DRIVE: One-bit Distributed Mean Estimation

Anonymous Author(s)
Affiliation
Address
email

Abstract

We consider the problem where n clients transmit d-dimensional real-valued vectors using $d(1 + o(1))$ bits each, in a manner that allows the receiver to approximately reconstruct their mean. Such compression problems naturally arise in distributed and federated learning. We provide novel mathematical results and derive computationally efficient algorithms that are more accurate than previous compression techniques. We evaluate our methods on a collection of distributed and federated learning tasks, using a variety of datasets, and show a consistent improvement over the state of the art.

1 Introduction

In many computational settings, one wishes to transmit a d-dimensional real-valued vector. For example, in distributed and federated learning scenarios, multiple participants (a.k.a. *clients*) in distributed SGD send gradients to a parameter server that averages them and updates the model parameters accordingly [1]. In these applications and others (e.g., traditional machine learning methods such K-Means and power iteration [2] or other methods such as geometric monitoring [3]), sending *approximations* of vectors may suffice. Moreover, the vectors' dimension d is often large (e.g., in neural networks, d can exceed a billion [4, 5, 6]), so sending compressed vectors is appealing.

Indeed, recent works have studied how to send vector approximations using representations that use a small number of bits per entry (e.g., [2, 7, 8, 9, 10, 11]). Most relevant to our work are solutions that address the distributed mean estimation problem. For example, [2] uses the randomized Hadamard transform followed by stochastic quantization (a.k.a. randomized rounding). When each of the n clients transmits $O(d)$ bits, their Normalized Mean Squared Error (NMSE) is bounded by $O\big(\frac{\log d}{n}\big)$. They also show a $O(\frac{1}{n})$ bound with $O(d)$ bits via variable-length encoding, albeit at a higher computational cost. The sampling method of [10] yields an $O\left(\frac{r \cdot R}{n}\right)$ NMSE bound using $d(1 + o(1))$ bits *in expectation*, where r is each coordinate's representation length and R is the normalized average variance of the sent vectors. Recently, researchers proposed to use Kashin's representation [11, 12, 13]. Broadly speaking, it allows representing a d-dimensional vector in (a higher) dimension $\lambda \cdot d$ for some $\lambda > 1$ using small coefficients. This results in an $O\Big(\frac{\lambda^2}{(\sqrt{\lambda}-1)^4 \cdot n}\Big)$ NMSE bound, where each client transmits $\lambda \cdot d(1 + o(1))$ bits [13].

We step back and focus on approximating d-dimensional vectors using $d(1 + o(1))$ bits (e.g., one bit per dimension and a lower order overhead). We develop novel biased and unbiased compression techniques based on (uniform as well as structured) random rotations in high-dimensional spheres. Intuitively, after a rotation, the coordinates are identically distributed, allowing us to estimate each coordinate with respect to the resulting distribution. Our algorithms do not require expensive operations, such as variable-length encoding or computing the Kashin's representation, and are fast and easy to implement. We obtain an $O\left(\frac{1}{n}\right)$ NMSE bound using $d(1 + o(1))$ bits, regardless of the coordinates' representation length, improving over previous works. Evaluation results indicate that this translates to a consistent improvement over the state of the art in different distributed and federated learning tasks.

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

2 Problem Formulation and Notation

1b - Vector Estimation. We start by formally defining the *1b - vector estimation* problem. A sender, called Alice, gets a real-valued vector $x \in \mathbb{R}^d$ and sends it using a $d(1 + o(1))$ bits message (i.e., asymptotically *one bit per coordinate*). The receiver, called Bob, uses the message to derive an estimate $\widehat{x}$ of the original vector x. We are interested in the quantity $\|x - \widehat{x}\|_2^2$, which is the sum of squared errors (SSE), and its expected value, the Mean Squared Error (MSE). For ease of exposition, we hereafter assume that $x \neq 0$ as this special case can be handled with one additional bit. Our goal is to minimize the *vector*-NMSE (denoted *vNMSE*), defined as the normalized MSE, i.e., $\frac{\mathbb{E}\left[\|x-\widehat{x}\|_2^2\right]}{\|x\|_2^2}$.

1b - Distributed Mean Estimation. The above problem naturally generalizes to the *1b - Distributed Mean Estimation* problem. Here, we have a set of $n \in \mathbb{N}$ *clients* and a *server*. Each client $\mathfrak{c} \in \{1, \ldots, n\}$ has its own vector $x_{(\mathfrak{c})} \in \mathbb{R}^d$, which it sends using a $d(1+o(1))$-bits message to the server. The server then produces an estimate $\widehat{x_{\text{avg}}} \in \mathbb{R}^d$ of the average $x_{\text{avg}} = \frac{1}{n}\sum_{\mathfrak{c}=1}^{n} x_{(\mathfrak{c})}$ with the goal of minimizing its *NMSE*, defined as the average estimate's MSE normalized by the average norm of the clients' original vectors, i.e., $\frac{\mathbb{E}\left[\|x_{\text{avg}}-\widehat{x_{\text{avg}}}\|_2^2\right]}{\frac{1}{n}\cdot\sum_{\mathfrak{c}=1}^{n}\left\|x_{(\mathfrak{c})}\right\|_2^2}$.

Notation. We use the following notation and definitions throughout the paper:

Subscripts. x_i denotes the i'th *coordinate* of the vector x, to distinguish it from client $\mathfrak{c}$'s vector $x_{(\mathfrak{c})}$.

Binary-sign. For a vector $x \in \mathbb{R}^d$, we denote its binary-sign function as $\text{sign}(x)$, where $\text{sign}(x)_i = 1$ if $x_i \geqslant 0$ and $\text{sign}(x)_i = -1$ if $x_i < 0$.

Unit vector. For any (non-zero) real-valued vector $x \in \mathbb{R}^d$, we denote its normalized vector by $\check{x} = \frac{x}{\|x\|_2}$. That is, $\check{x}$ and x has the same direction and it holds that $\|\check{x}\|_2 = 1$.

Rotation Matrix. A matrix $R \in \mathbb{R}^{d\times d}$ is a rotation matrix if $R^T R = I$. The set of all rotation matrices is denoted as $\mathcal{O}(d)$. It follows that $\forall R \in \mathcal{O}(d)$: $det(R) \in \{-1, 1\}$ and $\forall x \in \mathbb{R}^d$: $\|x\|_2 = \|Rx\|_2$.
Random Rotation. A random rotation $\mathcal{R}$ is a distribution over all random rotations in $\mathcal{O}(d)$. For ease of exposition, we abuse the notation and given $x \in \mathbb{R}^d$ denote the random rotation of x by $\mathcal{R}(x) = Rx$, where R is drawn from $\mathcal{R}$. Similarly, $\mathcal{R}^{-1}(x) = R^{-1}x = R^T x$ is the inverse rotation.

Rotation Property. A quantity that determines the guarantees of our algorithms is $\mathcal{L}^d_{\mathcal{R},x} = \frac{\|\mathcal{R}(\check{x})\|_1^2}{d}$ (note the use of the L_1 norm). We show that rotations with high $\mathcal{L}^d_{\mathcal{R},x}$ values yield better estimates.

Shared Randomness. We assume that Alice and Bob have access to shared randomness, e.g., by agreeing on a common PRNG seed. Shared randomness is studied in the context of communication complexity (e.g., [14]) and communication reduction in machine learning systems (e.g., [2, 7]). In our context, it means that Alice and Bob can generate the same random rotations without communication.

3 The DRIVE Algorithm

We start by presenting DRIVE (Deterministically RoundIng randomly rotated VEctors), a novel 1b - Vector Estimation algorithm. Later, we extend DRIVE to the 1b - Distributed Mean Estimation problem. In DRIVE, Alice uses shared randomness to sample a rotation matrix $R \sim \mathcal{R}$ and rotates the vector $x \in \mathbb{R}^d$ by computing $\mathcal{R}(x) = Rx$. It then calculates S, a scalar quantity we explain below. Alice then sends $\big(S, \text{sign}(\mathcal{R}(x))\big)$ to Bob. As we discuss later, sending $\big(S, \text{sign}(\mathcal{R}(x))\big)$ requires $d(1 + o(1))$ bits. In turn, Bob computes $\widehat{\mathcal{R}(x)} = S \cdot \text{sign}(\mathcal{R}(x)) \in \{-S, +S\}^d$. It then uses the shared randomness to generate the same rotation matrix and employs the inverse rotation, i.e., estimates $\widehat{x} = \mathcal{R}^{-1}(\widehat{\mathcal{R}(x)})$. The pseudocode of DRIVE appears in Algorithm 1.

The properties of DRIVE depend on the rotation $\mathcal{R}$ and the *scale parameter* S. We consider both uniform rotations, that provide stronger guarantees, and structured rotations that are orders of magnitude faster to compute. As for the scale $S = S(x, R)$, its exact formula determines the characteristics of DRIVE's estimate, e.g., having minimal vNMSE or being unbiased. The latter allows us to apply DRIVE to the 1b - Distributed Mean Estimation (Section 4.2) and get an NMSE that decreases proportionally to the number of clients.

Algorithm 1 DRIVE

Alice:	**Bob:**
1: Compute $\mathcal{R}(x),\ S$.	1: Compute $\widehat{\mathcal{R}(x)} = S \cdot \text{sign}\big(\mathcal{R}(x)\big)$.
2: Send $\big(S, \text{sign}(\mathcal{R}(x))\big)$ to Bob.	2: Estimate $\widehat{x} = \mathcal{R}^{-1}\big(\widehat{\mathcal{R}(x)}\big)$.

We now prove a general result on the SSE of DRIVE that applies to any random rotation $\mathcal{R}$ and any vector $x \in \mathbb{R}^d$. In the following sections, we use this result to obtain the vNMSE when considering specific rotations and scaling methods as well as analyzing their guarantees.

Theorem 1. *The SSE of DRIVE is:* $\|x - \widehat{x}\|_2^2 = \|x\|_2^2 - 2 \cdot S \cdot \|\mathcal{R}(x)\|_1 + d \cdot S^2$.

Proof. The SSE in estimating $\mathcal{R}(x)$ using $\widehat{\mathcal{R}(x)}$ equals that of estimating x using $\widehat{x}$. Therefore,

$$\|x - \widehat{x}\|_2^2 = \|\mathcal{R}(x - \widehat{x})\|_2^2 = \|\mathcal{R}(x) - \mathcal{R}(\widehat{x})\|_2^2 = \left\|\mathcal{R}(x) - \widehat{\mathcal{R}(x)}\right\|_2^2$$
$$= \|\mathcal{R}(x)\|_2^2 - 2\left\langle \mathcal{R}(x), \widehat{\mathcal{R}(x)} \right\rangle + \left\|\widehat{\mathcal{R}(x)}\right\|_2^2 = \|x\|_2^2 - 2\left\langle \mathcal{R}(x), \widehat{\mathcal{R}(x)} \right\rangle + \left\|\widehat{\mathcal{R}(x)}\right\|_2^2 . \quad (1)$$

Next, we have that,

$$\left\langle \mathcal{R}(x), \widehat{\mathcal{R}(x)} \right\rangle = \sum_{i=1}^d \mathcal{R}(x)_i \cdot \widehat{\mathcal{R}(x)}_i = S \cdot \sum_{i=1}^d \mathcal{R}(x)_i \cdot \text{sign}\big(\mathcal{R}(x)_i\big) = S \cdot \|\mathcal{R}(x)\|_1 \ , \quad (2)$$
$$\left\|\widehat{\mathcal{R}(x)}\right\|_2^2 = \sum_{i=1}^d \widehat{\mathcal{R}(x)}_i^2 = d \cdot S^2 \ . \quad (3)$$

Substituting Eq. (2) and Eq. (3) in Eq. (1) yields the result. □

4 DRIVE With a Uniform Random Rotation

We first consider the thoroughly studied uniform random rotation (e.g., [15, 16, 17, 18, 19]), which we denote by $\mathcal{R}_U$. The sampled matrix is denoted by $R_U \sim \mathcal{R}_U$, that is, $\mathcal{R}_U(x) = R_U \cdot x$. An appealing property of a uniform random rotation is that, as we show later, it admits a scaling that results in a low constant vNMSE even with unbiased estimates.

4.1 1b - Vector Estimation

Using Theorem 1, we obtain the following result. The result holds for any rotation, including $\mathcal{R}_U$.

Lemma 1. *For any $x \in \mathbb{R}^d$, DRIVE's SSE is minimized by $S = \frac{\|\mathcal{R}(x)\|_1}{d}$ (that is, $S = \frac{\|Rx\|_1}{d}$ is determined after $R \sim \mathcal{R}$ is sampled). This yields a vNMSE of $\frac{\mathbb{E}\left[\|x - \widehat{x}\|_2^2\right]}{\|x\|_2^2} = 1 - \mathbb{E}\left[\mathcal{L}_{\mathcal{R},x}^d\right]$.*

Proof. By Theorem 1, to minimize the SSE we require

$$\frac{\partial}{\partial S}\left(\|x\|_2^2 - 2 \cdot S \cdot \|\mathcal{R}(x)\|_1 + d \cdot S^2\right) = -2 \cdot \|\mathcal{R}(x)\|_1 + 2 \cdot d \cdot S = 0 \ ,$$

leading to $S = \frac{\|\mathcal{R}(x)\|_1}{d}$. Then, the SSE of DRIVE becomes:

$$\|x - \widehat{x}\|_2^2 = \|x\|_2^2 - 2 \cdot S \cdot \|\mathcal{R}(x)\|_1 + d \cdot S^2 = \|x\|_2^2 - 2 \cdot \frac{\|\mathcal{R}(x)\|_1^2}{d} + d \cdot \frac{\|\mathcal{R}(x)\|_1^2}{d^2}$$
$$= \|x\|_2^2 - \frac{\|\mathcal{R}(x)\|_1^2}{d} = \|x\|_2^2 - \frac{\|x\|_2^2 \cdot \|\mathcal{R}(\breve{x})\|_1^2}{d} = \|x\|_2^2 \left(1 - \frac{\|\mathcal{R}(\breve{x})\|_1^2}{d}\right) .$$

Thus, the normalized SSE is $\frac{\|x - \widehat{x}\|_2^2}{\|x\|_2^2} = 1 - \mathcal{L}_{\mathcal{R},x}^d$. Taking expectation yields the result. □

Interestingly, for the uniform random rotation, $\mathcal{L}_{\mathcal{R}_U,x}^d$ follows the same distribution for all x. This is because, by the definition of $\mathcal{R}_U$, it holds that $\mathcal{R}_U(\breve{x})$ is distributed uniformly over the unit sphere for any x. Therefore DRIVE's vNMSE depends only on the dimension d. We next analyze the vNMSE attainable by the best possible S, as given in Lemma 1, when the algorithm uses $\mathcal{R}_U$ and is not required to be unbiased. In particular, we state the following theorem whose proof appears in Appendix A.1 (all appendices appear in the Supplementary Material).

Theorem 2. *For any $x \in \mathbb{R}^d$, the vNMSE of DRIVE with $S = \frac{\|\mathcal{R}_U(x)\|_1}{d}$ is $\left(1 - \frac{2}{\pi}\right)\left(1 - \frac{1}{d}\right)$.*

4.2 1b - Distributed Mean Estimation

An appealing property of DRIVE with a uniform random rotation, established in this section, is that with a proper scaling parameter S, the estimate is unbiased. That is, for any $x \in \mathbb{R}^d$, our scale guarantees that $\mathbb{E}\left[\widehat{x}\right] = x$. Unbiasedness is useful when generalizing to the Distributed Mean Estimation problem. Intuitively, when n clients send their vectors, any biased algorithm would result in an NMSE that may not decrease with respect to n. For example, if they have the same input vector, the bias would remain after averaging. Instead, an unbiased encoding algorithm has the property that when all clients act (e.g., use different PRNG seeds) independently, the NMSE decreases proportionally to $\frac{1}{n}$.

Another useful property of uniform random rotation is that its distribution is unchanged when composed with other rotations. We use it in the following theorem's proof, given in Appendix A.2.

Theorem 3. *For any $x \in \mathbb{R}^d$, set $S = \frac{\|x\|_2^2}{\|\mathcal{R}_U(x)\|_1}$. Then DRIVE satisfies $\mathbb{E}[\widehat{x}] = x$.*

Now, we proceed to obtain vNMSE guarantees for DRIVE's unbiased estimate.

Lemma 2. *For any $x \in \mathbb{R}^d$, DRIVE with $S = \frac{\|x\|_2^2}{\|\mathcal{R}_U(x)\|_1}$ has a vNMSE of $\mathbb{E}\left[\frac{1}{\mathcal{L}^d_{\mathcal{R}_U,x}}\right] - 1$.*

Proof. By Theorem 1, the SSE of the algorithm satisfies:

$$\|x\|_2^2 - 2 \cdot S \cdot \|R_U \cdot x\|_1 + d \cdot S^2 = \|x\|_2^2 - 2 \cdot \frac{\|x\|_2^2}{\|R_U \cdot x\|_1} \cdot \|R_U \cdot x\|_1 + d \cdot \left(\frac{\|x\|_2^2}{\|R_U \cdot x\|_1}\right)^2$$

$$= d \cdot \left(\frac{\|x\|_2^2}{\|x\|_2 \|R_U \cdot \check{x}\|_1}\right)^2 - \|x\|_2^2 = d \cdot \frac{\|x\|_2^2}{\|R_U \cdot \check{x}\|_1^2} - \|x\|_2^2 = \|x\|_2^2 \cdot \left(\left(\frac{d}{\|R_U \cdot \check{x}\|_1^2}\right) - 1\right).$$

Normalizing by $\|x\|_2^2$ and taking expectation over R_U concludes the proof. □

Our goal is to derive an upper bound on the above expression and thus upper-bound the vNMSE. Most importantly, we show that even though the estimate is unbiased and we use only a single bit per coordinate, the vNMSE does not increase with the dimension and is bounded by a small constant. In particular, in Appendix A.3, we prove the following:

Theorem 4. *For any $x \in \mathbb{R}^d$, the vNMSE of DRIVE with $S = \frac{\|x\|_2^2}{\|\mathcal{R}_U(x)\|_1}$ satisfies:*

(i) For all $d \geqslant 2$, it is at most 2.92. (ii) For all $d \geqslant 135$, it is at most $\frac{\pi}{2} - 1 + \sqrt{\frac{(6\pi^3 - 12\pi^2) \cdot \ln d + 1}{d}}$.

This theorem yields strong bounds on the vNMSE. For example, the vNMSE is lower than 1 for $d \geqslant 4096$ and lower than 0.673 for $d \geqslant 10^5$. Finally, we obtain the following corollary,

Corollary 1. *For any $x \in \mathbb{R}^d$, the vNMSE tends to $\frac{\pi}{2} - 1 \approx 0.571$ as $d \to \infty$.*

Recall that DRIVE's above scale S is a function of both x and the sampled R_U. An alternative approach is to *deterministically* set S to its mean (by taking an expectation over R_U). As we prove in Appendix A.4, the resulting scale is $\frac{\|x\|_2^2 \cdot (d-1) \cdot \mathrm{B}(\frac{1}{2}, \frac{d-1}{2})}{2d}$, where B is the Beta function. Interestingly, this scale no longer depends on x but only on its norm. In the appendix, we also prove that the resulting vNMSE is bounded by $\frac{\pi}{2} - 1$ *for any* d. In practice, the benefit is marginal, and this approach is not applicable to our improved DRIVE$^+$ algorithm (Section 5) or to other rotations (Section 6).

Finally, with a vNMSE guarantee for the unbiased estimate by DRIVE, we obtain the following key result for the 1b - Distributed Mean Estimation problem, whose proof appears in Appendix A.5. We note that this result guarantees (e.g., see [20]) that distributed SGD, where the participants' gradients are compressed with DRIVE, converges at the same asymptotic rate as without compression.

Theorem 5. *Assume n clients, each with its own vector $x_{(\mathfrak{c})} \in \mathbb{R}^d$. Let each client independently sample $R_{U,\mathfrak{c}} \sim \mathcal{R}_U$ and set its scale to $\frac{\left\|x_{(\mathfrak{c})}\right\|_2^2}{\left\|R_{U,\mathfrak{c}} \cdot x_{(\mathfrak{c})}\right\|_1}$. Then, the server average estimate's NMSE satisfies: $\frac{\mathbb{E}\left[\|x_{avg} - \widehat{x_{avg}}\|_2^2\right]}{\frac{1}{n} \cdot \sum_{\mathfrak{c}=1}^n \left\|x_{(\mathfrak{c})}\right\|_2^2} \leqslant \frac{vNMSE}{n}$, where vNMSE is given by Lemma 2 and is bounded by Theorem 4.*

To the best of our knowledge, DRIVE is the first algorithm with a provable NMSE of $O(\frac{1}{n})$ for the 1b - Distributed Mean Estimation problem (i.e., with $d(1 + o(1))$ bits). In practice, we use only $d + O(1)$ bits to implement DRIVE. We use the $d(1 + o(1))$ notation to ensure compatibility with the theoretical results; see Appendix B for a discussion.

5 Reducing the vNMSE with DRIVE$^+$

To reduce the vNMSE further, we introduce the DRIVE$^+$ algorithm. In DRIVE$^+$, we also use a scale parameter, denoted $S^+ = S^+(x, R)$ to differentiate it from the scale S of DRIVE. Here, instead of reconstructing the rotated vector in a symmetric manner, i.e., $\widehat{\mathcal{R}(x)} \in S \cdot \{-1, 1\}^d$, we have that $\widehat{\mathcal{R}(x)} \in S^+ \cdot \{c_1, c_2\}^d$ where c_1, c_2 are computed using K-Means clustering with $K = 2$ over the d entries of the rotated vector $\mathcal{R}(x)$. That is, c_1, c_2 are chosen to minimize the SSE over any choice of two values. This does not increase the (asymptotic) time complexity over the random rotations considered in this paper as solving K-Means for the special case of one-dimensional data is deterministically solvable in $O(d \log d)$ (e.g., [21]). Notice that DRIVE$^+$ still requires $d(1 + o(1))$ bits as we communicate $(S^+ \cdot c_1, S^+ \cdot c_2)$ and a single bit per coordinate, indicating its nearest centroid. We defer the pseudocode and analyses of DRIVE$^+$ to Appendix C. We show that with proper scaling, for both the 1b - Vector Estimation and 1b - Distributed Mean Estimation problems, DRIVE$^+$ yields guarantees that are at least as strong as those of DRIVE.

6 DRIVE with a Structured Random Rotation

Uniform random rotation generation usually relies on QR factorization (e.g., see [22]), which requires $O(d^3)$ time and $O(d^2)$ space. Therefore, uniform random rotation can only be used in practice to rotate low-dimensional vectors. This is impractical for neural network architectures with many millions of parameters. To that end, we continue to analyze DRIVE and DRIVE$^+$ with the (randomized) Hadamard transform, a.k.a. *structured* random rotation [2, 23], and admits a fast *in-place*, parallelizable, $O(d \log d)$ time implementation [24, 25, 26]. We start with a few definitions.

Definition 1. *The Walsh-Hadamard matrix ([27]) $H_{2^k} \in \{+1, -1\}^{2^k \times 2^k}$ is recursively defined via:* $H_{2^k} = \begin{pmatrix} H_{2^{k-1}} & H_{2^{k-1}} \\ H_{2^{k-1}} & -H_{2^{k-1}} \end{pmatrix}$ *and* $H_1 = (1)$. *Also,* $(\frac{1}{\sqrt{d}} H) \cdot (\frac{1}{\sqrt{d}} H)^T = I$ *and* $det(\frac{1}{\sqrt{d}} H) \in [-1, 1]$.

Definition 2. *Let R_H denote the rotation matrix $\frac{HD}{\sqrt{d}} \in \mathbb{R}^{d \times d}$, where H is a Walsh-Hadamard matrix and D is a diagonal matrix whose diagonal entries are i.i.d. Rademacher random variables (i.e., taking values uniformly in ± 1). Then $\mathcal{R}_H(x) = R_H \cdot x = \frac{1}{\sqrt{d}} H \cdot (x_1 \cdot D_{11}, \ldots, x_d \cdot D_{dd})^T$ is the randomized Hadamard transform of x and $\mathcal{R}_H^{-1}(x) = R_H^T \cdot x = \frac{DH}{\sqrt{d}} \cdot x$ is the inverse transform.*

6.1 1b - Vector Estimation

Recall that the vNMSE of DRIVE, when minimized using $S = \frac{\|\mathcal{R}(x)\|_1}{d}$, is $1 - \mathbb{E}\left[\mathcal{L}^d_{\mathcal{R},x}\right]$ (see Lemma 1). We now bound this quantity of DRIVE with a structured random rotation.

Lemma 3. *For any dimension $d \geqslant 2$ and vector $x \in \mathbb{R}^d$, the vNMSE of DRIVE with a structured random rotation and scale $S = \frac{\|\mathcal{R}_H(x)\|_1}{d}$ is: $1 - \mathbb{E}\left[\mathcal{L}^d_{\mathcal{R}_H,x}\right] \leqslant \frac{1}{2}$.*

Proof. Observe that for all i, $\mathbb{E}\left[|\mathcal{R}_H(x)_i|\right] = \mathbb{E}\left[\left|\sum_{j=1}^d \frac{x_j}{\sqrt{d}} H_{ij} D_{jj}\right|\right]$. Since $\{H_{ij} D_{jj} \mid j \in [d]\}$ are i.i.d. Rademacher random variables we can use the Khintchine inequality [28, 29] which implies that $\frac{1}{\sqrt{2d}} \cdot \|x\|_2 \leqslant \mathbb{E}\left[|\mathcal{R}_H(x)_i|\right] \leqslant \frac{1}{\sqrt{d}} \cdot \|x\|_2$ (see [30, 31] for simplified proofs). We conclude that:

$$\mathbb{E}\left[\mathcal{L}^d_{\mathcal{R}_H,x}\right] = \frac{1}{d} \cdot \mathbb{E}\left[\|\mathcal{R}_H(\breve{x})\|_1^2\right] \geqslant \frac{1}{d} \cdot \mathbb{E}\left[\|\mathcal{R}_H(\breve{x})\|_1\right]^2 \geqslant \frac{1}{d} \cdot \left(\sum_{i=1}^d \frac{1}{\sqrt{2d}}\right)^2 = \frac{1}{2}\,.$$

This bound is sharp since for $d \geqslant 2$ we have that $\mathcal{L}^d_{\mathcal{R}_H,x} = \frac{1}{2}$ for $x = (\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}, 0, \ldots, 0)^T$. □

Observe that unlike for the uniform random rotation, $\mathbb{E}\left[\mathcal{L}^d_{\mathcal{R}_H,x}\right]$ depends on x. We also note that this bound of $\frac{1}{2}$ applies to DRIVE$^+$ (with scale $S^+ = 1$) as we show in Appendix C.2.

6.2 1b - Distributed Mean Estimation

For an arbitrary $x \in \mathbb{R}^d$ and $\mathcal{R}$, and in particular for $\mathcal{R}_H$, the estimates of DRIVE cannot be made unbiased. For example, for $x = (\frac{2}{3}, \frac{1}{3})^T$ we have that $\text{sign}(\mathcal{R}_H(x)) = (D_{11}, D_{11})^T$ and thus $\widehat{\mathcal{R}_H(x)} = S \cdot (D_{11}, D_{11})^T$. This implies that $\widehat{x} = \mathcal{R}_H^{-1}(\widehat{\mathcal{R}_H(x)}) = \frac{1}{\sqrt{2}} \cdot D \cdot H \cdot S \cdot (D_{11}, D_{11})^T = \frac{1}{\sqrt{2}} \cdot S \cdot D \cdot (D_{11}, 0)^T = \frac{1}{\sqrt{2}} \cdot S \cdot (D_{11}^2, 0)^T = (\frac{S}{\sqrt{2}}, 0)^T$. Therefore, $\mathbb{E}[\widehat{x}] \neq x$ regardless of the scale.

Nevertheless, we next provide evidence for why when the input vector is high dimensional and admits finite moments, a structured random rotation performs similarly to a uniform random rotation, yielding all the appealing aforementioned properties. Indeed, it is a common observation that the distribution of machine learning workloads and, in particular, neural network gradients are governed by such distributions (e.g., lognormal [32] or normal [33, 34]).

We seek to show that at high dimensions, the distribution of $\mathcal{R}_H(x)$ is sufficiently similar to that of $\mathcal{R}_U(x)$. By definition, the distribution of $\mathcal{R}_U(x)$ is that of a uniformly at random distributed point on a sphere. Previous studies of this distribution for high dimensions (e.g., [35, 36, 37, 38]) have shown that individual coordinates of $\mathcal{R}_U(x)$ converge to the same normal distribution and that these coordinates are "weakly" dependent in the sense that the joint distribution of every $O(1)$-sized subset of coordinates is similar to that of independent normal variables for large d.

We hereafter assume that $x = (x_1, \ldots, x_d)$, where the x_is are i.i.d. and that $\mathbb{E}[x_j^2] = \sigma^2$ and $\mathbb{E}[|x_j|^3] = \rho < \infty$ for all j. We show that $\mathcal{R}_H(x)_i$ converges to the same normal distribution for all i. Let $F_{i,d}(x)$ be the cumulative distribution function (CDF) of $\frac{1}{\sigma} \cdot \mathcal{R}_H(x)_i$ and Φ be the CDF of the standard normal distribution. The following lemma, proven in Appendix D.1, shows the convergence.

Lemma 4. *For all i, $\mathcal{R}_H(x)_i$ converges to a normal variable:* $\sup_{x \in \mathbb{R}} |F_{i,d}(x) - \Phi(x)| \leqslant \frac{0.409 \cdot \rho}{\sigma^3 \sqrt{d}}$.

With this result, we continue to lay out evidence for the "weak dependency" among the coordinates. We do so by calculating the moments of their joint distribution in increasing subset sizes showing that these moments converge to that of independent normal variables. Previous work has shown that a structured random rotation on vectors with specific distributions results in "weakly dependent" normal variables. This line of research [39, 40, 41] utilized the Hadamard transform for a different purpose. Their goal was to develop a computationally cheap method to generate independent normally distributed variables from simpler (e.g., uniform) distributions. We apply their analysis to our setting.

We partially rely on the following observation that the Hadamard matrix satisfies.

Observation 1. *([39]) The Hadamard product (coordinate-wise product), $H_{\langle i \rangle} \circ H_{\langle \ell \rangle}$, of two rows $H_{\langle i \rangle}, H_{\langle \ell \rangle}$ in the Hadamard matrix yields another row at the matrix $H_{\langle i \rangle} \circ H_{\langle \ell \rangle} = H_{\langle 1+(i-1) \oplus (\ell-1) \rangle}$. Here, $(i-1) \oplus (\ell-1)$ is the bitwise xor of the $(\log d)$-sized binary representation of $(i-1)$ and $(\ell-1)$. It follows that $\sum_{j=1}^{d} H_{ij} H_{\ell j} = \sum_{j=1}^{d} (H_{\langle i \rangle} \circ H_{\langle \ell \rangle})_j = \sum_{j=1}^{d} H_{1+(i-1) \oplus (\ell-1), j}$.*

We now analyze the moments of the rotated variables, starting with the following observation. It follows from the sign-symmetry of D and matches the joint distribution of i.i.d. normal variables.

Observation 2. *All odd moments containing $\mathcal{R}_H(x)$ entries are 0. That is,*

$$\forall q \in \mathbb{N}, \forall i_1, \ldots, i_{2q+1} \in \{1, \ldots, d\} : \mathbb{E}\left[\mathcal{R}_H(x)_{i_1} \cdot \ldots \cdot \mathcal{R}_H(x)_{i_{2q+1}}\right] = 0.$$

Therefore, we need to examine only even moments. We start with showing that the second moments also match with the distribution of independent normal variables.

Lemma 5. *For all $i \neq \ell$ it holds that $\mathbb{E}\left[(\mathcal{R}_H \cdot x)_i \cdot (\mathcal{R}_H \cdot x)_\ell\right] = 0$, whereas $\mathbb{E}\left[(\mathcal{R}_H \cdot x)_i^2\right] = \sigma^2$.*

Proof. Since $\{D_{jj} \mid d \in \{1, \ldots, d\}\}$ are sign-symmetric and i.i.d., $\mathbb{E}\big[(\mathcal{R}_H \cdot x)_i \cdot (\mathcal{R}_H \cdot x)_\ell\big] = \mathbb{E}\big[\frac{1}{d}(\sum_{j=1}^{d} x_j H_{ij} D_{jj}) \cdot (\sum_{j=1}^{d} x_j H_{\ell j} D_{jj})\big] = \mathbb{E}\left[x_j^2\right] \cdot \frac{1}{d} \cdot \sum_{j=1}^{d} H_{ij} H_{\ell j}$. Notice that $\sum_{j=1}^{d} H_{1j} = d$ and $\sum_{j=1}^{d} H_{ij} = 0$ for all $i > 1$. Thus, by Observation 1 we get 0 if $i \neq \ell$ and σ^2 otherwise. $\square$

We have established that the coordinates are pairwise uncorrelated. Similar but more involved analysis shows that the same trend continues under the assumption of the existence of x's higher moments. In Appendix D.2 we analyze the 4th moments showing that they indeed approach the 4th moments of independent normal variables with a rate of $\frac{1}{d}$; the reader is referred to [39] for further intuition and higher moments analysis. We therefore expect that using DRIVE and DRIVE$^+$ with Hadamard transform will yield similar results to that of a uniform random rotation at high dimensions and when the input vectors respect the finite moments assumption.

In addition to the theoretical evidence, in Figure 1, we show experimental results comparing the measured NMSE for the 1b - Distributed Mean Estimation problem with $n = 10$ clients (all given the *same* vector so biases do not cancel out) for DRIVE and DRIVE$^+$ using both uniform and structured random rotations over three different distributions. The results indicate that all variants yield similar NMSEs in reasonable dimensions, in line with the theoretical guarantee of Corollary 1 and Theorem 5.

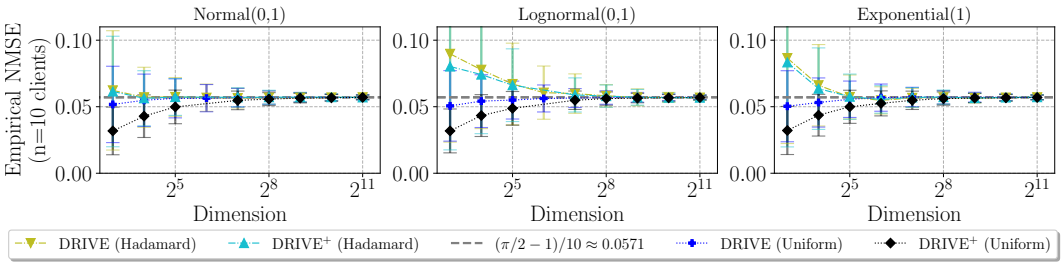

Figure 1: Distributed mean estimation comparison: each data point is averaged over 10^3 trials. In each trial, *the same* (randomly sampled) vector is sent by $n = 10$ clients.

Dimension (d)	Hadamard + 1-Bit SQ	Kashin + 1-Bit SQ	Drive (Uniform)	Drive$^+$ (Uniform)	Drive (Hadamard)	Drive$^+$ (Hadamard)
128	0.5308, *0.34*	0.2550, *2.12*	0.0567, *40.4*	**0.0547**, *40.7*	0.0581, *0.36*	0.0595, *0.72*
8,192	1.3338, *0.57*	0.3180, *3.42*	**0.0571**, *5088*	**0.0571**, *5101*	**0.0571**, *0.60*	**0.0571**, *1.06*
524,288	2.1456, *0.79*	0.3178, *4.69*	—	—	**0.0571**, *0.82*	**0.0571**, *1.35*
33,554,432	2.9332, *27.1*	0.3179, *332*	—	—	**0.0571**, *27.2*	**0.0571**, *37.8*

Table 1: Empirical NMSE and average per-vector encoding time (in milliseconds, on an RTX 3090 GPU) for distributed mean estimation with $n = 10$ clients (same as in Figure 1) and Lognormal(0,1) distribution. Each entry is a (NMSE, *time*) tuple and the most accurate result is highlighted in **bold**.

7 Evaluation

We evaluate DRIVE and DRIVE$^+$, comparing them to standard and recent state-of-the-art techniques. We consider classic distributed learning tasks as well as federated learning tasks (e.g., where the data distribution is not i.i.d. and clients may change over time). All the distributed tasks are implemented over PyTorch [42] and all the federated tasks are implemented over TensorFlow Federated [43]. We focus our comparison on vector quantization algorithms and recent sketching techniques and exclude sparsification methods (e.g., [44, 45, 46, 47]) and methods that involve client-side memory (e.g., error feedback [48, 49]) since these can often work in conjunction with our algorithms.

Datasets. We use MNIST [50], EMNIST [51], CIFAR-10 and CIFAR-100 [52] for image classification tasks; a next-character-prediction task using the Shakespeare dataset [53]; and a next-word-prediction task using the Stack Overflow dataset [54]. Additional details appear in Appendix E.1.

Algorithms. We compare against several alternative algorithms: (1) *FedAvg* [1] that uses the full vectors (i.e., each coordinate is represented using a 32-bit float); (2) Hadamard transform followed by 1-bit stochastic quantization (SQ) [2, 10]; (3) Kashin's representation followed by 1-bit stochastic quantization [11]; (4) *TernGrad* [8], which clips coordinates larger than 2.5 times the standard deviation, then performs 1-bit stochastic quantization on the absolute values and separately sends their signs and the maximum coordinate for scale (we note that TernGrad is a low-bit variant of a well-known algorithm called *QSGD* [9], and we use TernGrad since we found it to perform better in our experiments); and (5-6) *Sketched-SGD* [55] and *FetchSGD* [56], which are both count-sketch [57] based algorithms designed for distributed and federated learning, respectively.

We note that Hadamard with 1-bit stochastic quantization is our most fair comparison, as it uses the same number of bits as DRIVE$^+$ (and slightly more than DRIVE) and has similar computational costs. This contrasts with Kashin's representation, where both the number of bits and the computational costs are higher. For example, a standard TensorFlow Federated implementation (e.g., see "CLASS KASHINHADAMARDENCODINGSTAGE" hyperparameters at [58]) uses a minimum of 1.17 bits per coordinate, and three iterations of the algorithm resulting in five Hadamard transforms for each vector. Also, note that TernGrad uses an extra bit per coordinate for sending the sign. Moreover, the clipping performed by TernGrad is a heuristic procedure, which is orthogonal to our work.

For each task, we use a subset of datasets and the most relevant competition. Detailed configuration information and additional results appear in Appendix E. We next examine vNMSE-Speed tradeoffs, detail the distributed and federated learning experiments, and summarize their results.

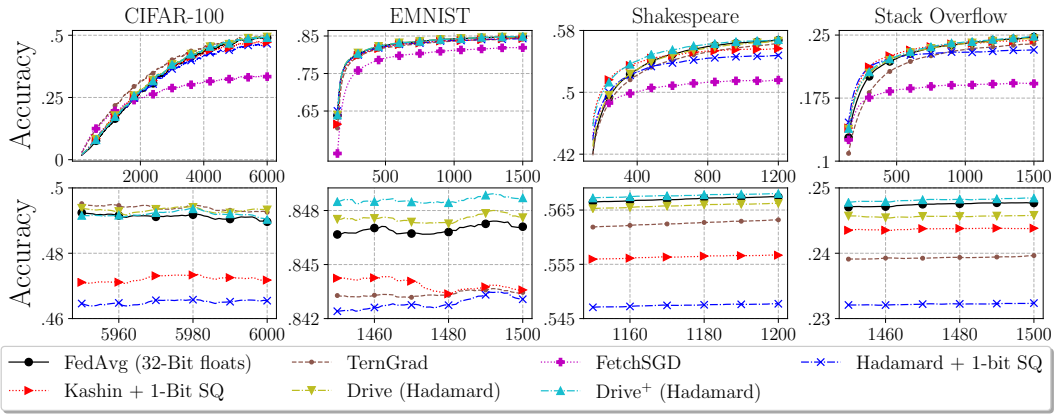

Figure 2: Accuracy per round on various federated learning tasks. Smoothing is done using a rolling mean with a window size of 150. The second row zooms-in on the last 50 rounds.

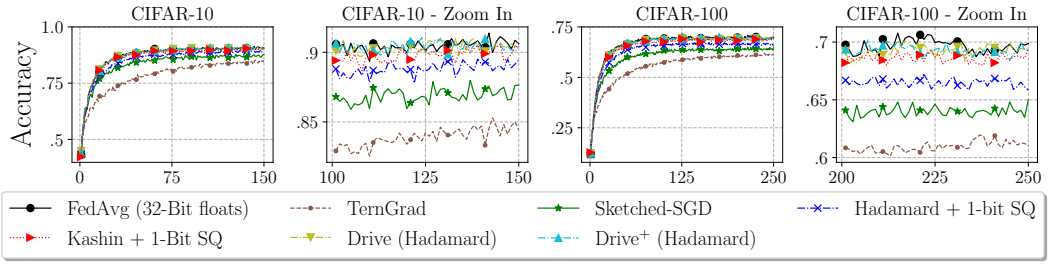

Figure 3: Accuracy per round on distributed learning tasks, with a zoom-in on the last 50 rounds.

vNMSE-Speed Tradeoff. Appearing in Table 1, the results show that our algorithms offer the lowest NMSE and that the gap increases with the dimension. As expected, DRIVE and DRIVE$^+$ with uniform rotation are more accurate for small dimensions but are significantly slower. Similarly, DRIVE is as accurate as DRIVE$^+$, and both are significantly more accurate than Kashin (by a factor of 4.4×-5.5×) and Hadamard (9.3×-51×) with stochastic quantization. Additionally, DRIVE is 5.7×-12× faster than Kashin and about as fast as Hadamard. In Appendix E.2 we discuss the results (including why DRIVE$^+$ may have slightly higher NMSE than DRIVE in certain scenarios), give the complete experiment specification, and provide measurements on a commodity machine.

Federated Learning. We evaluate over four tasks: (1) EMNIST over customized CNN architecture with two convolutional layers with $\approx 1.2M$ parameters [59]; (2) CIFAR-100 over ResNet-18 [52]; (3) a next-character-prediction task using the Shakespeare dataset [1]; (4) a next-word-prediction task using the Stack Overflow dataset [60]. Both (3) and (4) use LSTM recurrent models [61] with $\approx 820K$ and $\approx 4M$ parameters, respectively. We use code, client partitioning, models, hyperparameters, and validation metrics from the federated learning benchmark of [60]. The results are depicted in Figure 2. Additional details and hyperparameter configurations are detailed in Appendix E.3.

Distributed CNN Training. We evaluate distributed CNN training with 10 clients in two configurations: (1) CIFAR-10 dataset with ResNet-9; (2) CIFAR-100 with ResNet-18 [52, 62]. Additional details and hyperparameter configurations are presented in Appendix E.4. Figure 3 depicts the results.

Distributed Power Iteration. Power Iteration is a simple method for approximating the dominant Eigenvalues and Eigenvectors of a matrix. It is often used as a sub-routine in more complicated tasks such as Principal Component Analysis. In our distributed setting, at each epoch, the server broadcasts the current estimated top eigenvector to all clients. Then, each client: (1) updates the top eigenvector based on its local data; (2) computes its diff to the current estimated top eigenvector; (3) compresses the diff vector; (4) sends it to the server, possibly scaled by a *learning rate* to ensure convergence to due the high estimation variance. Figure 4 presents the results in a setting with 10 and 100 clients and a learning rate of 0.1. In Appendix E.5, we present additional results with 1000 clients.

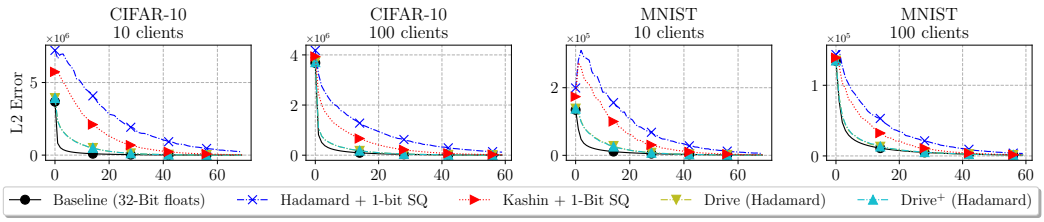

Figure 4: Distributed power iteration. Y-axis is the L2 error (lower is better). X-axis is the epoch.

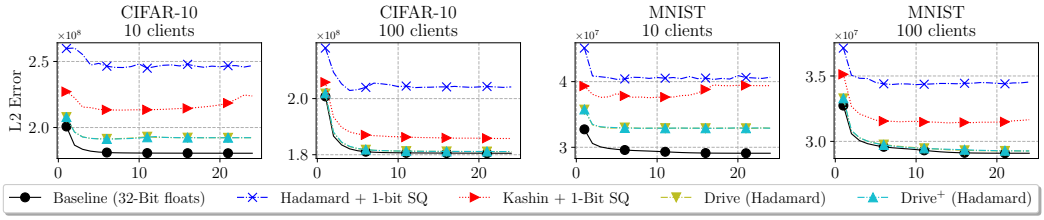

Figure 5: Distributed K-Means. Y-axis is the total L2 error (lower is better). X-axis is the epoch. Note that the Y-axis does not starts at 0 for better readability.

Distributed K-Means. In this distributed learning task, the server coordinates a run of Lloyd's algorithm over a distributed dataset. At each training round, the server broadcasts the centroids' coordinates to the clients. Then, each client: (1) updates these coordinates based on its local data; (2) compresses them; (3) sends them to the server. The centroids weights (i.e., the number of observations assigned to each centroid) require fewer bits and are essential for K-Means convergence quickly; therefore, the weights are sent without compression. Figure 5 presents the results with 10 and 100 clients. In Appendix E.5 we show evaluation results with 1000 clients.

Evaluation Summary. Overall, it is evident that DRIVE and DRIVE$^+$ consistently offer markedly favorable results in comparison to the alternatives in our setting. Kashin's representation appears to offer the best competition, albeit at somewhat higher computational complexity and bandwidth requirements. The lesser performance of the sketch-based techniques is attributed to the high noise of the sketch under such a low ($d(1+o(1))$ bits) communication requirement. This is because the number of counters they can use is too low, making too many coordinates map into each counter.

8 Conclusions

In this paper, we studied the vector and distributed mean estimation problems. These problems are applicable to distributed and federated learning, where clients communicate real-valued vectors (e.g., gradients) to a server for averaging. To the best of our knowledge, our algorithms are the first with a provable error of $O(\frac{1}{n})$ for the 1b - Distributed Mean Estimation problem (i.e., with $d(1 + o(1))$ bits). As shown in [13], any algorithm that uses $O(d)$ shared random bits (e.g., our Hadamard-based variant) has a vNMSE of $\Omega(1)$, i.e., DRIVE and DRIVE$^+$ are asymptotically optimal; additional discussion is given in Appendix F. Our experiments, carried over various tasks and datasets, indicate that our algorithms improve over the state of the art. All the results presented in this paper are fully reproducible by our source code, available at [26].

Our work has several limitations. Unlike previous works, our algorithms' guarantees with more than one bit per coordinate are not established. For example, while it is possible to run DRIVE$^+$ with more than $k = 2$ centroids (e.g., $k = 8$ for $d(3 + o(1))$ bits), it is unclear how to choose the scale parameter to make the estimate unbiased and what is the resulting vNMSE (and thus the NMSE). Further, the distributed mean estimation's analysis of our Hadamard-based variants is based on an assumption (Section 6.2) about the vector distributions. While machine learning workloads, and DNN gradients in particular (e.g., [32, 33, 34]), were observed to follow such distributions, this assumption may not hold for other applications. It is a challenging research question whether there exists other structured rotations with low computational complexity and stronger guarantees.

References

[1] H. Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. Communication-Efficient Learning of Deep Networks from Decentralized Data. In *Artificial Intelligence and Statistics*, pages 1273–1282, 2017.

[2] Ananda Theertha Suresh, X Yu Felix, Sanjiv Kumar, and H Brendan McMahan. Distributed Mean Estimation With Limited Communication. In *International Conference on Machine Learning*, pages 3329–3337. PMLR, 2017.

[3] Yuval Alfassi, Moshe Gabel, Gal Yehuda, and Daniel Keren. A Distance-Based Scheme for Reducing Bandwidth in Distributed Geometric Monitoring. In *2021 IEEE 37th International Conference on Data Engineering (ICDE)*, 2021.

[4] Jeffrey Dean, Greg Corrado, Rajat Monga, Kai Chen, Matthieu Devin, Mark Mao, Marc' aurelio Ranzato, Andrew Senior, Paul Tucker, Ke Yang, Quoc Le, and Andrew Ng. Large Scale Distributed Deep Networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012.

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

[6] Yanping Huang, Youlong Cheng, Ankur Bapna, Orhan Firat, Dehao Chen, Mia Chen, HyoukJoong Lee, Jiquan Ngiam, Quoc V Le, Yonghui Wu, and zhifeng Chen. GPipe: Efficient Training of Giant Neural Networks using Pipeline Parallelism. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[7] Ran Ben-Basat, Michael Mitzenmacher, and Shay Vargaftik. How to Send a Real Number Using a Single Bit (and Some Shared Randomness). *arXiv preprint arXiv:2010.02331*, 2020.

[8] 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.

[9] Dan Alistarh, Demjan Grubic, Jerry Li, Ryota Tomioka, and Milan Vojnovic. QSGD: Communication-Efficient Sgd via Gradient Quantization and Encoding. *Advances in Neural Information Processing Systems*, 30:1709–1720, 2017.

[10] Jakub Konečnỳ and Peter Richtárik. Randomized Distributed Mean Estimation: Accuracy vs. Communication. *Frontiers in Applied Mathematics and Statistics*, 4:62, 2018.

[11] Sebastian Caldas, Jakub Konečný, H Brendan McMahan, and Ameet Talwalkar. Expanding the Reach of Federated Learning by Reducing Client Resource Requirements. *arXiv preprint arXiv:1812.07210*, 2018.

[12] Yurii Lyubarskii and Roman Vershynin. Uncertainty Principles and Vector Quantization. *IEEE Transactions on Information Theory*, 56(7):3491–3501, 2010.

[13] Mher Safaryan, Egor Shulgin, and Peter Richtárik. Uncertainty Principle for Communication Compression in Distributed and Federated Learning and the Search for an Optimal Compressor. *arXiv preprint arXiv:2002.08958*, 2020.

[14] Ilan Newman. Private vs. Common Random Bits in Communication Complexity. *Information processing letters*, 39(2):67–71, 1991.

[15] Francesco Mezzadri. How to Generate Random Matrices From the Classical Compact Groups. *arXiv preprint math-ph/0609050*, 2006.

[16] RWM Wedderburn. Generating Random Rotations. *Research Report*, 1975.

[17] Richard M Heiberger. Generation of Random Orthogonal Matrices. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 27(2):199–206, 1978.

[18] Gilbert W Stewart. The Efficient Generation of Random Orthogonal Matrices With an Application to Condition Estimators. *SIAM Journal on Numerical Analysis*, 17(3):403–409, 1980.

[19] Martin A Tanner and Ronald A Thisted. Remark as r42: A Remark on as 127. Generation of Random Orthogonal Matrices. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 31(2):190–192, 1982.

[20] Aleksandr Beznosikov, Samuel Horváth, Peter Richtárik, and Mher Safaryan. On biased compression for distributed learning. *arXiv preprint arXiv:2002.12410*, 2020.

[21] Allan Grønlund, Kasper Green Larsen, Alexander Mathiasen, Jesper Sindahl Nielsen, Stefan Schneider, and Mingzhou Song. Fast Exact K-Means, K-Medians and Bregman Divergence Clustering in 1D. *arXiv preprint arXiv:1701.07204*, 2017.

[22] Mario Lezcano Casado. GeoTorch's Documentation. A Library for Constrained Optimization and Manifold Optimization for Deep Learning in Pytorch. `https://geotorch.readthedocs.io/en/latest/_modules/geotorch/so.html`, 2021. accessed 17-May-21.

[23] Nir Ailon and Bernard Chazelle. The Fast Johnson–Lindenstrauss Transform and Approximate Nearest Neighbors. *SIAM Journal on computing*, 39(1):302–322, 2009.

[24] Bernard J. Fino and V. Ralph Algazi. Unified matrix treatment of the fast walsh-hadamard transform. *IEEE Transactions on Computers*, 25(11):1142–1146, 1976.

[25] Chunyuan Li, Heerad Farkhoor, Rosanne Liu, and Jason Yosinski. Measuring the intrinsic dimension of objective landscapes. In *International Conference on Learning Representations*, 2018. Code available at: `https://github.com/uber-research/intrinsic-dimension`.

[26] DRIVE open source code, 2021. Currently available in the Supplementary Material. Will be made publicly available upon the acceptance of the paper.

[27] Kathy J Horadam. *Hadamard Matrices and Their Applications*. Princeton university press, 2012.

[28] Aleksandr Khintchine. Über Dyadische Brüche. *Mathematische Zeitschrift*, 18(1):109–116, 1923.

[29] S Szarek. On the Best Constants in the Khinchin Inequality. *Studia Mathematica*, 2(58):197–208, 1976.

[30] Yuval Filmus. Khintchine-Kahane using Fourier Analysis. *Posted at `http://www.cs.toronto.edu/yuvalf/KK.pdf`*, 2012.

[31] Rafał Latała and Krzysztof Oleszkiewicz. On the Best Constant In the Khinchin-Kahane Inequality. *Studia Mathematica*, 109(1):101–104, 1994.

[32] Brian Chmiel, Liad Ben-Uri, Moran Shkolnik, Elad Hoffer, Ron Banner, and Daniel Soudry. Neural Gradients Are Near-Lognormal: Improved Quantized and Sparse Training. *arXiv preprint arXiv:2006.08173*, 2020.

[33] Ron Banner, Yury Nahshan, Elad Hoffer, and Daniel Soudry. Post-Training 4-Bit Quantization of Convolution Networks for Rapid-Deployment. *arXiv preprint arXiv:1810.05723*, 2018.

[34] Xucheng Ye, Pengcheng Dai, Junyu Luo, Xin Guo, Yingjie Qi, Jianlei Yang, and Yiran Chen. Accelerating CNN Training by Pruning Activation Gradients. In *European Conference on Computer Vision*, pages 322–338. Springer, 2020.

[35] Marcus Spruill et al. Asymptotic Distribution of Coordinates on High Dimensional Spheres. *Electronic communications in probability*, 12:234–247, 2007.

[36] Persi Diaconis and David Freedman. A Dozen de Finetti-Style Results in Search of a Theory. In *Annales de l'IHP Probabilités et statistiques*, volume 23, pages 397–423, 1987.

[37] Svetlozar T Rachev, L Ruschendorf, et al. Approximate Independence of Distributions on Spheres and Their Stability Properties. *The Annals of Probability*, 19(3):1311–1337, 1991.

[38] Adriaan J Stam. Limit Theorems for Uniform Distributions on Spheres in High-Dimensional Euclidean Spaces. *Journal of Applied probability*, 19(1):221–228, 1982.

[39] Charles M Rader. A New Method of Generating Gaussian Random Variables by Computer. Technical report, MASSACHUSETTS INST OF TECH LEXINGTON LINCOLN LAB, 1969.

[40] David B Thomas. Parallel Generation of Gaussian Random Numbers Using the Table-Hadamard Transform. In *2013 IEEE 21st Annual International Symposium on Field-Programmable Custom Computing Machines*, pages 161–168. IEEE, 2013.

[41] Tamás Herendi, Thomas Siegl, and Robert F Tichy. Fast Gaussian Random Number Generation Using Linear Transformations. *Computing*, 59(2):163–181, 1997.

[42] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8026–8037. Curran Associates, Inc., 2019.

[43] Google. TensorFlow Federated: Machine Learning on Decentralized Data. `https://www.tensorflow.org/federated`, 2020. accessed 25-Mar-20.

[44] Jakub Konečný, H. Brendan McMahan, Felix X. Yu, Peter Richtárik, Ananda Theertha Suresh, and Dave Bacon. Federated Learning: Strategies for Improving Communication Efficiency, 2017.

[45] Hongyi Wang, Scott Sievert, Shengchao Liu, Zachary B. Charles, Dimitris S. Papailiopoulos, and Stephen Wright. ATOMO: Communication-efficient Learning via Atomic Sparsification. In *NeurIPS*, pages 9872–9883, 2018.

[46] Thijs Vogels, Sai Praneeth Karimireddy, and Martin Jaggi. PowerSGD: Practical Low-Rank Gradient Compression for Distributed Optimization. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32, 2019.

[47] Sebastian U Stich, Jean-Baptiste Cordonnier, and Martin Jaggi. Sparsified SGD with Memory. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

[48] 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.

[49] Sai Praneeth Karimireddy, Quentin Rebjock, Sebastian Stich, and Martin Jaggi. Error Feedback Fixes SignSGD and other Gradient Compression Schemes. In *International Conference on Machine Learning*, pages 3252–3261, 2019.

[50] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-Based Learning Applied to Document Recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[51] Gregory Cohen, Saeed Afshar, Jonathan Tapson, and Andre Van Schaik. EMNIST: Extending MNIST to Handwritten Letters. In *2017 International Joint Conference on Neural Networks (IJCNN)*, pages 2921–2926. IEEE, 2017.

[52] Alex Krizhevsky, Geoffrey Hinton, et al. Learning Multiple Layers of Features From Tiny Images. 2009.

[53] William Shakespeare. The Complete Works of William Shakespeare. `https://www.gutenberg.org/ebooks/100`.

[54] Stack Overflow Data. `https://www.kaggle.com/stackoverflow/stackoverflow`. accessed 01-Mar-21.

[55] Nikita Ivkin, Daniel Rothchild, Enayat Ullah, Vladimir Braverman, Ion Stoica, and Raman Arora. Communication-Efficient Distributed Sgd With Sketching. *arXiv preprint arXiv:1903.04488*, 2019.

[56] Daniel Rothchild, Ashwinee Panda, Enayat Ullah, Nikita Ivkin, Ion Stoica, Vladimir Braverman, Joseph Gonzalez, and Raman Arora. FetchSGD: Communication-Efficient Federated Learning With Sketching. In *International Conference on Machine Learning*, pages 8253–8265. PMLR, 2020.

[57] 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.

[58] The TensorFlow Authors. TensorFlow Federated: Compression via Kashin's representation from Hadamard transform. `https://github.com/tensorflow/model-optimization/blob/9193d70f6e7c9f78f7c63336bd68620c4bc6c2ca/tensorflow_model_optimization/python/core/internal/tensor_encoding/stages/research/kashin.py#L92`. accessed 16-May-21.

[59] Sebastian Caldas, Sai Meher Karthik Duddu, Peter Wu, Tian Li, Jakub Konečný, H. Brendan McMahan, Virginia Smith, and Ameet Talwalkar. LEAF: A Benchmark for Federated Settings, 2019.

[60] Sashank Reddi, Zachary Charles, Manzil Zaheer, Zachary Garrett, Keith Rush, Jakub Konečný, Sanjiv Kumar, and H. Brendan McMahan. Adaptive Federated Optimization, 2020.

[61] S. Hochreiter and J. Schmidhuber. Long Short-Term Memory. *Neural Computation*, 9:1735–1780, 1997.

[62] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[63] Mervin E Muller. A Note on a Method for Generating Points Uniformly on N-Dimensional Spheres. *Communications of the ACM*, 2(4):19–20, 1959.

[64] George Bennett. Probability Inequalities for the Sum of Independent Random Variables. *Journal of the American Statistical Association*, 57(297):33–45, 1962.

[65] Andreas Maurer et al. A Bound on the Deviation Probability for Sums of Non-Negative Random Variables. *J. Inequalities in Pure and Applied Mathematics*, 4(1):15, 2003.

[66] Paweł J Szabłowski. Uniform Distributions on Spheres in Finite Dimensional L_α and Their Generalizations. *Journal of multivariate analysis*, 64(2):103–117, 1998.

[67] Carl-Gustav Esseen. A Moment Inequality With an Application to the Central Limit Theorem. *Scandinavian Actuarial Journal*, 1956(2):160–170, 1956.

[68] Google. TensorFlow Model Optimization Toolkit. `https://www.tensorflow.org/model_optimization`, 2021. accessed 19-Mar-21.

[69] Apache license, version 2.0. `https://www.apache.org/licenses/LICENSE-2.0`. accessed 19-Mar-21.

[70] The TensorFlow Authors. TensorFlow Federated: EncodedSumFactory class. `https://github.com/tensorflow/federated/blob/v0.19.0/tensorflow_federated/python/aggregators/encoded.py#L40-L142`. accessed 19-May-21.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] See Section 8.
   (c) Did you discuss any potential negative societal impacts of your work? [No] We do not believe there is an inherit negative societal impact in this work.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes]
   (b) Did you include complete proofs of all theoretical results? [Yes]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] In the Supplemental Material. It will be released as open-source after publication.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] see Section 7 and Appendix E.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [No] Most experiments can be easily reproduced but require considerable execution time and compute resources. We did include error bars in Figure 1, where it was feasible.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See Section 7 and Appendix E.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] See Section 7 and Appendix E.
   (b) Did you mention the license of the assets? [Yes] See Section 7 and Appendix E.
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes] We include the source code of our algorithms in the Supplemental Material. It will be released as open-source after publication.
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A] We used well-known datasets whose creators use either publicly available data or received proper consent.
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A] We used well-known datasets, which do not contain personally identifiable information or offensive content.
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A] We did not use crowdsourcing or conducted research with human subjects.
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A] We did not use crowdsourcing or conducted research with human subjects.
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A] We did not use crowdsourcing or conducted research with human subjects.