$

437 and

$$\sum_{t=0}^{T-1} w_t \eta_t^2 = \sum_{t=0}^{T-1} \frac{64}{\mu^2} = \frac{64T}{\mu^2}. \quad (52)$$

438 Let  $S_T := \sum_{t=0}^{T-1} w_t = \frac{T}{6} (2T^2 + 6aT - 3T + 6a^2 - 6a + 1)$ . Observe

$$S_T \geq \frac{1}{3} T^3 + \underbrace{aT^2 - \frac{1}{2} T^2 + a^2 T - aT}_{=T^2(a-\frac{1}{2})+T(a^2-a) \geq 0} \geq \frac{1}{3} T^3. \quad (53)$$

439 for  $a \geq 1, T \geq 0$ .  $\square$

S: this is only useful when  $T \gg 1$ , if  $a$  is crazy large then we want to keep  $a \dots$