Communication-efficient Distributed SGD with Sketching

Nikita Ivkin [*†]
Amazon
ivkin@amazon.com

Daniel Rothchild [*]
UC Berkeley
drothchild@berkeley.edu

Enayat Ullah [*]
Johns Hopkins University
enayat@jhu.edu

Vladimir Braverman [‡]
Johns Hopkins University
vova@cs.jhu.edu

Ion Stoica
UC Berkeley
istoica@berkeley.edu

Raman Arora
Johns Hopkins University
arora@cs.jhu.edu

Abstract

Large-scale distributed training of neural networks is often limited by network bandwidth, wherein the communication time overwhelms the local computation time. Motivated by the success of sketching methods in sub-linear/streaming algorithms, we introduce SKETCHED-SGD[4], an algorithm for carrying out distributed SGD by communicating sketches instead of full gradients. We show that SKETCHED-SGD has favorable convergence rates on several classes of functions. When considering all communication – both of gradients and of updated model weights – SKETCHED-SGD reduces the amount of communication required compared to other gradient compression methods from $\mathcal{O}(d)$ or $\mathcal{O}(W)$ to $\mathcal{O}(\log d)$, where d is the number of model parameters and W is the number of workers participating in training. We run experiments on a transformer model, an LSTM, and a residual network, demonstrating up to a 40x reduction in total communication cost with no loss in final model performance. We also show experimentally that SKETCHED-SGD scales to at least 256 workers without increasing communication cost or degrading model performance.

1 Introduction

Modern machine learning training workloads are commonly distributed across many machines using data-parallel synchronous stochastic gradient descent. At each iteration, W worker nodes split a mini-batch of size B; each worker computes the gradient of the loss on its portion of the data, and then a parameter server sums each worker's gradient to yield the full mini-batch gradient. After using this gradient to update the model parameters, the parameter server must send back the updated weights to each worker. We emphasize that our method can naturally be extended to other topologies as well (e.g. ring, complete, etc.) – in particular we would then communicate sketches over a minimum spanning tree of the communication graph. However, for ease of exposition, in this work we focus exclusively on the star topology. For a fixed batch size B, the amount of data each worker processes – and therefore the amount of computation required – is inversely proportional to W. On the other hand, the amount of communication required per worker is independent of W. Even with optimal interleaving of the communication and computation, the total training time is at least the maximum

[*]equal contribution
[†]This work was done while the author was at Johns Hopkins University.
[‡]This work was done, in part, while the author was visiting the Simons Institute for the Theory of Computing.
[4]Code is available at https://github.com/dhroth/sketchedsgd

33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada.

of the per-worker communication time and per-worker computation time. Increasing the number of workers W therefore yields an increasingly marginal reduction in the training time, despite increasing the overall training cost (number of machines times training time) linearly in W.

Several approaches address this issue by using a large batch size to increase the per-worker computation time [You et al., 2017, Goyal et al., 2017]. However, theoretical and empirical evidence both suggest that there is a maximum mini-batch size beyond which the number of iterations required to converge stops decreasing, and generalization error begins to increase [Ma et al., 2017, Li et al., 2014, Golmant et al., 2018, Shallue et al., 2018, Keskar et al., 2016, Hoffer et al., 2017]. In this paper, we aim instead to decrease the communication cost per worker. We use a technique from streaming algorithms called sketching, which allows us to recover favorable convergence guarantees of vanilla SGD. In short, our algorithm has workers send gradient sketches of size $\mathcal{O}(\log d)$ instead of the gradients themselves. Although other methods for reducing the communication cost exist, to our knowledge ours is the only one that gives a per-worker communication cost that is sub-linear in d and constant in W. In practice, we show that our method achieves high compression for large d with no loss in model accuracy, and that it scales as expected to large W.

2 Related Work

Most existing methods for reducing communication cost in synchronous data-parallel distributed SGD either quantize or sparsify gradients. A number of quantization methods have been proposed. These methods either achieve only a constant reduction in the communication cost per iteration [Wen et al., 2017, Bernstein et al., 2018], or achieve an asymptotic reduction in communication cost per iteration at the expense of an equal (or greater) asymptotic increase in the number of iterations required [Alistarh et al., 2017]. Even in the latter case, the total communication required for all of training sees no asymptotic improvement.

Other methods sparsify the gradients instead of quantizing each gradient element [Stich et al., 2018, Alistarh et al., 2018, Lin et al., 2017]. A popular heuristic is to send the top-k coordinates of the local worker gradients and then average them to obtain an approximate mini-batch gradient. These methods can achieve good performance in practice, but they suffer from a few drawbacks. They currently have no convergence guarantees, since the estimated mini-batch gradient can be very far from the true mini-batch gradient (unless explicitly assumed, as in e.g. Alistarh et al. [2018]), which precludes appealing to any known convergence result. Another drawback is that, although these methods achieve high compression rates when the workers transmit gradients to the parameter server, the return communication of the updated model parameters grows as $\mathcal{O}(W)$: the local top-k of each worker may be disjoint, so there can be as many as kW parameters updated each iteration. This $\mathcal{O}(W)$ communication cost is not just a technicality, since reducing the back-communication to $\mathcal{O}(k)$ would require sparsifying the sum of the local top-k, which could hinder convergence. Because of this scaling, local top-k methods suffer from poor compression in settings with large W.

From another standpoint, all gradient compression techniques yield either biased or unbiased gradient estimates. A number of quantization methods are crafted specifically to yield unbiased estimates, such that the theoretical guarantees of SGD continue to apply [Alistarh et al., 2017, Wen et al., 2017]. However, even without these guarantees, a number of methods using biased gradient estimates were also found to work well in practice [Bernstein et al., 2018, Seide et al., 2014, Strom, 2015]. Recently, Stich et al. [2018], Karimireddy et al. [2019] gave convergence guarantees for this kind of biased compression algorithm, showing that accumulating compression error locally in the workers can overcome the bias in the weight updates as long as the compression algorithm obeys certain properties. Our method falls into this category, and we prove that compressing gradients with sketches obeys these properties and therefore enjoys the convergence guarantees in Stich et al. [2018]. In effect, we introduce a method that extends the theoretical results of Stich et al. [2018] from a single machine to the distributed setting. Concurrently with this work, Koloskova et al. [2019] also introduce a distributed learning algorithm with favorable convergence guarantees, in which workers communicate compressed gradients over an arbitrary network topology.

Prior work has proposed applying sketching to address the communication bottleneck in distributed and Federated Learning [Konečnỳ et al., 2016, Jiang et al., 2018]. However, these methods either do not have provable guarantees, or they apply sketches only to portions of the data, failing to alleviate the $\Omega(Wd)$ communication overhead. In particular, Konečnỳ et al. [2016] propose "sketched updates"

in Federated Learning for structured problems, and Jiang et al. [2018] introduce a range of hashing and quantization techniques to improve the constant in $\mathcal{O}(Wd)$.

Another line of work that we draw from applies sketching techniques to learning tasks where the model itself cannot fit in memory [Aghazadeh et al., 2018, Tai et al., 2018]. In our setting, we can afford to keep a dense version of the model in memory, and we only make use of the memory-saving properties of sketches to reduce communication between nodes participating in distributed learning.

3 Preliminaries

SGD. Let $\mathrm{w} \in \mathbb{R}^d$ be the parameters of the model to be trained and $f_i(\mathrm{w})$ be the loss incurred by w at the i^{th} data point $(\mathrm{x}_i, y_i) \sim \mathcal{D}$. The objective is to minimize the generalization error $f(\mathrm{w}) = \mathop{\mathbb{E}}_{(\mathrm{x}_i,y_i)\sim\mathcal{D}}[f_i(\mathrm{w})]$. In large-scale machine learning, this objective is typically minimized using mini-batch stochastic gradient descent: given a step size η_t, at each iteration, w is updated as $\mathrm{w}_{t+1} = \mathrm{w}_t - \eta_t \mathrm{g}_t$, where $\mathrm{g}_t = \nabla_{\mathrm{w}} \sum_{i\in\mathcal{M}} f_i(\mathrm{w})$ is the gradient of the loss computed on a minibatch $\mathcal{M}$. If $\mathcal{M}$ is randomly selected, then the gradient estimates g_t are unbiased: i.e. $\mathbb{E}\left[g_t | \{\mathrm{w}_i\}_{i=0}^{t-1}\right] = \nabla f(\mathrm{w}_{t-1})$. As is standard, we further assume that the g_t have bounded moment and variance: $\mathbb{E}\left[\|\mathrm{g}_t\|_2^2 | \{\mathrm{w}_i\}_{i=0}^{t-1}\right] \leq G^2$ and $\mathbb{E}\left[\|\mathrm{g}_t - \nabla f(\mathrm{w}_t)\|_2^2 | \{\mathrm{w}_i\}_{i=0}^{t-1}\right] \leq \sigma^2$ for constants G and σ. We adopt the usual definitions for smooth and strongly convex functions:

Definition 1 (Smooth strongly convex function)**.** *$f : \mathbb{R}^d \to \mathbb{R}$ is a L-smooth and μ-strongly convex if the following hold $\forall\ \mathrm{w}_1, \mathrm{w}_2 \in \mathbb{R}^d$,*

1. $\|\nabla f(\mathrm{w}_2) - \nabla f(\mathrm{w}_2)\| \leq L \|\mathrm{w}_2 - \mathrm{w}_1\|$ *(Smoothness)*
2. $f(\mathrm{w}_2) \geq f(\mathrm{w}_1) + \langle \nabla f(\mathrm{w}_1), \mathrm{w}_2 - \mathrm{w}_1 \rangle + \frac{\mu}{2} \|\mathrm{w}_2 - \mathrm{w}_1\|^2$ *(Strong convexity)*

For smooth strongly convex functions, SGD converges at a rate of $\mathcal{O}\left(\frac{G^2 L}{\mu T}\right)$ [Rakhlin et al., 2012].

Count Sketch. Our primary interest is in finding large coordinates (or "heavy hitters") of a gradient vector $\mathrm{g} \in \mathbb{R}^d$. Heavy hitter sketches originated in the streaming model, where the vector g is defined by a sequence of updates $\{(i_j, w_j)\}_{j=1}^n$, such that the j-th update modifies the i_j-th coordinate of g as g_{i_j} += w_j [Charikar et al., 2002, Cormode and Muthukrishnan, 2005, Braverman et al., 2017]. In the streaming model, sketches must use memory sublinear in both d and n.

In this work we compress a gradient vector g into a sketch $S(\mathrm{g})$ of size $O(\frac{1}{\varepsilon}\log d)$ using a Count Sketch [Charikar et al., 2002]. A Count Sketch $S(\mathrm{g})$ approximates every coordinate of g with an ℓ_2 guarantee: it is always possible to recover $\hat{\mathrm{g}}_i$ from $S(\mathrm{g})$ such that $\mathrm{g}_i^2 - \varepsilon\|\mathrm{g}\|_2^2 \leq \hat{\mathrm{g}}_i^2 \leq \mathrm{g}_i^2 + \varepsilon\|\mathrm{g}\|_2^2$. In addition, $S(\mathrm{g})$ can approximate the ℓ_2 norm of the entire gradient. These two properties let a sketch find every ℓ_2 heavy hitter, i.e. every coordinate i such that $\mathrm{g}_i^2 > \varepsilon\|\mathrm{g}\|_2^2$. With a small enough ε, the set of heavy hitters can be used as approximation of top-k largest coordinates of gradient vector g.

Due to its linearity, the Count Sketch is widely adopted in distributed systems. Consider the case of a parameter server and two workers hosting vectors g_1 and g_2. To reduce communication, both workers can send the parameter server sketches $S(\mathrm{g}_1)$ and $S(\mathrm{g}_2)$ instead of the vectors themselves. The parameter server can then merge these sketches as $S(\mathrm{g}) = S(\mathrm{g}_1 + \mathrm{g}_2) = S(\mathrm{g}_1) + S(\mathrm{g}_2)$. This lets the parameter server find the approximate top-k largest coordinates in a vector distributed among many workers. We defer a more detailed discussion of the Count Sketch to Appendix C.

4 Sketched SGD

In SKETCHED-SGD, each worker transmits a sketch of its gradient instead of the gradient itself, as described above. The parameter server sums the workers' sketches, and then recovers the largest gradient elements by magnitude from the summed sketch. To improve the compression properties of sketching, we then perform a second round of communication, in which the parameter server requests the exact values of the top-k, and uses the sum of those in the weight update. This algorithm for recovering top-k elements from a sketch is summarized in Algorithm 1.

Every iteration, only k values of each worker's gradient are included in the final weight update. Instead of discarding the remaining $d - k$ gradient elements, it is important both theoretically and

empirically to accumulate these elements in local error accumulation vectors, which are then added to the next iteration's gradient [Karimireddy et al., 2019, Stich et al., 2018]. This process is summarized in Algorithm 2.

Algorithm 1 HEAVYMIX

Input: S - sketch of gradient g; k - parameter
1: Query $\hat{\ell}_2^2 = (1 \pm 0.5)\|\mathrm{g}\|_2^2$ from sketch S
2: $\forall i$ query $\hat{\mathrm{g}}_i^2 = \mathrm{g}_i^2 \pm \frac{1}{2k}\|\mathrm{g}\|_2^2$ from sketch S
3: $H \leftarrow \left\{i | \hat{\mathrm{g}}_i \geq \hat{\ell}_2^2/k\right\}$ and $NH \leftarrow \left\{i |\ \hat{\mathrm{g}}_i < \hat{\ell}_2^2/k\right\}$
4: $\mathrm{Top}_k = H \cup \mathrm{rand}_l(NH)$, where $l = k - |H|$
5: second round of communication to get exact values of Top_k
Output: $\tilde{\mathrm{g}}$: $\forall i \in \mathrm{Top}_k : \tilde{\mathrm{g}}_i = \mathrm{g}_i$ and $\forall i \notin \mathrm{Top}_k : \tilde{\mathrm{g}}_i = 0$

Algorithm 2 SKETCHED-SGD

Input: k, ξ, T, W
1: $\eta_t \leftarrow \frac{1}{t+\xi}, q_t \leftarrow (\xi + t)^2, Q_T = \sum_{t=1}^{T} q_t, \mathrm{a}_0 = 0$
2: **for** $t = 1, 2, \cdots T$ **do**
3: Compute stochastic gradient g_t^i — Worker$_i$
4: Error correction: $\bar{\mathrm{g}}_t^i = \eta_t \mathrm{g}_t^i + \mathrm{a}_{t-1}^i$ — Worker$_i$
5: Compute sketches S_t^i of $\bar{\mathrm{g}}_t^i$ and send to Parameter Server — Worker$_i$
6: Aggregate sketches $\mathrm{S}_t = \frac{1}{W}\sum_{i=1}^{W} \mathrm{S}_t^i$ — Parameter Server
7: $\tilde{\mathrm{g}}_t$ = HEAVYMIX(S_t, k) — Parameter Server
8: Update $\mathrm{w}_{t+1} = \mathrm{w}_t - \tilde{\mathrm{g}}_t$ and send $\tilde{\mathrm{g}}_t$ (which is k-sparse) to Workers — Parameter Server
9: Error accumulation: $\mathrm{a}_t^i = \bar{\mathrm{g}}_t^i - \tilde{\mathrm{g}}_t$ — Worker$_i$
10: **end for**
Output: $\hat{\mathrm{w}}_T = \frac{1}{Q_T}\sum_{t=1}^{T} q_t \mathrm{w}_t$

We now state convergence results for SKETCHED-SGD. Proofs are deferred to Appendix A.

Theorem 1 (strongly convex, smooth)**.** *Let $f : \mathbb{R}^d \to \mathbb{R}$ be a L-smooth μ-strongly convex function, and let the data be shared among W workers. Given $0 < k \leq d, 0 < \alpha, and \delta < 1$, Algorithm 2* SKETCHED-SGD *run with sketch size $= \mathcal{O}\left(k \log(dT/\delta)\right)$, step size $\eta_t = \frac{1}{t+\xi}$, with $\xi > 2 + \frac{d(1+\beta)}{k(1+\rho)}$, with $\beta > 4$ and $\rho = \frac{4\beta}{(\beta-4)(\beta+1)^2}$ after T steps outputs $\hat{\mathrm{w}}_T$ such that the following holds,*

1. *With probability at least $1 - \delta$, $\mathbb{E}\left[f(\hat{\mathrm{w}}_T)\right] - f(\mathrm{w}^*) \leq \mathcal{O}\left(\frac{\sigma^2}{\mu T} + \frac{d^2 G^2 L}{k^2 \mu^2 T^2} + \frac{d^3 G^3}{k^3 \mu T^3}\right)$*
2. *The total communication per update is $\Theta(k \log(dT/\delta) W)$ bits.*

Remarks

1. The convergence rate for vanilla SGD is $\mathcal{O}(1/T)$. Therefore, our error is larger the SGD error when $T = o((d/k)^2)$, and approaches the SGD error for $T = \Omega((d/k)^2)$.
2. Although not stated in this theorem, Stich et al. [2018] show that using the top-k coordinates of the true mini-batch gradient as the SGD update step yields a convergence rate equivalent to that of SKETCHED-SGD. We therefore use this "true top-k" method as a baseline for our results.
3. Note that the leading term in the error is $O(\sigma^2/T)$ (as opposed to $O(G^2/T)$ in [Stich et al., 2018]); this implies that in setting where the largest minibatch size allowed is too large to fit in one machine, and going distributed allows us to use larger mini-batches, the variance reduces by a factor W. This reduces the number of iterations required (asymptotically) linearly with W.
4. As is standard, the above high probability bound can be converted to an expectation (over randomness in sketching) bound; this is stated as Theorem 6 in the Appendix A.
5. The result of [Karimireddy et al., 2019] allows us to extend our theorems to smooth nonconvex and non-smooth convex functions; these are presented as Theorems 4 and 5 in the Appendix B..

Proof Sketch. The proof consists of two parts. First, we show that SKETCHED-SGD satisfies the criteria in Stich et al. [2018], from which we obtain a convergence result when running SKETCHED-SGD on a single machine. We then use properties of the Count Sketch to extend this result to the distributed setting.

For the first part, the key idea is to show that our heavy hitter recovery routine HEAVYMIX satisfies a *contraction* property, defined below.

Definition 2 (τ-contraction [Stich et al., 2018])**.** *A τ-contraction operator is a possibly randomized operator comp : $\mathbb{R}^d \to \mathbb{R}^d$ that satisfies: $\forall \mathrm{x} \in \mathbb{R}^d$, $\mathbb{E}\left[\|\mathrm{x} - comp(\mathrm{x})\|^2\right] \leq (1-\tau)\|\mathrm{x}\|^2$*

Given a contraction operator with $\tau = k/d$, and assuming that the stochastic gradients g are unbiased and bounded as $\mathbb{E}\left[\|\mathrm{g}\|^2\right] \leq G^2$, choosing the step-size appropriately, Stich et al. [2018] give a convergence rate of $\mathcal{O}\left(\frac{G^2}{\mu T} + \frac{d^2G^2L}{k^2\mu^2T^2} + \frac{d^3G^3}{k^3\mu T^3}\right)$ for sparsified SGD with error accumulation. As stated in Lemma 1, HEAVYMIX satisfies this contraction property, and therefore inherits this (single-machine) convergence result:

Lemma 1. HEAVYMIX*, with sketch size $\Theta(k\log(d/\delta))$ is a k/d-contraction with probability $\geq 1-\delta$.*

This completes the first part of the proof. To extend SKETCHED-SGD to the distributed setting, we exploit the fact that Count Sketches are linear, and can approximate ℓ_2 norms. The full proof is deferred to Appendix A.

5 Empirical Results

5.1 Training Algorithm

In practice, we modify SKETCHED-SGD in the following ways

- We employ momentum when training. Following Lin et al. [2017], we use momentum correction and momentum factor masking. Momentum factor masking mitigates the effects of stale momentum, and momentum correction is a way to do error feedback on SGD with momentum [Karimireddy et al., 2019].
- We use the Count Sketch to identify heavy coordinates, however we perform an additional round of communication to collect the exact values of those coordinates. In addition, to identify the top k heavy coordinates, we query the Count Sketch, and then each of the workers, for the top Pk elements instead; this is a common technique used with sketching to improve stability. The total resulting communication cost is $Pk + |S| + k$ per worker, where $|S|$ is the size of the sketch, and the last k corresponds to the the updated model parameters the parameter server must send back to the workers.
- We transmit gradients of the bias terms uncompressed. The number of bias terms in our models is $< 1\%$ of the total number of parameters.

Our emperical training procedure is summarized in Algorithm 3.

Algorithm 3 EMPIRICAL TRAINING

Input: k, η_t, m, T

$\forall i : \mathrm{u}^i, \mathrm{v}^i \leftarrow 0$
Initialize w_0^i from the same random seed on each Worker.
for $t = 1, 2, \ldots T$ **do**
  Compute stochastic gradient g_t^i — Worker$_i$
  Momentum: $\mathrm{u}^i \leftarrow m\mathrm{u}^i + \mathrm{g}_t^i$ — Worker$_i$
  Error accumulation: $\mathrm{v}^i \leftarrow \mathrm{v}^i + \mathrm{u}^i$ — Worker$_i$
  Compute sketch S_t^i of v^i and send to Parameter Server — Worker$_i$
  Aggregate sketches $\mathrm{S}_t = \frac{1}{W}\sum_{i=1}^W \mathrm{S}_t^i$ — Parameter Server
  Recover the top-Pk coordinates from S_t: $\tilde{\mathrm{g}}_t = top_{Pk}(S_t)$ — Parameter Server
  Query all workers for exact values of nonzero elements in $\tilde{\mathrm{g}}_t$; store the sum in $\tilde{\mathrm{g}}_t$ — Parameter Server
  Send the k-sparse $\tilde{\mathrm{g}}_t$ to Workers — Parameter Server
  update $\mathrm{w}_{t+1}^i = \mathrm{w}_t^i - \eta_t\tilde{\mathrm{g}}_t$ on each worker — Worker$_i$
  $\mathrm{u}^i, \mathrm{v}^i \leftarrow 0$, for all i s.t. $\tilde{\mathrm{g}}_t^i \neq 0$ — Worker$_i$
end for

5.2 Sketching Implementation

We implement a parallelized Count Sketch with PyTorch [Paszke et al., 2017]. The Count Sketch data structure supports a query method, which returns a provable $\pm\varepsilon\|\mathrm{g}\|_2$ approximation to each

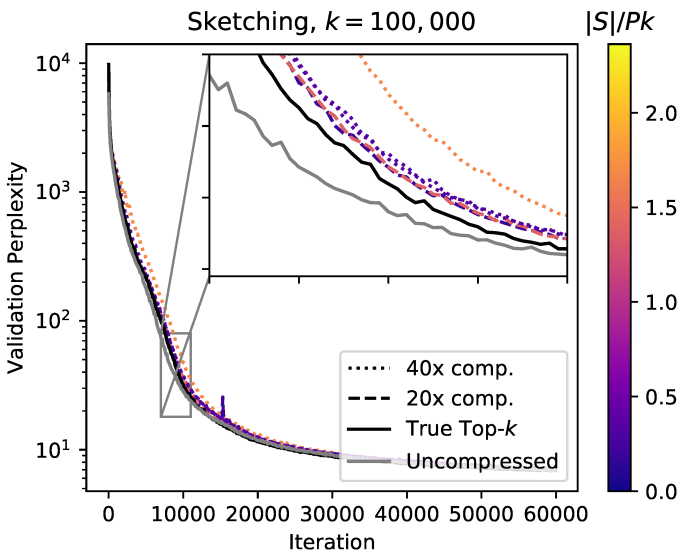


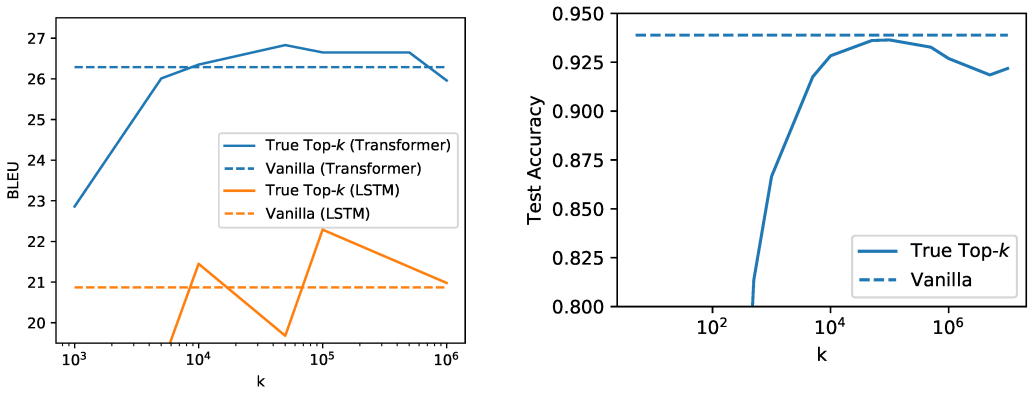
Figure 1: Learning curves for a transformer model trained on the WMT 2014 English to German translation task. All models included here achieve comparable BLEU scores after 60,000 iterations (see Table 1). Each run used 4 workers.

coordinate value. However, to the best of our knowledge, there is no efficient way to find heavy coordinates in the presence of negative inputs. Fortunately, in our application, it is computationally efficient on the GPU to simply query the sketch for every gradient coordinate, and then choose the largest elements.

5.3 Large d

First, we show that SKETCHED-SGD achieves high compression with no loss in accuracy. Because the sketch size grows as $\mathcal{O}(\log d)$, we expect to see the greatest compression rates for large d. Accordingly, we test on a transformer model with 90M parameters, and on a stacked LSTM model with 73M parameters. We train both models on the WMT 2014 English to German translation task, and we use code from the OpenNMT project [Klein et al., 2017]. In all cases, the compression factor for SKETCHED-SGD is computed as $2d/(|S| + Pk + k)$, where $2d$ is the cost to send a (dense) gradient and receive a new (dense) parameter vector, $|S|$ is the sketch size, Pk is the number of elements sent in the second round of communication, and the last k represents the number of modified parameter values that must be sent back to each worker.

SKETCHED-SGD achieves the same theoretical convergence rate as top-k SGD, in which the weight update consists of the top-k elements of the full mini-batch gradient. We therefore perform experiments with SKETCHED-SGD using a value of k that yields good performance for top-k SGD. Figure 2 shows top-k results over a range of values of k. Curiously, performance starts to degrade for large k. Although performance on the training data should in principle strictly improve for larger k, sparsifying gradients regularizes the model, so $k < d$ may yield optimal performance on the test set. In addition, we expect performance to degrade on both the training and test sets for large k due to momentum factor masking. To mitigate stale momentum updates, momentum factor masking zeros the velocity vector at the k coordinates that were updated in each iteration. In the limit $k = d$, this completely negates the momentum, hindering convergence. For all SKETCHED-SGD experiments on these two models, we use $k = 100,000$, for which top-k SGD yields a BLEU score of 26.65 for the transformer and 22.2 for the LSTM. For reference, uncompressed distributed SGD with the same hyperparameters achieves a BLEU of 26.29 for the transformer and 20.87 for the LSTM. Using SKETCHED-SGD, we can obtain, with no loss in BLEU, a 40x reduction in the total communication cost during training, including the cost to disseminate updated model parameters. See Table 1 for a summary of BLEU results. Compression numbers include both the communication required to send gradients as well as the cost to send back the new model parameters. We do not include the cost to



(a) WMT14 Translation Task

(b) CIFAR-10 Classification Task

Figure 2: True top-k results for a range of k. Left: two models (transformer and LSTM) on the WMT 2014 English to German translation task. Right: a residual network on the CIFAR-10 classification task. For the larger models (left), true top-k slightly outperforms the baseline for a range of k. We suspect this is because k-sparsifying gradients serves to regularize the model.

	BLEU (transformer)	BLEU (LSTM)
Uncompressed Distributed SGD	26.29	20.87
Top-100,000 SGD	26.65	22.2
SKETCHED-SGD, 20x compression	26.87[5]	–
SKETCHED-SGD, 40x compression	26.79[6]	20.95 [7]

Table 1: BLEU scores on the test data achieved for uncompressed distributed SGD, top-k SGD, and SKETCHED-SGD with 20x and 40x compression. Compression rates represent the total reduction in communication, including the cost to transmit the updated model parameters. Larger BLEU score is better. For both models, top-k SGD with $k = 100,000$ achieves a higher BLEU score than uncompressed distributed SGD. This difference may be within the error bars, but if not, it may be that stepping in only the direction of the top-k is serving as a regularizer on the optimizer. Our main experiments are on the transformer model, for which we run additional experiments using 20x compression that we did not complete for the LSTM model.

request the Pk coordinates, nor to specify which k model parameters have been updated, since these quantities can be efficiently coded, and contribute little to the overall communication.

Given that our algorithm involves a second round of communication in which Pk gradient elements are transmitted, we investigate the tradeoff between a large sketch size and a large value of P. Approaching a sketch size of zero corresponds to using a weight update that is the top-k of a randomly chosen set of Pk gradient coordinates. Experiments with extremely small sketch size $|S|$ or extremely small values of P tended to diverge or achieve very low BLEU score. For values of $|S|/Pk$ closer to 1, we plot learning curves in Figure 1. As expected, uncompressed SGD trains fastest, followed by top-k SGD, then 20x compression SKETCHED-SGD, then 40x compression SKETCHED-SGD. For the two 20x compression runs, the ratio of the sketch size to the number of exact gradient values computed has little effect on convergence speed. However, the higher compression runs prefer a relatively larger value of P.

5.4 Large W

To re-iterate, the per-worker communication cost for SKETCHED-SGD is not only sub-linear in d, but also independent of W. To demonstrate the power of this experimentally, we train a residual

[5] Sketch size: 5 rows by 1M columns; $P = 36$.

[6] Sketch size: 15 rows by 180,000 columns; $P = 16$.

[7] Sketch size: 5 rows by 180,000 columns, $P = 26$

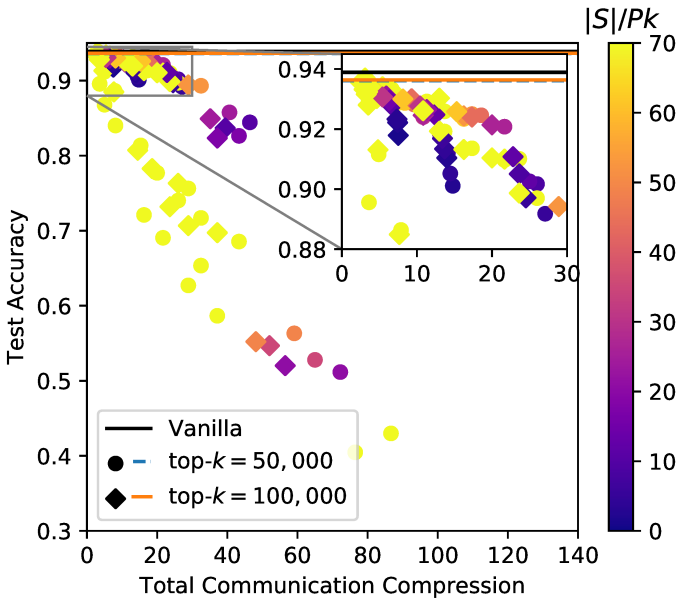


Figure 3: Tradeoff between compression and model accuracy for a residual network trained on CIFAR-10. We show results for $k = 50,000$ as well as $k = 100,000$, and color code each trained model based on the ratio of sketch size to the cost of the second round of communication. The (nearly overlapping) solid orange and dashed blue lines show the accuracy achieved by top$-k$ SGD for the two values of k, and the black line shows the accuracy achieved by uncompressed distributed SGD. All models in this plot were trained with 4 workers.

network on the CIFAR-10 dataset with SKETCHED-SGD, using up to 256 workers [Krizhevsky and Hinton, 2009]. We compare to local top-k, a method where each worker computes and transmits only the top-k elements of its gradient. The version of local top-k SGD we compare to is similar to Deep Gradient Compression, except we do not clip gradients, and we warm up the learning rate instead of the sparsity [Lin et al., 2017]. Results are shown in Figure 4. Neither algorithm sees an appreciable drop in accuracy with more workers, up to $W = 256$. However, while the communication cost of SKETCHED-SGD is constant in W, the communication cost for local top-k scales with W until reaching $\Theta(d)$. This scaling occurs because the local top-k of each worker might be disjoint, leading to as many as kW parameters being updated. In practice, we do in fact observe nearly linear scaling of the number of parameters updated each iteration, until saturating at d (dashed orange line in Figure 4). For $W = 256$, the communication of the updated model parameters back to each worker is nearly dense ($d \approx 6.5 \times 10^6$), reducing the overall compression of local top-k to at best $\sim 2\times$.

For a fixed small number of workers ($W = 4$), we also investigate the tradeoff between compression rate and final test accuracy. Figure 3 shows this tradeoff for two values of k and a wide range of sketch sizes and values of P. As expected, increasing the compression rate leads to decreasing test accuracy. In addition, as evidenced by the color coding, using a very large sketch size compared to Pk tends to yield poor results. Although high compression rates decrease accuracy, in our experience, it is possible to make up for this accuracy drop by training longer. For example, choosing one of the points in Figure 3, training with 17x compression for the usual number of iterations gives 92.5% test accuracy. Training with 50% more iterations (reducing to 11x overall compression) restores accuracy to 94%. In Figure 3, every model is trained for the same number of iterations.

6 Discussion

In this work we introduce SKETCHED-SGD, an algorithm for reducing the communication cost in distributed SGD using sketching. We provide theoretical and experimental evidence that our method can help alleviate the difficulties of scaling SGD to many workers. While uncompressed distributed SGD requires communication of size $2d$, and other gradient compressions improve this to $\mathcal{O}(d)$

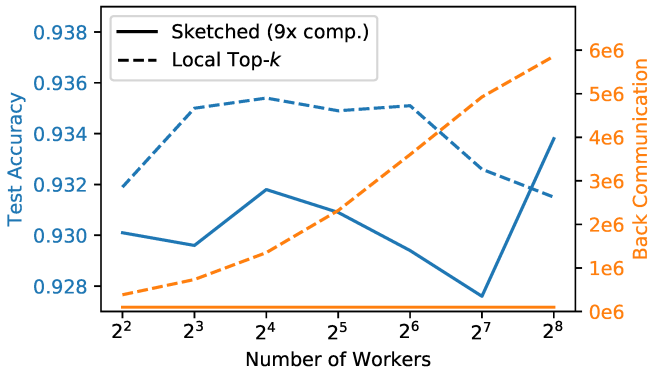


Figure 4: Comparison between SKETCHED-SGD and local top-k SGD on CIFAR10. Neither algorithm sees an appreciable drop in performance for up to 256 workers, but the amount of communication required for local top-k grows quickly to $\approx d = 6.5 \times 10^6$ as the number of workers increases. As a result, the best overall compression that local top-k can achieve for many workers is 2x.

or $\mathcal{O}(W)$, SKETCHED-SGD further reduces the necessary communication to $\mathcal{O}(\log d)$. Besides reducing communication, our method provably converges at the same rate as SGD, and in practice we are able to reduce the total communication needed by up to 40x without experiencing a loss in model quality.

A number of other techniques for efficient training could be combined with SKETCHED-SGD, including gradient quantization and asynchronous updates. We expect that the advantages asynchronous updates bring to regular SGD will carry over to SKETCHED-SGD. And given that elements of gradient sketches are sums of gradient elements, we expect that quantizing sketches will lead to similar tradeoffs as quantizing the gradients themselves. Preliminary experiments show that quantizing sketches to 16 bits when training our ResNets on CIFAR-10 leads to no drop in accuracy, but we leave a full evaluation of combining quantization, as well as asynchronous updates, with SKETCHED-SGD to future work.

Machine learning models are constantly growing in size (e.g. OpenAI's GPT-2, a transformer with 1.5 billion parameters [Radford et al., 2019]), and training is being carried out on a larger and larger number of compute nodes. As communication increasingly becomes a bottleneck for large-scale training, we argue that a method that requires only $\mathcal{O}(\log d)$ communication has the potential to enable a wide range of machine learning workloads that are currently infeasible, from highly parallel training in the cloud, to Federated Learning at the edge [McMahan et al., 2016].

7 Acknowledgements

This research was supported, in part, by NSF BIGDATA grants IIS-1546482 and IIS-1838139, NSF CAREER grant 1652257, ONR Award N00014-18-1-2364 and the Lifelong Learning Machines program from DARPA/MTO. This material is based upon work supported by the National Science Foundation Graduate Research Fellowship under Grant No. DGE 1752814.

References

Pankaj K Agarwal, Graham Cormode, Zengfeng Huang, Jeff M Phillips, Zhewei Wei, and Ke Yi. Mergeable summaries. *ACM Transactions on Database Systems (TODS)*, 38(4):26, 2013.

Amirali Aghazadeh, Ryan Spring, Daniel Lejeune, Gautam Dasarathy, Anshumali Shrivastava, and richard baraniuk. MISSION: Ultra large-scale feature selection using count-sketches. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 80–88,

Stockholmsmässan, Stockholm Sweden, 10–15 Jul 2018. PMLR. URL http://proceedings.mlr.press/v80/aghazadeh18a.html.

Dan Alistarh, Demjan Grubic, Jerry Li, Ryota Tomioka, and Milan Vojnovic. Qsgd: Communication-efficient sgd via gradient quantization and encoding. In *Advances in Neural Information Processing Systems*, pages 1709–1720, 2017.

Dan Alistarh, Torsten Hoefler, Mikael Johansson, Nikola Konstantinov, Sarit Khirirat, and Cédric Renggli. The convergence of sparsified gradient methods. In *Advances in Neural Information Processing Systems*, pages 5977–5987, 2018.

Noga Alon, Yossi Matias, and Mario Szegedy. The space complexity of approximating the frequency moments. *Journal of Computer and system sciences*, 58(1):137–147, 1999.

Jeremy Bernstein, Yu-Xiang Wang, Kamyar Azizzadenesheli, and Anima Anandkumar. signsgd: compressed optimisation for non-convex problems. *arXiv preprint arXiv:1802.04434*, 2018.

Vladimir Braverman, Stephen R Chestnut, Nikita Ivkin, Jelani Nelson, Zhengyu Wang, and David P Woodruff. Bptree: An ℓ_2 heavy hitters algorithm using constant memory. In *Proceedings of the 36th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*, pages 361–376. ACM, 2017.

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

Cody Coleman, Deepak Narayanan, Daniel Kang, Tian Zhao, Jian Zhang, Luigi Nardi, Peter Bailis, Kunle Olukotun, Chris Ré, and Matei Zaharia. Dawnbench: An end-to-end deep learning benchmark and competition. *Training*, 100(101):102, 2017.

Graham Cormode and Shan Muthukrishnan. An improved data stream summary: the count-min sketch and its applications. *Journal of Algorithms*, 55(1):58–75, 2005.

Noah Golmant, Nikita Vemuri, Zhewei Yao, Vladimir Feinberg, Amir Gholami, Kai Rothauge, Michael W Mahoney, and Joseph Gonzalez. On the computational inefficiency of large batch sizes for stochastic gradient descent. *arXiv preprint arXiv:1811.12941*, 2018.

Priya Goyal, Piotr Dollár, Ross Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. Accurate, large minibatch sgd: Training imagenet in 1 hour. *arXiv preprint arXiv:1706.02677*, 2017.

Elad Hoffer, Itay Hubara, and Daniel Soudry. Train longer, generalize better: closing the generalization gap in large batch training of neural networks. In *Advances in Neural Information Processing Systems*, pages 1731–1741, 2017.

Nikita Ivkin, Zaoxing Liu, Lin F Yang, Srinivas Suresh Kumar, Gerard Lemson, Mark Neyrinck, Alexander S Szalay, Vladimir Braverman, and Tamas Budavari. Scalable streaming tools for analyzing n-body simulations: Finding halos and investigating excursion sets in one pass. *Astronomy and computing*, 23:166–179, 2018.

Jiawei Jiang, Fangcheng Fu, Tong Yang, and Bin Cui. Sketchml: Accelerating distributed machine learning with data sketches. In *Proceedings of the 2018 International Conference on Management of Data*, pages 1269–1284. ACM, 2018.

Sai Praneeth Karimireddy, Quentin Rebjock, Sebastian U Stich, and Martin Jaggi. Error feedback fixes signsgd and other gradient compression schemes. *arXiv preprint arXiv:1901.09847*, 2019.

Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. On large-batch training for deep learning: Generalization gap and sharp minima. *arXiv preprint arXiv:1609.04836*, 2016.

Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander Rush. OpenNMT: Open-source toolkit for neural machine translation. In *Proceedings of ACL 2017, System Demonstrations*, pages 67–72, Vancouver, Canada, July 2017. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/P17-4012.

Anastasia Koloskova, Sebastian U Stich, and Martin Jaggi. Decentralized stochastic optimization and gossip algorithms with compressed communication. *arXiv preprint arXiv:1902.00340*, 2019.

Jakub Konečnỳ, H Brendan McMahan, Felix X Yu, Peter Richtárik, Ananda Theertha Suresh, and Dave Bacon. Federated learning: Strategies for improving communication efficiency. *arXiv preprint arXiv:1610.05492*, 2016.

Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.

Mu Li, Tong Zhang, Yuqiang Chen, and Alexander J Smola. Efficient mini-batch training for stochastic optimization. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 661–670. ACM, 2014.

Yujun Lin, Song Han, Huizi Mao, Yu Wang, and William J Dally. Deep gradient compression: Reducing the communication bandwidth for distributed training. *arXiv preprint arXiv:1712.01887*, 2017.

Siyuan Ma, Raef Bassily, and Mikhail Belkin. The power of interpolation: Understanding the effectiveness of sgd in modern over-parametrized learning. *arXiv preprint arXiv:1712.06559*, 2017.

Horia Mania, Xinghao Pan, Dimitris Papailiopoulos, Benjamin Recht, Kannan Ramchandran, and Michael I Jordan. Perturbed iterate analysis for asynchronous stochastic optimization. *arXiv preprint arXiv:1507.06970*, 2015.

H. Brendan McMahan, Eider Moore, Daniel Ramage, and Blaise Agüera y Arcas. Federated learning of deep networks using model averaging. *CoRR*, abs/1602.05629, 2016. URL http://arxiv.org/abs/1602.05629.

Shanmugavelayutham Muthukrishnan et al. Data streams: Algorithms and applications. *Foundations and Trends® in Theoretical Computer Science*, 1(2):117–236, 2005.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch, 2017.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. *OpenAI Blog*, 1:8, 2019.

Alexander Rakhlin, Ohad Shamir, Karthik Sridharan, et al. Making gradient descent optimal for strongly convex stochastic optimization. In *ICML*, volume 12, pages 1571–1578. Citeseer, 2012.

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

Christopher J Shallue, Jaehoon Lee, Joe Antognini, Jascha Sohl-Dickstein, Roy Frostig, and George E Dahl. Measuring the effects of data parallelism on neural network training. *arXiv preprint arXiv:1811.03600*, 2018.

Sebastian U Stich, Jean-Baptiste Cordonnier, and Martin Jaggi. Sparsified sgd with memory. In *Advances in Neural Information Processing Systems*, pages 4452–4463, 2018.

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

Kai Sheng Tai, Vatsal Sharan, Peter Bailis, and Gregory Valiant. Sketching linear classifiers over data streams. In *Proceedings of the 2018 International Conference on Management of Data*, pages 757–772. ACM, 2018.

Wei Wen, Cong Xu, Feng Yan, Chunpeng Wu, Yandan Wang, Yiran Chen, and Hai Li. Terngrad: Ternary gradients to reduce communication in distributed deep learning. In *Advances in neural information processing systems*, pages 1509–1519, 2017.

Yang You, Igor Gitman, and Boris Ginsburg. Large batch training of convolutional networks. *arXiv preprint arXiv:1708.03888*, 2017.

Supplementary

A Proofs

Proof of Lemma 1. Given $\mathrm{g} \in \mathbb{R}$, the HEAVYMIX algorithm extracts all $(1/k, \ell_2^2)$-heavy elements from a Count Sketch S of g. Let $\hat{\mathrm{g}}$ be the values of all elements recovered from its sketch. For a fixed k, we create two sets H (heavy), and NH (not-heavy). All coordinates of $\hat{\mathrm{g}}$ with values at least $\frac{1}{k}\hat{\ell}_2^2$ are put in H, and all others in NH, where $\hat{\ell}_2$ is the estimate of $\|\mathrm{g}\|_2$ from the Count Sketch. Note that the number of elements in H can be at most k. Then, we sample uniformly at random $l = k - |H|$ elements from NH , and finally output its union with H. We then do a second round of communication to get exact values of these k elements.

Note that, because of the second round of communication in HEAVYMIX and the properties of the Count Sketch, with probability at least $1 - \delta$ we get the exact values of all elements in H. Call this the "heavy hitters recovery" event. Let g_H be a vector equal to g at the coordinates in H, and zero otherwise. Define g_{NH} analogously. Conditioning on the heavy hitters recovery event, and taking expectation over the random sampling, we have

$$\begin{aligned}\mathbb{E}\left[\left\|\mathrm{g} - \tilde{\mathrm{g}}\right\|^2\right] &= \left\|\mathrm{g}_H - \bar{\mathrm{g}}_H\right\|^2 + \mathbb{E}\left[\left\|\mathrm{g}_{NH} - \mathrm{rand}_l\left(\mathrm{g}_{NH}\right)\right\|^2\right] \\ &\leq \left(1 - \frac{k - |H|}{d - |H|}\right)\left\|\mathrm{g}_{NH}\right\|^2 \leq \left(1 - \frac{k - |H|}{d - |H|}\right)\left(1 - \frac{|H|}{2k}\right)\|\mathrm{g}\|^2\end{aligned}$$

Note that, because we condition on the heavy hitter recovery event, $\bar{\mathrm{g}}_H = \mathrm{g}_H$ due to the second round communication (line 9 of Algorithm 3). The first inequality follows using Lemma 1 from Stich et al. [2018]. The second inequality follows from the fact that the heavy elements have values at least $\frac{1}{k}\hat{\ell}_2^2 \geq \frac{1}{2k}\|\mathrm{g}\|^2$, and therefore $\left\|\mathrm{g}_{NH}\right\|^2 = \|\mathrm{g}\|^2 - \left\|\mathrm{g}_H\right\|^2 \leq \left(1 - \frac{|H|}{2k}\right)\|\mathrm{g}\|^2$.

Simplifying the expression, we get

$$\mathbb{E}\left[\left\|\mathrm{g} - \tilde{\mathrm{g}}\right\|^2\right] \leq \left(\frac{2k - |H|}{2k}\right)\left(\frac{d - k}{d - |H|}\right)\|\mathrm{g}\|^2 = \left(\frac{2k - |H|}{2k}\right)\left(\frac{d}{d - |H|}\right)\left(1 - \frac{k}{d}\right)\|\mathrm{g}\|^2.$$

Note that the first two terms can be bounded as follows:

$$\left(\frac{2k - |H|}{2k}\right)\left(\frac{d}{d - |H|}\right) \leq 1 \iff kd - |H|\,d \leq kd - 2k\,|H| \iff |H|\,(d - 2k) \geq 0$$

which holds when $k \leq d/2$ thereby completing the proof.

□

A.1 Proof of the main theorem

Proof of Theorem 1. First note that, from linearity of sketches), the top-k (or heavy) elements from the merged sketch $\mathrm{S}_t = \sum_{i=1}^{W} \mathrm{S}_t^i$ are the top-k of the sum of vectors that were sketched. We have already shown in Lemma 1 that that extracting the top-k elements from S_T using HEAVYMIX gives us a k-contraction on the sum of gradients. Moreover since the guarantee is relative and norms are positive homogeneous, the same holds for the average, i.e. when dividing by W. Now since the average of stochastic gradients is still an unbiased estimate, this reduces to SKETCHED-SGD on one machine, and the convergence therefore follows from Theorem 2. □

A key ingredient is the result in the one machine setting, stated below.

Theorem 2. *Let $f : \mathbb{R}^d \to \mathbb{R}$ be a L-smooth μ-strongly convex function. Given $T > 0$ and $0 < k \leq d, 0 < \delta < 1$, and a τ_k-contraction, Algorithm 2* SKETCHED-SGD *with sketch size $\mathcal{O}\left(k \log(dT/\delta)\right)$ and step size $\eta_t = \frac{1}{t+\xi}$, with $\xi > 1 + \frac{1+\beta}{\tau_k(1+\rho)}$, with $\beta > 4$ and $\rho = \frac{4\beta}{(\beta-4)(\beta+1)^2}$, after T steps outputs $\hat{\mathrm{w}}_T$ such that with probability at least $1 - \delta$*

$$\mathbb{E}\left[f(\hat{\mathrm{w}}_T)\right] - f(\mathrm{w}^*) \leq \mathcal{O}\left(\frac{\sigma^2}{\mu T} + \frac{G^2 L}{\tau_k^2 \mu^2 T^2} + \frac{G^3}{\tau_k^3 \mu T^3}\right)$$

Proof of Theorem 2. The proof, as in Stich et al. [2018], just follows using convexity and Lemmas 3,2 and fact 3. The lemmas which are exactly same as Stich et al. [2018], are stated as facts. However, the proofs of lemmas, which change are stated in full for completeness, with the changes highlighted.

From convexity we have that

$$f\left(\frac{1}{Q_T}\sum_{i=1}^{T} q_t \mathrm{w}_t\right) - f(\mathrm{w}^*) \leq \frac{1}{Q_T}\sum_{t=1}^{T} q_t f(\mathrm{w}_t) - f(\mathrm{w}^*) = \frac{1}{Q_T}\sum_{t=1}^{T} q_t \left(f(\mathrm{w}_t) - f(\mathrm{w}^*)\right)$$

Define $\epsilon_t = f(\mathrm{w}_t) - f(\mathrm{w}^*)$, the excess error of iterate t. From Lemma 2 we have,

$$\mathbb{E}\left[\|\tilde{\mathrm{w}}_{t+1} - \mathrm{w}^*\|^2\right] \leq \left(1 - \frac{\eta_t \mu}{2}\right)\mathbb{E}\left[\|\tilde{\mathrm{w}}_t - \mathrm{w}^*\|^2\right] + \sigma^2\eta_t^2 - \left(1 - \frac{2}{\xi}\right)\epsilon_t\eta_t + (\mu + 2L)\mathbb{E}\left[\|\mathrm{a}_t\|^2\right]\eta_t$$

Bounding the last term using Lemma 3, with probability at least $1 - \delta$, we get,

$$\mathbb{E}\left[\|\tilde{\mathrm{w}}_{t+1} - \mathrm{w}^*\|^2\right] \leq \left(1 - \frac{\eta_t \mu}{2}\right)\mathbb{E}\left[\|\tilde{\mathrm{w}}_t - \mathrm{w}^*\|^2\right] + \sigma^2\eta_t^2 - \left(1 - \frac{2}{\xi}\right)\epsilon_t\eta_t + \frac{(\mu + 2L)4\beta G^2}{\tau_k^2(\beta - 4)}\eta_t^3$$

where τ_k is the contraction we get from HEAVYMIX. We have alreay show that $\tau_k \leq \frac{k}{d}$.

Now using Lemma 3 and the fist equation, we get,

$$f\left(\frac{1}{Q_T}\sum_{i=1}^{T} q_t \mathrm{w}_t\right) - f(\mathrm{w}^*) \leq \frac{\mu\xi^4\mathbb{E}\left[\|\mathrm{w}_0 - \mathrm{w}^*\|^2\right]}{8(\xi - 2)Q_T} + \frac{4T(T + 2\xi)\xi\sigma^2}{\mu(\xi - 2)Q_T} + \frac{256(\mu + 2L)\beta\xi G^2 T}{\mu^2(\beta - 4)\tau_k^2(\xi - 2)Q_T}$$

Note that $\xi > 2 + \frac{1+\beta}{\tau_k(1+\rho)}$. Moreover $Q_T = \sum_{t=1}^{T} q_t = \sum_{t=1}^{T}(\xi + t)^2 \geq \frac{1}{3}T^3$ upon expanding and using the conditions on ξ. Also $\xi/(\xi - 2) = \mathcal{O}\left(1 + 1/\tau_k\right)$.

Finally using $\sigma^2 \leq G^2$ and Fact 1 to bound $\mathbb{E}\left[\|\mathrm{w}_0 - \mathrm{w}^*\|^2\right] \leq 4G^2/\mu^2$ completes the proof.

□

Lemma 2. *Let $f : \mathbb{R}^d \to \mathbb{R}$ be a L-smooth μ-strongly convex function, and w^* be its minima. Let $\{\mathrm{w}_t\}_t$ be a sequence of iterates generated by Algorithm 2.*

Define error $\epsilon_t := \mathbb{E}\left[f(\mathrm{w}_t) - f(\mathrm{w}^*)\right]$ and $\tilde{\mathrm{w}}_{t+1} = \tilde{\mathrm{w}}_t - \eta_t \mathrm{g}_t$ be a stochastic gradient update step at time t, with $\mathbb{E}\left[\|\mathrm{g}_t - \nabla f(\mathrm{w}_t)\|^2\right] \leq \sigma^2$, $\mathbb{E}\left[\|\mathrm{g}_t\|^2\right] \leq G^2$ and $\eta_t = \frac{1}{\mu(t+\xi)}, \xi > 2$ then we have,

$$\mathbb{E}\left[\|\tilde{\mathrm{w}}_{t+1} - \mathrm{w}^*\|^2\right] \leq \left(1 - \frac{\eta_t \mu}{2}\right)\mathbb{E}\left[\|\tilde{\mathrm{w}}_t - \mathrm{w}^*\|^2\right] + \sigma^2\eta_t - \left(1 - \frac{2}{\xi}\right)\epsilon_t\eta_t + (\mu + 2L)\mathbb{E}\left[\|\mathrm{a}_t\|^2\right]\eta_t$$

Proof. This is the first step of the perturbed iterate analysis framework Mania et al. [2015]. We follow the steps as in Stich et al. [2018]. The only change is that the proof of Stich et al. [2018] works with bounded gradients i.e. $\mathbb{E}\left[\|\mathrm{g}\|^2\right] \leq G^2$. This assumption alone, doesn't provide the variance reduction effect in the distributed setting. We therefore adapt the analysis with the the variance bound $\mathbb{E}\left[\|\mathrm{g} - \nabla f(\mathrm{w})\|^2\right] \leq \sigma^2$.

$$\|\tilde{\mathrm{w}}_{t+1} - \mathrm{w}^*\|^2 = \|\tilde{\mathrm{w}}_{t+1} - \tilde{\mathrm{w}}_t + \tilde{\mathrm{w}}_t - \mathrm{w}^*\|^2 = \|\tilde{\mathrm{w}}_{t+1} - \tilde{\mathrm{w}}_t\|^2 + \|\tilde{\mathrm{w}}_t - \mathrm{w}^*\|^2 + 2\langle\tilde{\mathrm{w}}_t - \mathrm{w}^*, \tilde{\mathrm{w}}_{t+1} - \tilde{\mathrm{w}}_t\rangle$$
$$= \eta_t^2\|g_t\|^2 + \|\tilde{\mathrm{w}}_t - \mathrm{w}^*\|^2 + 2\langle\tilde{\mathrm{w}}_t - \mathrm{w}^*, \tilde{\mathrm{w}}_{t+1} - \tilde{\mathrm{w}}_t\rangle = \eta_t^2\|g_t - \nabla f(\mathrm{w}_t)\|^2 + \eta_t^2\|\nabla f(\mathrm{w}_t)\|^2 + \|\tilde{\mathrm{w}}_t - \mathrm{w}^*\|^2$$
$$+ 2\eta_t\langle\mathrm{g}_t - \nabla f(\mathrm{w}_t), \nabla f(\mathrm{w}_t)\rangle + 2\eta_t\langle\mathrm{w}^* - \tilde{\mathrm{w}}_t, \mathrm{g}_t\rangle$$

Taking expectation with respect to the randomness of the last stochastic gradient, we have that the term $\langle\mathrm{g}_t - \nabla f(\mathrm{w}_t), \nabla f(\mathrm{w}_t)\rangle = 0$ by $\mathbb{E}\left[\mathrm{g}_t\right] = \nabla f(\mathrm{w}_t)$. Moreover, the term $\mathbb{E}\left[\mathrm{g}_t - \nabla f(\mathrm{w}_t)\right]^2 \leq \sigma^2$. We expand the last term as,

$$\langle\mathrm{w}^* - \tilde{\mathrm{w}}_t, \nabla f(\mathrm{w}_t)\rangle = \langle\mathrm{w}^* - \mathrm{w}_t, \nabla f(\mathrm{w}_t)\rangle + \langle\mathrm{w}_t - \tilde{\mathrm{w}}_t, \nabla f(\mathrm{w}_t)\rangle$$

The first term is bounded by μ-strong convexity as,

$$\begin{aligned} & f(\mathrm{w}^*) \geq f(\mathrm{w}_t) + \langle \nabla f(\mathrm{w}_t), \mathrm{w}^* - \mathrm{w}_t \rangle + \frac{\mu}{2} \left\| \mathrm{w}_t - \mathrm{w}^* \right\|^2 \\ & \iff \langle \nabla f(\mathrm{w}_t), \mathrm{w}^* - \mathrm{w}_t \rangle \leq f(\mathrm{w}^*) - f(\mathrm{w}_t) - \frac{\mu}{2} \left\| \mathrm{w}_t - \mathrm{w}^* \right\|^2 \\ & \leq -\epsilon_t + \frac{\mu}{2} \mu \left\| \tilde{\mathrm{w}}_t - \mathrm{w}_t \right\|^2 - \frac{\mu}{4} \left\| \mathrm{w}^* - \tilde{\mathrm{w}}_t \right\|^2 \end{aligned}$$

where in the last step, we define $\epsilon_t := f(\mathrm{w}_t) - f(\mathrm{w}^*)$ and use $\left\| \mathrm{u} + \mathrm{v} \right\|^2 \leq 2(\left\| \mathrm{u} \right\|^2 + \left\| \mathrm{v} \right\|^2)$. The second term is bounded by using $2 \left\langle \mathrm{u}, \mathrm{v} \right\rangle \leq a \left\| \mathrm{u} \right\|^2 + \frac{1}{a} \left\| \mathrm{v} \right\|^2$ as follows,

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

Moreover, from Fact 2, we have $\left\| \nabla f(\mathrm{w}_t) \right\|^2 \leq 2L\epsilon_t$. Taking expectation and putting everything together, we get,

$$\begin{aligned} \mathbb{E} \left[\left\| \tilde{\mathrm{w}}_{t+1} - \mathrm{w}^* \right\|^2 \right] \leq & \left(1 - \frac{\mu \eta_t}{2} \right) \mathbb{E} \left[\left\| \tilde{\mathrm{w}}_t - \mathrm{w}^* \right\|^2 \right] + \eta_t^2 \sigma^2 \\ & + (\mu + 2L) \eta_t \mathbb{E} \left[\left\| \mathrm{w}_t - \tilde{\mathrm{w}}_t \right\|^2 \right] + \left(2L\eta_t^2 - \eta_t \right) \epsilon_t \end{aligned}$$

We now claim that the last term $2L\eta_t^2 - \eta_t \leq -\frac{\xi - 2}{\xi} \eta_t$ or equivalently $2L\eta_t^2 - \left(1 - \frac{\xi - 2}{\xi} \right) \eta_t \leq 0$. Note that this is a quadratic in η_t which is satisfied between its roots 0 and $\frac{1}{L\xi}$. So it suffices to show is that our step sizes are in this range. In particular, the second root (which is positive by choice of ξ) should be no less than step size. We have $\eta_t = \frac{1}{\mu(t+\xi)}$, $\eta_t \leq \frac{1}{\mu\xi} \; \forall \; t$, the second root $\frac{1}{L\xi} \geq \frac{1}{\mu\xi}$ because smoothness parameter $L \geq \mu$, the strong convexity parameter, or equivalently the condition number $\kappa := L/\mu \geq 1$. Combining the above with $\mathrm{a}_t = \mathrm{w}_t - \tilde{\mathrm{w}}_t$, we get,

$$\begin{aligned} \mathbb{E} \left[\left\| \tilde{\mathrm{w}}_{t+1} - \mathrm{w}^* \right\|^2 \right] \leq & \left(1 - \frac{\mu \eta_t}{2} \right) \mathbb{E} \left[\left\| \tilde{\mathrm{w}}_t - \mathrm{w}^* \right\| \right] + \eta_t^2 \sigma^2 \\ & + (\mu + 2L) \eta_t \mathbb{E} \left[\left\| \mathrm{a}_t \right\|^2 \right] - \left(1 - \frac{2}{\xi} \right) \eta_t \epsilon_t \end{aligned}$$

□

Fact 1. *Rakhlin et al. [2012] Let $f : \mathbb{R}^d \to$ be a μ-strongly convex function, and w^* be its minima. Let g be an unbiased stochastic gradient at point w such that $\mathbb{E} \left[\left\| \mathrm{g} \right\|^2 \right] \leq G^2$, then*

$$\mathbb{E} \left[\left\| \mathrm{w} - \mathrm{w}^* \right\|^2 \right] \leq \frac{4G^2}{\mu^2}$$

Fact 2. *For L-smooth convex function f with minima w^*, then the following holds for all points w,*

$$\left\| \nabla f(\mathrm{w}) - \nabla f(\mathrm{w}^*) \right\|^2 \leq 2L(f(\mathrm{w}) - f(\mathrm{w}^*))$$

Fact 3. *Stich et al. [2018] Let $\{b_t\}_{t \geq 0}, b_t \geq 0$ and $\{\epsilon_t\}_{t \geq 0}, \epsilon_t \geq 0$ be sequences such that,*

$$b_{t+1} \leq \left(1 - \frac{\mu \eta_t}{2} \right) b_t - \epsilon_t \eta_t + A\eta^2 + B\eta^3$$

for constants $A, B > 0, \mu \geq 0, \xi > 1$. Then,

$$\frac{1}{Q_T} \sum_{t=0}^{T-1} q_t \epsilon_t \leq \frac{\mu \xi^3 b_0}{8 Q_T} + \frac{4T(T + 2\xi)A}{\mu Q_T} + \frac{64TB}{\mu^2 Q_T}$$

for $\eta_t = \frac{8}{\mu(\xi + t)}, q_t = (\xi + t)^2, Q_T = \sum_{t=0}^{T-1} q_t \geq \frac{T^3}{3}$

Fact 4. *Stich et al. [2018] Let $\{h_t\}_{t>0}$ be a sequence satisfying $h_0 = 0$ and*

$$h_{t+1} \leq \min\left\{(1-\tau/2)\, h_t + \frac{2}{\tau_k}\eta_t^2 A, (t+1)\sum_{i=0}^{t}\eta_i^2 A\right\}$$

for constant $A > 0$, then with $\eta_t = \frac{1}{t+\xi}$ with $\xi > 1 + \frac{1+\beta}{\tau_k(1+\rho)}$, with $\beta > 4$ and $\rho = \frac{4\beta}{(\beta-4)(\beta+1)^2}$, for $t \geq 0$ we get,

$$h_t \leq \frac{4\beta}{(\beta-4)} \cdot \frac{\eta_t^2 A}{\tau_k^2}$$

Lemma 3. *With probability at least $1-\delta$*

$$\mathbb{E}\left[\left\|\mathrm{a}_t\right\|^2\right] \leq \frac{4\beta}{(\beta-4)} \cdot \frac{\eta_t^2 G^2}{\tau_k^2}$$

Proof of Lemma 3. The proof repeats the steps in Stich et al. [2018] with minor modifications. In particular, the compression is provided by the recovery guarantees of Count Sketch, and we do a union bound over all its instances. We write the proof in full for the sake of completeness. Note that

$$\mathrm{a}_t = \mathrm{a}_{t-1} + \eta_{t-1}\mathrm{g}_{t-1} - \tilde{\mathrm{g}}_{t-1}$$

We first claim that $\mathbb{E}\left[\left\|\mathrm{a}_t\right\|^2\right] \leq t\eta_t^2 G^2$. Since $\mathrm{a}_0 = 0$, we have $\mathrm{a}_t = \sum_{i=1}^{t}(\mathrm{a}_i - \mathrm{a}_{i-1}) = \sum_{i=0}^{t-1}(\eta_i \mathrm{g}_i - \tilde{\mathrm{g}}_i)$. Using $(\sum_{i=1}^{n} a_i)^2 \leq (n+1)\sum_{i=1}^{n} a_i^2$ and taking expectation, we have

$$\mathbb{E}\left[\left\|\mathrm{a}_t\right\|^2\right] \leq t\sum_{i=0}^{t-1}\mathbb{E}\left[\left\|\eta_i \mathrm{g}_i - \tilde{\mathrm{g}}_i\right\|^2\right] \leq t\sum_{i=0}^{t-1}\eta_i^2 G^2$$

Also, from the guarantee of Count Sketch, we have that, with probability at least $1 - \delta/T$, the following holds give that our compression is a τ_k contraction.

Therefore

$$\left\|\mathrm{a}_{t+1}\right\|^2 \leq (1-\tau_k)\left\|\mathrm{a}_t + \eta_t \mathrm{g}_t\right\|^2$$

Using inequality $(a+b)^2 \leq (1+\gamma)a^2 + (1+\gamma^{-1})b^2, \gamma > 0$ with $\gamma = \frac{\tau_k}{2}$, we get

$$\begin{aligned}\left\|\mathrm{a}_{t+1}\right\|^2 &\leq \tau_k\left((1+\gamma)\left\|\mathrm{a}_t\right\|^2 + \left(1+\gamma^{-1}\right)\eta_t^2\left\|\mathrm{g}_t\right\|^2\right) \\ &\leq \frac{(2-\tau_k)}{2}\left\|\mathrm{a}_{t-1}\right\|^2 + \frac{2}{\tau_k}\eta_t^2\left\|\mathrm{g}_t\right\|^2\end{aligned}$$

Taking expectation on the randomness of the stochastic gradient oracle, and using $\mathbb{E}\left[\left\|\mathrm{g}_t\right\|^2\right] \leq G^2$, we have,

$$\mathbb{E}\left[\left\|\mathrm{a}_{t+1}\right\|^2\right] \leq \frac{(2-\tau_k)}{2}\mathbb{E}\left[\left\|\mathrm{a}_t\right\|^2\right] + \frac{2}{\tau_k}\eta_t^2 G^2$$

Note that for a fixed $t \leq T$ this recurrence holds with probability at least $1 - \delta/T$. Using a union bound, this holds for all $t \in [T]$ with probability at least $1-\delta$. Conditioning on this and using Fact 4 completes the proof. □

B Auxiliary results

We state the result of Stich et al. [2018] in full here.

Fact 5 ([Stich et al., 2018]). *Let $f : \mathbb{R}^d \to \mathbb{R}$ be a L-smooth μ-strongly convex function. Given $T > 0$ and $0 < k \leq d$, sparsified SGD with step size $\eta_t = \frac{1}{t+\xi}$, with $\xi > 1 + \frac{d(1+\beta)}{k(1+\rho)}$, with $\beta > 4$ and $\rho = \frac{4\beta}{(\beta-4)(\beta+1)^2}$, after T steps outputs $\hat{\mathrm{w}}_T$:*

$$\mathbb{E}\left[f(\hat{\mathrm{w}}_T)\right] - f(\mathrm{w}^*) \leq \mathcal{O}\left(\frac{G^2}{\mu T} + \frac{d^2G^2L}{k^2\mu^2T^2} + \frac{d^3G^3}{k^3\mu T^3}\right).$$

We now state theorem which uses on the norm bound on stochastic gradients. It follows by directly plugging the fact the HEAVYMIX is a k/d-contraction in the result of Stich et al. [2018].

Theorem 3. *Let $f : \mathbb{R}^d \to \mathbb{R}$ be a L-smooth μ-strongly convex function . Given $T > 0$ and $0 < k \leq d, 0 < \delta < 1$, Algorithm 2 on one machine, with access to stochastic gradients such that $\mathbb{E}\left[\|\mathrm{g}\|^2\right] \leq G^2$, with sketch size $\mathcal{O}\left(k \log(dT/\delta)\right)$ and step size $\eta_t = \frac{1}{t+\xi}$, with $\xi > 1 + \frac{d(1+\beta)}{k(1+\rho)}$, with $\beta > 4$ and $\rho = \frac{4\beta}{(\beta-4)(\beta+1)^2}$, after T steps outputs $\hat{\mathrm{w}}_T$ such that with probability at least $1-\delta$:*

$$\mathbb{E}\left[f(\hat{\mathrm{w}}_T)\right] - f(\mathrm{w}^*) \leq \mathcal{O}\left(\frac{G^2}{\mu T} + \frac{d^2G^2L}{k^2\mu^2T^2} + \frac{d^3G^3}{k^3\mu T^3}\right).$$

Theorem 4 ((non-convex, smooth)). *Let $\{\mathrm{w}_t\}_{t\geq 0}$ denote the iterates of Algorithm 2 one one machine, on an L-smooth function $f : \mathbb{R}^d \to \mathbb{R}$. Assume the stochastic gradients g satisfy $\mathbb{E}[\mathrm{g}] = \nabla f(\mathrm{w})$ and $\mathbb{E}[\|\mathrm{g}\|_2^2] \leq G^2$, and use a sketch of size $\mathcal{O}(k\log(dT/\delta))$, for $0 \leq \delta \leq 1$. Then, setting $\eta = 1/\sqrt{T+1}$ with probability at least $1-\delta$:*

$$\min_{t\in[T]} \|\nabla f(\mathrm{w}_t)\|^2 \leq \frac{2f_0}{\sqrt{(T+1)}} + \frac{LG^2}{2\sqrt{T+1}} + \frac{4L^2G^2(1-k/d)}{(k/d)^2(T+1)},$$

where $f_0 = f(\mathrm{w}_0) - f^\star$.

Theorem 5 ((convex, non-smooth)). *Let $\{\mathrm{w}_t\}_{t\geq 0}$ denote the iterates of Algorithm 2 one one machine, on a convex function $f : \mathbb{R}^d \to \mathbb{R}$. Define $\bar{\mathrm{w}}_\mathrm{t} = \frac{1}{T}\sum_{t=0}^{T} \mathrm{w}_t$. Assume the stochastic gradients g satisfy $\mathbb{E}[\mathrm{g}] = \nabla f(\mathrm{w})$ and $\mathbb{E}[\|\mathrm{g}\|_2^2] \leq G^2$, and use a sketch of size $\mathcal{O}(k\log(dT/\delta))$, for $0 \leq \delta \leq 1$. Then, setting $\eta = 1/\sqrt{T+1}$, with probability at least $1-\delta$:*

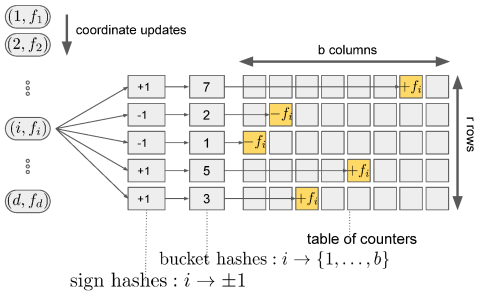
$$\mathbb{E}[f(\bar{\mathrm{w}}_\mathrm{t}) - f^\star] \leq \frac{\|\mathrm{w}_0 - \mathrm{w}^\star\|^2}{\sqrt{(T+1)}} + \left(1 + \frac{2\sqrt{1-k/d}}{k/d}\right)\frac{G^2}{\sqrt{T+1}}.$$

Our high probability bounds of Theorem 2 can be converted to bounds in expectation, stated below.

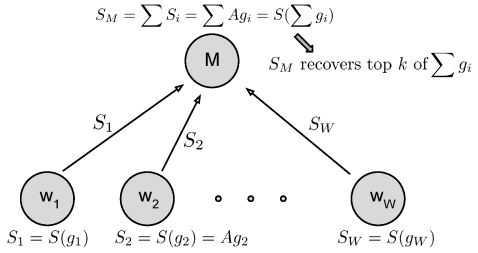
Theorem 6. *Let $f : \mathbb{R}^d \to \mathbb{R}$ be a L-smooth μ-strongly convex function. Given $T > 0$ and $0 < k \leq d, 0 < \delta < 1$, and a τ_k-contraction, Algorithm 2 one one machine, with sketch size $\mathcal{O}\left(k\log(dT/\delta)\right)$ and step size $\eta_t = \frac{1}{t+\xi}$, with $\xi > 1 + \frac{1+\beta}{\tau_k(1+\rho)}$, with $\beta > 4$ and $\rho = \frac{4\beta}{(\beta-4)(\beta+1)^2}$ and $\delta = \mathcal{O}\left(\frac{k}{poly(d)}\right)$ after T steps outputs $\hat{\mathrm{w}}_T$ such that*

$$\mathbb{E}_{\mathcal{A}}\mathbb{E}\left[f(\hat{\mathrm{w}}_T)\right] - f(\mathrm{w}^*) \leq \mathcal{O}\left(\frac{\sigma^2}{\mu T} + \frac{G^2L}{\tau_k^2\mu^2T^2} + \frac{G^3}{\tau_k^3\mu T^3}\right)$$

Proof. Lemma 1 gives that with probability at least $1-\delta$, HEAVYMIX is a k/d contraction. We leverage the fact that the elements of countsketch matrix are bounded to convert it to bound in expectation. As in the proof of lemma 1, given $\mathrm{g} \in \mathbb{R}$, the HEAVYMIX algorithm extracts all $(1/k, \ell_2^2)$-heavy elements from a Count Sketch S of g. Let $\hat{\mathrm{g}}$ be the values of all elements recovered from its sketch. For a fixed k, we create two sets H (heavy), and NH (not-heavy). All coordinates of $\hat{\mathrm{g}}$ with values at least $\frac{1}{k}\hat{\ell}_2^2$ are put in H, and all others in NH, where $\hat{\ell}_2$ is the estimate of $\|\mathrm{g}\|_2$



(a) Low level intuition behind the update step of the Count Sketch.



(b) Property of mergeability lets the parameter server approximate the heavy coordinates of the aggregate vector

from the Count Sketch. For a τ_k contraction with probability at least $1 - \delta$, we get the following expectation bound.

$$\begin{aligned}\mathbb{E}_{\mathcal{A}}\mathbb{E}\left[\left\|\mathrm{g} - \bar{\mathrm{g}}\right\|^2\right] &\leq (1-\delta)\left(1-\tau_k\right)\left\|\mathrm{g}\right\|^2 + \delta\mathcal{O}\left(\mathrm{poly}(d)\right)\left\|\mathrm{g}\right\|^2 \\ &\leq \left(1 - \frac{\tau_k}{2}\right)\left\|\mathrm{g}\right\|^2\end{aligned}$$

where the last time follows because we choose $\delta = \frac{\tau_k}{2\mathcal{O}(\mathrm{poly}(d))}$. Since HEAVYMIX is a k/d contraction, we get the expectation bound of $k/2d$ with $\delta = \frac{k}{2d\mathcal{O}(\mathrm{poly}(d))}$

□

C Sketching

Sketching gained its fame in the streaming model [Muthukrishnan et al., 2005]. A seminal paper by Alon et al. [1999] formalizes the model and delivers a series of important results, among which is the ℓ_2-norm sketch (later referred to as the AMS sketch). Given a stream of updates (a_i, w_i) to the d dimensional vector g (i.e. the i-th update is g_{a_i}+= w_i), the AMS sketch initializes a vector of random signs: $s = (s_j)_{j=1}^d, s_j = \pm 1$. On each update (a_i, w_i), it maintains the running sum S += $s_{a_i} w_i$, and at the end it reports S^2. Note that, if s_j are at least 2-wise independent, then $E(S^2) = E(\sum_i \mathrm{g}_i s_i)^2 = \sum_i \mathrm{g}_i^2 = \|\mathrm{g}\|_2^2$. Similarly, the authors show that 4-wise independence is enough to bound the variance by $4\|\mathrm{g}\|_2^2$. Averaging over independent repetitions running in parallel provides control over the variance, while the median filter (i.e. the majority vote) controls the probability of failure. Formally, the result can be summarized as follows: AMS sketch, with a large constant probability, finds $\hat{\ell}_2 = \|\mathrm{g}\|_2 \pm \varepsilon\|\mathrm{g}\|_2$ using only $\mathcal{O}\left(\frac{1}{\varepsilon^2}\right)$ space. Note that one does not need to explicitly store the entire vector s, as its values can be generated on thy fly using 4-wise independent hashing.

Definition 3. *Let* $\mathrm{g} \in \mathbb{R}^d$. *The* i*-th coordinate* g_i *of* g *is an* (α_1, ℓ_2)*-heavy hitter if* $|\mathrm{g}_i| \geq \alpha_1\|\mathrm{g}\|_2$. g_i *is an* (α_2, ℓ_2^2)*-heavy hitter if* $\mathrm{g}_i^2 \geq \alpha_2\|\mathrm{g}\|_2^2$.

The AMS sketch was later extended by Charikar et al. [2002] to detect heavy coordinates of the vector (see Definition 3). The resulting Count Sketch algorithm hashes the coordinates into b buckets, and sketches the ℓ_2 norm of each bucket. Assuming the histogram of the vector values is skewed, only a small number of buckets will have relatively large ℓ_2 norm. Intuitively, those buckets contain the heavy coordinates and therefore all coordinates hashed to other buckets can be discarded. Repeat the same routine independently and in parallel $\mathcal{O}\left(\log_b d\right)$ times, and all items except the heavy ones will be excluded. Details on how to combine proposed hashing and ℓ_2 sketching efficiently are presented in Figure 5a and Algorithm 4.

Count Sketch finds all (α, ℓ_2)-heavy coordinates and approximates their values with error $\pm\varepsilon\|\mathrm{g}\|_2$. It does so with a memory footprint of $\mathcal{O}\left(\frac{1}{\varepsilon^2\alpha^2}\log d\right)$. We are more interested in finding (α, ℓ_2^2)-heavy hitters, which, by an adjustment to Theorem 7, the Count Sketch can approximately find with a space complexity of $\mathcal{O}\left(\frac{1}{\alpha}\log d\right)$, or $\mathcal{O}\left(k \log d\right)$ if we choose $\alpha = \mathcal{O}\left(\frac{1}{k}\right)$.

Both the Count Sketch and the Count-Min Sketch, which is a similar algorithm presented by Cormode and Muthukrishnan [2005] that achieves a $\pm\varepsilon\ell_1$ guarantee, gained popularity in distributed systems primarily due to the mergeability property formally defined by Agarwal et al. [2013]: given a sketch $S(f)$ computed on the input vector f and a sketch $S(g)$ computed on input g, there exists a function F, s.t. $F(S(f), S(g))$ has the same approximation guarantees and the same memory footprint as $S(f+g)$. Note that sketching the entire vector can be rewritten as a linear operation $S(f) = Af$, and therefore $S(f+g) = S(f) + S(g)$. We take advantage of this crucial property in SKETCHED-SGD, since, on the parameter server, the sum of the workers' sketches is identical to the sketch that would have been produced with only a single worker operating on the entire batch.

Besides having sublinear memory footprint and mergeability, the Count Sketch is simple to implement and straight-forward to parallellize, facilitating GPU acceleration [Ivkin et al., 2018].

Charikar et al. [2002] define the following approximation scheme for finding the list T of the top-k coordinates: $\forall i \in [d] : i \in T \Rightarrow \mathrm{g}_i \geq (1-\varepsilon)\theta$ and $\mathrm{g}_i \geq (1+\varepsilon)\theta \Rightarrow i \in T$, where θ is chosen to be the k-th largest value of f.

Theorem 7 (Charikar et al., 2002)**.** *Count Sketch algorithm finds approximate top-k coordinates with probability at least $1-\delta$, in space $O\left(\log\frac{d}{\delta}\left(k + \frac{\|\mathrm{g}^{tail}\|_2^2}{(\varepsilon\theta)^2}\right)\right)$, where $\|\mathrm{g}^{tail}\|_2^2 = \sum_{i \notin top\ k} \mathrm{g}_i^2$ and θ is the k-th largest coordinate.*

Note that, if $\theta = \alpha\|\mathrm{g}\|_2$, Count Sketch finds all (α, ℓ_2)-heavy coordinates and approximates their values with error $\pm\varepsilon\|\mathrm{g}\|_2$. It does so with a memory footprint of $\mathcal{O}\left(\frac{1}{\varepsilon^2\alpha^2}\log d\right)$.

Algorithm 4 Count Sketch [Charikar et al., 2002]

function init(r, c):
  init sign hashes $\{s_j\}_{j=1}^r$ and bucket hashes $\{h_j\}_{j=1}^r$
  init $r \times c$ table of counters S
function update(i, f_i):
  for j in $1 \ldots r$:
    $S[j, h_j(i)]$ += $s_j(i) f_i$
function estimate(i):
  init length r array estimates
  for j in $1, \ldots, r$:
    estimates$[r] = s_j(i) S[j, h_j(i)]$
  return median(estimates)

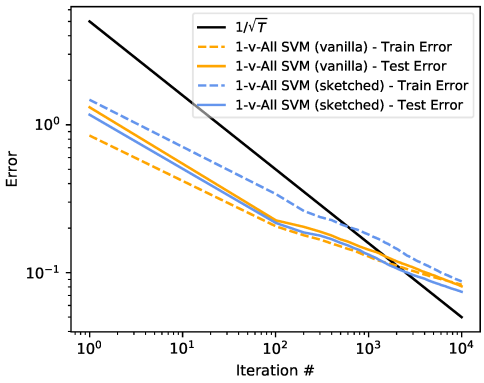
D Model Training Details

We train three models on two datasets. For the first two models, we use code from the OpenNMT project Klein et al. [2017], modified only to add functionality for SKETCHED-SGD. The command to reproduce the baseline transformer results is

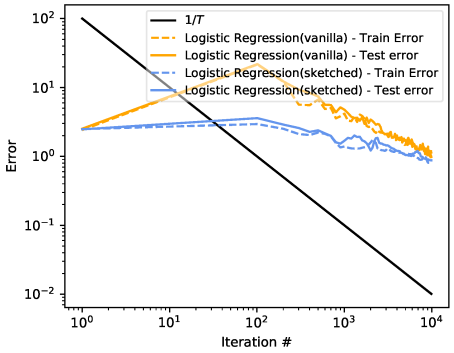
```
python  train.py -data $DATA_DIR -save_model baseline -world_size 1
        -gpu_ranks 0 -layers 6 -rnn_size 512 -word_vec_size 512
        -batch_type tokens -batch_size 1024 -train_steps 60000
        -max_generator_batches 0 -normalization tokens -dropout 0.1
        -accum_count 4 -max_grad_norm 0 -optim sgd -encoder_type transformer
        -decoder_type transformer -position_encoding -param_init 0
        -warmup_steps 16000  -learning_rate 1000 -param_init_glorot
        -momentum 0.9 -decay_method noam -label_smoothing 0.1
        -report_every 100 -valid_steps 100
```

The command to reproduce the baseline LSTM results is

```
python  train.py -data $DATA_DIR -save_model sketched -world_size 1
        -gpu_ranks 0 -layers 6 -rnn_size 512 -word_vec_size 512
        -batch_type tokens -batch_size 1024 -train_steps 60000
        -max_generator_batches 0 -normalization tokens -dropout 0.1
        -accum_count 4 -max_grad_norm 0 -optim sgd -encoder_type rnn
```



(a) log-log plot of training and test error against number of iterations of the average iterate for SVM trained on one class as positive and the rest as negative (1-v-all). For simplicity, we only show the plot for one class.



(b) log-log plot of training and test error of the number of iterations for regularized logistic regression. The regularization parameter was fixed as 0.01.

```
-decoder_type rnn -rnn_type LSTM -position_encoding -param_init 0
-warmup_steps 16000 -learning_rate 8000 -param_init_glorot
-momentum 0.9 -decay_method noam -label_smoothing 0.1
-report_every 100 -valid_steps 100
```

We run both models on the WMT 2014 English to German translation task, preprocessed with a standard tokenizer and then shuffled.

The last model is a residual network trained on CIFAR-10. We use the model from the winning entry of the DAWNBench competition in the category of fastest training time on CIFAR-10 Coleman et al. [2017]. We train this model with a batch size of 512, a learning rate varying linearly at each iteration from 0 (beginning of training) to 0.4 (epoch 5) back to 0 (end of training). We augment the training data by padding images with a 4-pixel black border, then cropping randomly back to 32x32, making 8x8 random black cutouts, and randomly flipping images horizontally. We use a cross-entropy loss with L2 regularization of magnitude 0.0005.

Each run is carried out on a single GPU – either a Titan X, Titan Xp, Titan V, Tesla P100, or Tesla V100.

E Additional experiments

E.1 MNIST

We train vanilla and sketched counterparts of two simple learning models: Support vector machines(SVM) and ℓ_2 regularized logistic regression on MNIST dataset. These are examples of optimizing non-smooth convex function and strongly convex smooth function respectively. We also compare against the theoretical rates obtained in Theorems 5 and 1. The sketch size used in these experiments is size 280 (40 columns and 7 rows), and the parameters k and P are set as, $k = 10, P = 10$, giving a compression of around 4; the number of workers is 4. Figure 5a and 5b shows the plots of training and test errors of these two models. In both the plots, we see that the train and test errors decreases with T in the same rates for vanilla and sketched models. However, these are conservative compared to the theoretical rate suggested.