Communication-Efficient Federated Learning via Robust Distributed Mean Estimation

Shay Vargaftik* Ran Ben Basat* Amit Portnoy*
VMware Research University College London Ben-Gurion University

Gal Mendelson Yaniv Ben-Itzhak Michael Mitzenmacher
Stanford University VMware Research Harvard University

Abstract

Federated learning commonly relies on algorithms such as distributed (mini-batch) SGD, where multiple clients compute their gradients and send them to a central coordinator for averaging and updating the model. To optimize the transmission time and the scalability of the training process, clients often use lossy compression to reduce the message sizes. DRIVE [1] is a recent state of the art algorithm that compresses gradients using one bit per coordinate (with some lower-order overhead). In this technical report, we generalize DRIVE to support any bandwidth constraint as well as extend it to support heterogeneous client resources and make it robust to packet loss.

1 Introduction

In the \(b\)-Distributed Mean Estimation (\(b\)-DME) problem, \(n\) clients send \(d\)-dimensional vectors to be averaged by a central coordinator, often named the Parameter Server (PS), while using only \(b\) bits per coordinate. This setting is natural in federated learning, where the PS averages the received clients’ parameter updates (e.g., neural network gradients) and adjusts its model [2]. As the vectors are often large (e.g., in neural networks, \(d\) can exceed a billion [3, 4, 5]), the network is often a bottleneck, and thus compressing the vectors is essential to reduce the training time and allow better scalability (e.g., see [6] and the references therein).

Standard quantization techniques, such as stochastic quantization (a.k.a. randomized rounding), compress real numbers into a finite number of bits while introducing a bounded error. Such approaches have a Mean Squared Error (MSE) that depends on the difference between the minimal and maximal coordinate of the compressed vector. In general, vectors (e.g., neural network gradients) can have large differences in the magnitude of the different coordinates, rendering direct quantization inapplicable for accurate compression. Therefore, previous works suggested to randomly rotate the input vector prior to quantization [7]. That is, the clients and PS draw rotation matrices according to some known distribution; the clients then send the quantization of the rotated vectors while the PS applies the inverse rotation on the estimated rotated vector. In particular, [7] proposed to draw the matrices according to the randomized Hadamard transform, which allows in-place GPU-friendly \(O(d \log d)\)-time vector rotations. Intuitively, the coordinates of a rotated vector are identically distributed, and the expected difference between the coordinates is provably small, thus enabling a more accurate quantization. More precisely, this approach achieves a Normalized MSE (NMSE) of \(O(\frac{1}{n})\) for any bandwidth constraint. The authors have also proposed a variable-length encoding approach that reaches an NMSE of \(O(\frac{1}{n})\) using \(O(d)\) bits. However, it is slower to implement and not always more accurate in practice.

The rotation approach was recently improved using the Kashin representation of vectors [8, 9, 10]. Roughly speaking, it allows representing a \(d\)-dimensional vector using \(O(d)\) coefficients, each smaller than \(O(\frac{1}{\sqrt{d}})\). Therefore, applying stochastic quantization to the Kashin coefficients allows an NMSE of \(O(\frac{1}{n})\) using \(O(d)\) bits. Compared with the Hadamard transform of [7], using Kashin’s representation yields a lower NMSE at the cost of additional computation [11, 8].

DRIVE [1] is a recently introduced algorithm for the \((1 + o(1))\)-DME problem (i.e., with one bit per coordinate and some lower-order overhead). To the best of our knowledge, DRIVE is the first algorithm with...

*Equal Contribution.
Algorithm 1 DRIVE [1]

\begin{align*}
\text{Client } c: & \quad 1: \text{Compute } R_c(x_c), S_c. \\
& \quad 2: \text{Send } (S_c, \text{sign}(R_c(x_c))) \text{ to PS.}
\end{align*}

\begin{align*}
\text{Parameter Server (PS):} & \quad 1: \text{For all } c, \text{compute } \hat{x}_c = S_c \cdot R_c^{-1}(\text{sign}(R_c(x_c))). \\
& \quad 2: \text{Estimate } \bar{x}_{avg} = \frac{1}{n} \sum_{c=1}^{n} \hat{x}_c.
\end{align*}

a constant NMSE that uses \( b = 1 + o(1) \). DRIVE shows consistent improvement over the state of the art, over a collection of distributed and federated learning tasks [1]. This technical report generalizes DRIVE to any \( b > 0 \) bandwidth constraint and discusses additional extensions (e.g., resiliency to arbitrary packet drops).

While a bit budget of one bit per coordinate was thoroughly studied [1, 11, 12, 13, 14, 15], and is applied to accelerate distributed learning systems [6, 10], our approach allows one to use DRIVE with \( b > 1 \) for deployment on fast networks or \( b < 1 \) (e.g., half a bit per coordinate) when the network is the bottleneck.

2 Preliminaries

We assume that each client has access to randomness that is shared with the PS. This assumption is standard (e.g., is used in [1, 7] [11]) and can be implemented by having a shared seed (e.g., the combination of client ID and epoch number) for a pseudo-random number generator. For example, a client can generate a random rotation matrix that is also available to the PS as they use the same random bits.

\textbf{Notation: } We use the following notations and definitions in our technical report:

- A matrix \( R \in \mathbb{R}^{d \times d} \) is called a rotation matrix if it satisfies \( R^T R = I \). This implies that for any \( x \in \mathbb{R}^d : \|Rx\|_2 = \|x\|_2 \).
- A random rotation \( \mathcal{R} \) is a distribution over the set of all rotation matrices. For ease of notation, we use \( R(x) \) to denote the random variable \( Rx \) where \( R \sim \mathcal{R} \). Similarly, \( \mathcal{R}^{-1}(x) \) denotes \( R^{-1}x \).
- We denote the \( i \)th coordinate of a vector \( x \) by \( x[i] \) to distinguish it from the input vector of the \( c \)th client, which is denoted by \( x_c \).

\textbf{Problem formulation: } The \( b \)- Distributed Mean Estimation (b-DME) problem considers a set of \( n \in \mathbb{N} \) clients and a PS. Each client \( c \in \{1, \ldots, n\} \) has its own vector \( x_c \in \mathbb{R}^d \), which it sends using a \( (d \cdot b) \)-bits message to the PS. The PS then produces an estimate \( \hat{x}_{avg} \in \mathbb{R}^d \) of the average \( x_{avg} = \frac{1}{n} \sum_{c=1}^{n} x_c \) with the goal of minimizing its normalized mean squared error (NMSE), defined as the average estimate’s mean squared error (MSE) normalized by the average norm of the clients’ original vectors, i.e., \( \frac{\mathbb{E}[\|x_{avg} - \hat{x}_{avg}\|^2]}{\|x_{avg}\|^2} \). We assume that \( \forall c : x_c \neq 0 \) as this can be handled with a single bit. When we discuss a single vector estimation, we denote the NMSE as vector-NMSE (vNMSE), defined as \( \frac{\mathbb{E}[\|x - \hat{x}\|^2]}{\|x\|^2} \), where \( \hat{x} \) is an estimate of \( x \).

3 The DRIVE Algorithm

The pseudocode for DRIVE (Deterministically Rounding randomly rotated VEctors) appears in Algorithm 1. The algorithm has two parameters that determine the compression properties. The first is the distribution \( \mathcal{R} \) from which the rotation matrices \( R_c \) are sampled. Second, we can determine the formula for the scale \( S_c \), calculated by each client \( c \), which is a scalar value that is tuned according to the optimization goals. We now explain its operation in more detail.

\textbf{Clients: } Each client \( c \) uses the shared randomness to sample a rotation matrix \( R_c \sim \mathcal{R} \) and rotates its vector \( x_c \in \mathbb{R}^d \) by computing \( R_c(x_c) \neq R_x x_c \). Then, it calculates a scale, \( S_c \), according to the desired optimization criteria as we explain below. Finally, it sends \( (S_c, \text{sign}(R_c(x_c))) \) to the PS.

\textbf{PS: } The PS reconstructs each client’s vector by first performing the inverse rotation, i.e., it uses the shared randomness to generate the same rotation matrix and computes \( R_c^{-1}(\text{sign}(R_c(x_c))) \). Then, the result is scaled by \( S_c \cdot R_c^{-1}(\text{sign}(R_c(x_c))) \). Finally, the PS averages all results to estimate the mean.
Algorithm 2 DRIVE with quantization level set $Q$

| Client $c$: |
|---|
| 1: Compute $\mathcal{R}_c(x_c)$. |
| 2: Compute $Q(\mathcal{R}_c(x_c))$. |
| 3: Compute $S_c$. |
| 4: Send $(S_c, Q(\mathcal{R}_c(x_c)))$ to PS. |

| Parameter Server (PS): |
|---|
| 1: For all $c$, compute $\hat{x}_c = S_c \cdot \mathcal{R}_c^{-1}(Q(\mathcal{R}_c(x_c)))$. |
| 2: Estimate $\hat{x}_{avg} = \frac{1}{n} \sum_{c=1}^{n} \hat{x}_c$. |

An appealing property of DRIVE is that each client can efficiently calculate the scale parameter $S_c$ to achieve different objectives. In particular, when the rotation matrices are drawn uniformly at random, setting $S_c = \frac{\|x_c\|^2}{\|\mathcal{R}_c(x_c)\|_1}$ results in an unbiased estimate of each vector and an appealing guarantee that the NMSE quickly tends to $\frac{2}{n} - 1\frac{\|x_c\|^2}{\|\mathcal{R}_c(x_c)\|_1}$ as the dimension $d$ increases. For the special case of $n = 1$, by setting $S_c = \frac{\|\mathcal{R}_c(x_c)\|_1}{d}$, it holds that vNMSE $\leq 1 - \frac{d}{2} \approx 0.363$ for any dimension $d$. We note that this scale results in a biased estimate and is less suitable to the case where $n > 1$ as it does not guarantee that the NMSE will decrease proportionally to $\frac{1}{n}$.

Since performing uniform random rotation (denoted by $\mathcal{R}_U$) is computationally expensive, the authors in [1] propose using a structured random rotation [2][7] (i.e., the randomized Hadamard transform, denoted by $\mathcal{R}_H$). The Hadamard transform admits an in-place GPU-friendly time implementation $[18][19]$. For $\mathcal{R}_H$, under the assumption that the gradients coordinates’ distribution admits finite moments, it was shown that using $\mathcal{R}_H$ yields a similar coordinate distribution to that of $\mathcal{R}_U$. It was also empirically shown in experiments that using $\mathcal{R}_H$ results in a similar NMSE as with $\mathcal{R}_U$ for various distributions and dimensions. For the $n = 1$ special case, it was shown that vNMSE $\leq 0.5$ for any $d$ without any assumptions on the gradient distribution.

4 Extensions

4.1 Multiple Bits Per-Coordinate

In this section, we extend the DRIVE algorithm to multiple bits. Intuitively, DRIVE leverages the fact that with the uniform distribution over rotation matrices, all coordinates have an identical distribution that converges to a normal distribution for high dimensions. Specifically, for an input vector $x \in \mathbb{R}^d$, we have that as $d$ tends to infinity, each coordinate $\mathcal{R}_U(x)[i]$’s distribution tends to $\mathcal{N}(0, \frac{1}{d})$ [2]. With this observation, with $b = 1 + o(1)$ bits per coordinate, DRIVE was designed with the quantization levels $\{-S,+S\}$ and maps each rotated coordinate to its closest level [1]. Namely, all positive coordinates are mapped to $+S$ and the negative ones to $-S$. Here, $S$ is the scale, and it is configured according to the desired properties, e.g., having minimal vNMSE or being unbiased. Leveraging this distribution assumption, with $b > 1$ bits per coordinate, we can use multiple quantization levels to approximate the normal distribution.

There are multiple ways to select a set of quantization levels, depending on the desired properties, e.g., compression speed or maximal accuracy, as we explain in the following sections. Here, we provide the pseudo-code for the extended DRIVE in Algorithm 2 given a pre-defined set of quantization levels $Q$. Intuitively, the error of our algorithm depends on how well $Q$ approximates the distribution of the rotated coordinates (which approaches the normal distribution as $d$ increases), while the size of $Q$ (or its entropy) determines the number of bits the algorithm requires. We discuss potential choices for $Q$ in the following section. In our extension, each client $c$ first computes the rotated vector $\mathcal{R}_c(x_c)$. Next, it computes the closest quantized vector to $\mathcal{R}_c(x_c)$:

$$Q(\mathcal{R}_c(x_c)) = \arg\min_{y \in \mathcal{Q}} \left\| \mathcal{R}_c(x_c) - y \right\|_2 \left\| \frac{x_c}{\sqrt{d}} \right\|_2.$$ 

That is, $Q(\mathcal{R}_c(x_c))$ is a $d$-dimensional vector with entries from $Q$ after scaling by $\frac{x_c}{\sqrt{d}}$. Notice that this can be efficiently implemented as each coordinate is mapped independently to its closest scaled quantization level. Intuitively, the quantization levels $Q$ approximate the $\mathcal{N}(0, 1)$ distribution while each coordinate in $\mathcal{R}_c(x_c)$

\footnote{For finite $d$ values, the coordinates follow a shifted Beta distribution that is close to normal even for $d$ values that are just a few hundreds. We refer the reader to [1] for more details.}
has a distribution that approaches (as $d \to \infty$) $\mathcal{N}\left(0, \frac{||z||^2}{d}\right)$. This way, we can use the same quantization levels for all clients regardless of their vector norms. Despite having a distribution that is not normal for any finite $d$, we later show that our algorithm can derive unbiased estimates for all reasonable quantization level sets, including those that approximate $\mathcal{N}(0, 1)$. Finally, the algorithm computes the scale $S_c$ and sends $(S_c, Q(R_c(x_c)))$ to the parameter server. In turn, the PS receives $\{(S_c, Q(R_c(x_c))) \mid c \in \{1, \ldots, n\}\}$, estimates each input vector as $\hat{x}_c = S_c \cdot R_c^{-1}(Q(R_c(x_c)))$ and computes their average. We propose that each client would set its scale to $S_c = \frac{||x_c||^2}{\|Q(R_c(x_c))\|1}$. In Appendix A, we prove the following theorem which states that with the proposed scale and using a uniform random rotation $R_U$, any choice of quantization levels set $Q$ that is sign-symmetric results in unbiased estimates. When clients sample their rotation matrices independently, it leads to an NMSE that decreases proportionally to $1/n$.

**Theorem 1.** Consider a set of quantization levels $Q$ that satisfies $Q(-y) = -Q(y)$ for any $y \in \mathbb{R}^d$. Then, for any $x \in \mathbb{R}^d$, using DRIVE with a scale $S = \frac{||x||^2}{\|Q(R_U(x))\|1}$ results in $\mathbb{E}[\hat{x}] = x$.

We note that our DRIVE extension generalizes the original algorithm. Indeed, notice that for $Q = \{-\psi, +\psi\}$ for some $\psi > 0$ (which is implementable using a single bit per coordinate), Algorithm 2 coincides with the $(1 + o(1))$ algorithm (which has $\hat{x}_c = \frac{||x_c||^2}{\|Q(R_0(x_c))\|1} \cdot R_c^{-1}(\text{sign}(R_c(x_c)))$ as described in Section B). To see this, notice that $Q(R_c(x_c)) = \psi \cdot \text{sign}(R_c(x_c))$ and $S_c = \frac{||x_c||^2}{\|Q(R_0(x_c))\|1} = \frac{||x_c||^2}{\psi \|R_0(x_c)\|1}$. Therefore,

$$\hat{x}_c = S_c \cdot R_c^{-1}(Q(R_c(x_c))) = S_c \cdot R_c^{-1}(\psi \cdot \text{sign}(R_c(x_c))) = \frac{||x_c||^2}{\|R_0(x_c)\|1} \cdot R_c^{-1}(\text{sign}(R_c(x_c))).$$

### 4.2 Choosing the Quantization Level Set $Q$ for Computational Efficiency

Using $b$ bits per coordinate, we can use the $z = 2^b$ quantization levels that minimize the expected squared distance from a normal random variable to its closest quantization level. This is known as a **Lloyd-Max Scalar Quantizer** [20] [21] for the normal distribution. Given a set of quantization levels $Q$, we denote its expected squared error by

$$V_Q = \int_{-\infty}^{\infty} \frac{1}{\sqrt{2\pi}} e^{-t^2/2} \cdot \min \{ (t - q)^2 \mid q \in \mathcal{Q} \} \cdot \text{dt}.$$  

Given a size constraint $|\mathcal{Q}| = z$, we denote by $Q_z$ the optimal set of quantization levels. That is,

$$Q_z = \text{argmin}_{|\mathcal{Q}| = z} V_Q.$$

Given the $Q_z$ quantization levels, we get an accurate representation for the rotated coordinates. Recall that the distribution of each rotated coordinate $R(x)[i]$ (as $d \to \infty$) approaches $\mathcal{N}(0, ||x||^2/d)$. That is, we can compute the set of optimal quantization levels $Q_z$ once and use it for all sufficiently large $d$. We list the values relevant to $b = 1, 2, 3$ bits in Table [I].

| #bits ($b$) | #quantization levels ($z$) | Quantization Levels $Q_z$ | Expected Sq. Error $V_{Q_z}$ |
|-------------|-----------------------------|-----------------------------|-----------------------------|
| 0           | 1                           | \{±\sqrt{\frac{1}{2}}\} ≈ \{±0.707107\} | 1                           |
| 1           | 2                           | \{±\sqrt{\frac{1}{4}}\} ≈ \{±0.707107\} | 1 - \frac{2}{\pi} ≈ 0.363380 |
| 2           | 4                           | \{±0.45278, ±1.51048\}       | 0.117481                    |
| 3           | 8                           | \{±0.245094, ±0.756005, ±1.343909, ±2.151946\} | 0.034547                   |

Table 1: The quantization levels and expected squared error of the $\mathcal{N}(0, 1)$ distribution.

**Implementation notes:** For any integer $b \geq 1$, with $Q = Q_{2^b}$, sending $(S_c, Q(R_c(x_c)))$ requires $b(1 + o(1))$ bits. We refer the reader to the discussion in [I] about the number of bits needed to represent the scale. We also note that $Q(R_c(x_c))$ is computable in $O(b \cdot d)$ time using binary search and is amenable to an efficient parallel GPU implementation, e.g., using `torch.bucketize`. In Section 4.3, we generalize the solution for any real $b > 0$.

Note that one can get a lower error by optimizing the quantization levels for the actual distribution of the rotated coordinates (which is shifted Beta as explained in [I]). However, the distribution rapidly approaches the normal distribution as $d$ grows, and our focus is on federated learning where $d$ can exceed a billion.
Expected Squared Error (which is encodable with ECVQ) [25]. The algorithm proposed in [25] has several tunable parameters that may affect the output of the algorithm.

A standard approach to compressing vectors with a small number of possible values that are not distributed equally is using entropy encoding such as Huffman [23] or arithmetic encoding [24]. Intuitively, the number of times each quantization level appears in $Q/R_{c}(x, e)$ may not be $d/z$. Indeed, for $z > 2$, the probability of each quantization level is not uniform. For example, for a vector of i.i.d. $N(0, 1)$ entries and $z = 4$ (two bits per coordinate on average), the expected frequency of the quantization levels $\pm 0.45278$ is $\approx 0.337$ while the quantization levels $\pm 1.510418$ are only expected to appear in $\approx 0.163$ of the entries. This suggests that, for a large enough dimension $d$, we should be able to compress the vector to approximately $H_{Q_{a}} + \epsilon$ bits per coordinate, where $H_{Q_{a}} = -2 \cdot (0.337 \log_{2}(0.337) + 0.163 \log_{2}(0.163)) \approx 1.91$ is the entropy of the distribution of $Q_{a}$.

Formally, for each $q \in Q$ let $T_{c} \subseteq R$ denote all real numbers whose closest quantization level is $c$, and let $p_{q} = \int_{c \in T_{c}} e^{-t^{2}/2} dt$ denote its probability. Then $H_{Q} = \sum_{q \in Q} -p_{q} \log_{2}(p_{q})$ is the entropy of the distribution induced by $N(0, 1)$ and $Q$. Such encoding allows us to use more quantization levels for the same bit budget. For example, using $z = 9$ quantization levels, we get an entropy of $H_{Q_{a}} \approx 2.98$, which would allow us to use three bits per coordinate for large enough vectors. In particular, this would reduce the error, as $V_{Q_{a}} \approx 0.0278529 \approx 0.806 \cdot V_{Q_{a}}$ at the cost of additional computation.

Taking this a step further, it is possible to consider the resulting entropy when choosing the set of quantization levels. To optimize the quantization, we are looking for a set $Q$ with minimal $V_{Q}$ such that its entropy is bounded by $b - \epsilon$. This problem is called Entropy-Constrained Vector Quantization (ECVQ) [25]. The algorithm proposed in [25] has several tunable parameters that may affect the output of the algorithm. We implemented the algorithm and scanned a large variety of parameter values. For $b = 3$, the best obtained quantization levels were $Q_{ECVQ,b=3} \approx \{0.0, \pm 0.522, \pm 1.043, \pm 1.567, \pm 2.091, \pm 2.617, \pm 3.145, \pm 3.676, \pm 4.210, \pm 4.747, \pm 5.289, \pm 5.859\}$, which resulted in an expected squared error of $V_{Q_{ECVQ,b=3}} \approx 0.02274$.

We propose a different approach that is more computationally efficient. Given a bandwidth constraint $b > 1$, let $\Delta_{b} > 0$ denote the smallest real number such that the entropy of $Q_{b} = \{\Delta_{b} \cdot n | n \in Z\}$ is at most $b - \epsilon$ with respect to $N(0, 1)$. That is, we compute a constant $\Delta_{b}$ such that the set of all quantization levels of the form $n \cdot \Delta_{b}$ (for all integer $n$ values) has an entropy smaller than $b$. For example, using $b = 3$ bits per coordinate, we have $\Delta_{3} \approx 0.5226$. Then, $V_{Q_{b}} \approx 0.0227594$, an improvement of 17.3% over $V_{Q_{b}}$ (which we can encode with $b = 3$ bits per coordinate by applying entropy encoding over $Q_{b}$) and of 34% over $V_{Q_{b}}$ (which is encodable with $b = 3$ without compression). Our expected squared error also satisfies $V_{Q_{b}} \approx 1.0008 \cdot V_{Q_{ECVQ,b=3}}$, i.e., only 0.08% worse than allowing non-uniform spacing between the quantization levels. The benefit is that our approach allows computing the quantization much faster as each $y \in R$ is efficiently mapped into the quantization level $\Delta_{b} \cdot \left\lfloor \frac{y}{\Delta_{b}} + 1/2 \right\rfloor$ (instead of performing a binary search over $Q_{ECVQ,b}$).

As noted, the algorithm is unbiased for any sign-symmetric set of quantization levels, including $Q_{b}$.

The rate-distortion theory implies a lower bound on $V_{Q}$ for any quantization level set $Q$ [26]. Specifically, it implies that $V_{Q} \geq 4^{-b}$ for any $Q$ such that $H_{Q} \leq b$. We illustrate the different quantization levels and their expected squared error in Figure 1. As shown, our Equal Intervals approach is close to the lower bound while allowing computationally efficient quantization.

Figure 1: The expected squared error of the different quantization level sets approximating $N(0, 1)$.

4.3 Reducing the Error with Entropy Encoding

The benefit is that our approach allows computing the quantization much faster as each $y \in R$ is efficiently mapped into the quantization level $\Delta_{b} \cdot \left\lfloor \frac{y}{\Delta_{b}} + 1/2 \right\rfloor$ (instead of performing a binary search over $Q_{ECVQ,b}$).
We note that in practice, the frequency of a quantization level \( q \in \mathcal{Q} \) would not be exactly \( p_q \). Nonetheless, using arithmetic-coding, we can get an encoding that uses \((H_{\mathcal{Q}} + \epsilon)(1 + o(1))\) bits per coordinate on average. A proof sketch, which assumes that the coordinates are independent (in practice, they are weakly dependent for a sufficiently large dimension \( d \)) follows. We defer the formal proof to future work. As indicated by [27, Chapter 10], from the fact that each coordinate \( i \) of a vector \( y \in \mathbb{R}^d \), decreases the length of the encoded interval by a factor of \( p_{\mathcal{Q}(y[i])} \), where \( \mathcal{Q}(y[i]) \) is the random variable that represents the quantization level of \( y[i] \). Therefore, the length of the interval of the vector is \( L_y = \prod_{i=0}^{d-1} p_{\mathcal{Q}(y[i])} \) which means that the representation of \( y \) requires \( \log_2 (1/L_y) + 1 \leq 2 + \sum_{i=0}^{d-1} \log_2 1/p_{\mathcal{Q}(y[i])} \). A standard application of the Chernoff bound suffices to complete the argument.

### 4.4 Using Heterogeneous Quantization Level Sets

Heretofore, we considered a single set of quantization levels \( \mathcal{Q} \) and quantized all coordinates according to \( \mathcal{Q} \). Here, we consider a more generalized algorithm where \( \mathcal{Q} = (\mathcal{Q}[0], \mathcal{Q}[1], \ldots, \mathcal{Q}[d-1]) \) is a vector a quantization level sets such that \( \mathcal{Q}(\mathcal{R}_c(x_c)) \) means that each coordinate \( \mathcal{R}_c(x_c)[i] \) is mapped to the closest quantization level in \( \mathcal{Q}[i] \). Notice that, using the same scale, the unbiasedness proof (given in Appendix A) directly generalizes to this setting as well as long as each \( \mathcal{Q}[i] \) is sign-symmetric and there exists (at least one) \( \mathcal{Q}[i] \) such that \( |\mathcal{Q}[i]| \geq 2 \). Below, we give several applications for this generalization.

**Handling arbitrary bandwidth constraints:** There are many possible ways to use DRIVE given a budget of \( b \) bits per coordinate. For example, if \( b > 1 \), we can use the entropy-constrained vector quantization approach. However, in cases where we wish to optimize the compression time, using arithmetic encoding may be prohibitively slow.

Instead, we propose the following alternative. For a given \( b > 0 \) bandwidth constraint, we send \([(b - [b]) \cdot d] \) rotated coordinates using \( \mathcal{Q}[2\cdot[b]+1] \) and the rest using \( \mathcal{Q}[2\cdot[b]+1] \). For example, using \( b = 2/3 \), we send \([2d/3] \) coordinates using \( \mathcal{Q}[2] \) (which take one bit per coordinate) while the rest are not transmitted (as \( \mathcal{Q}[1] = [0] \)). Another example is \( b = 1.5 \) where half of the coordinates are sent using \( \mathcal{Q}[4] \) and the rest use \( \mathcal{Q}[2] \). Notice that the choice of which coordinates to send using more bits can be predetermined (e.g., the first ones) or randomly selected using shared randomness, but we do not need to explicitly communicate them.

We note that overcoming packet loss when using entropy constraint is challenging, as arithmetic encoding could fail even if a single bit is lost. In such a case, one may have to encode each packet separately, increasing the bit overhead.

**DRIVE-ing over lossy networks:** State of the art distributed and federated learning systems (e.g., [6, 16]) assume reliable packet delivery, e.g., using TCP to retransmit lost packets or using RDMA/RoCEv2, which depends on a lossless fabric. However, it is useful to design algorithms that can cope with occasional packet loss on standard IP networks. Indeed, a recent effort by [28] extends SGD to support packet loss. Intuitively, we can model packet loss as having coordinates that are quantized to \( \mathcal{Q}[1] = [0] \) in our algorithm. That is, our unbiasedness result does not require that the set of quantization levels would be pre-determined. Specifically, a client may use a different quantization level set \( \mathcal{Q}[i] \) to send the \( i \)th rotated coordinate, but if the packet is lost in traffic, the PS may use \( [0] \) when estimating the gradient. This way, the PS does not need to wait to receive all coordinates before estimating the average gradient. For example, this enables the use of lossy transport protocols such as UDP (e.g., as proposed by [28]) while estimating the average after enough time has lapsed for receiving the packets, but without waiting for potential retransmissions. Another benefit is that, since it does not require maintaining reliable delivery, lossy protocols often have reduced overheads (e.g., smaller headers that allow encoding more coordinates in each packet).

Interestingly, our approach is robust to adversarial packet loss. That is, we assume that the adversary can determine both the input \( \{x_c \mid c \in \{1, \ldots, n\}\} \) and the sequence numbers of the dropped packets (but not the shared randomness that is used to rotate the vector). In such a setting, the adversary cannot affect the expected squared error since it depends only on the number of lost rotated coordinates and not their index (recall that all rotated coordinates are identically distributed).

**Heterogeneous client bandwidth constraints:** Often in federated learning clients may have different resource constraints, particularly networking constraints [29]. Therefore, it is beneficial for an algorithm to allow clients to use different bandwidth constraints and assume different network loss rates. We note that
our DRIVE extension remains unbiased even when allowing each client to select a dedicated quantization levels set \( Q_c \) (in coordination with the PS).

### 4.5 Bounding the Resulting NMSE

We begin with bounding the sum of squared errors (SSE) of each client.

**Lemma 1.** Given \( x \in \mathbb{R}^d \), the SSE of DRIVE is: 
\[
\|x\|^2 - 2S \langle \mathcal{R}(x), Q(\mathcal{R}(x)) \rangle + S^2 \|Q(\mathcal{R}(x))\|^2.
\]

**Proof.** The SSE in estimating \( \mathcal{R}(x) \) using \( S \cdot Q(\mathcal{R}(x)) \) equals that of estimating \( x \) using \( \hat{x} \). Therefore,
\[
\|x - \hat{x}\|^2 = \|\mathcal{R}(x) - \mathcal{R}(\hat{x})\|^2 = \|\mathcal{R}(x) - \mathcal{R}^{-1}(S \cdot Q(\mathcal{R}(x)))\|^2 = \|\mathcal{R}(x) - S \cdot Q(\mathcal{R}(x))\|^2 = \|\mathcal{R}(x)\|^2 - 2S \langle \mathcal{R}(x), Q(\mathcal{R}(x)) \rangle + S^2 \|Q(\mathcal{R}(x))\|^2 = \|x\|^2 - 2S \langle \mathcal{R}(x), Q(\mathcal{R}(x)) \rangle + S^2 \|Q(\mathcal{R}(x))\|^2.
\]

We now continue to bound the vNMSE of each client. In particular, using the recommended scale for \( n > 1 \), i.e., \( S = \frac{\|x\|^2}{\|Q(\mathcal{R}(x))\|^2} \), the SSE of DRIVE becomes:
\[
\|x - \hat{x}\|^2 = \|x\|^2 - 2S \langle \mathcal{R}(x), Q(\mathcal{R}(x)) \rangle + S^2 \|Q(\mathcal{R}(x))\|^2 = \|x\|^2 \frac{\|Q(\mathcal{R}(x))\|^2}{\langle \mathcal{R}(x), Q(\mathcal{R}(x)) \rangle} - \|x\|^2.
\]

Thus, the resulting vNMSE in this case is:
\[
\mathbb{E}\left[ \frac{\|x - \hat{x}\|^2}{\|x\|^2} \right] = \mathbb{E}\left[ \frac{\|x\|^2 \|Q(\mathcal{R}(x))\|^2}{\langle \mathcal{R}(x), Q(\mathcal{R}(x)) \rangle^2} \right] - 1.
\]

It is of our future interest to investigate the above expression and the resulting NMSE for \( n > 1 \) with respect to choices of \( \mathcal{R} \) and \( Q \).

For example, we can use the property of uniform random rotation \( \mathcal{R}_U \), which maps vectors into a uniformly distributed point on a sphere, preserving the original vector’s norm, and \( Q = \{-\psi, +\psi\} \) (which meets the \( b = 1 + o(1) \) bandwidth requirement) to write:
\[
\mathbb{E}\left[ \frac{\|x - \hat{x}\|^2}{\|x\|^2} \right] = \mathbb{E}\left[ \frac{\|x\|^2 \|Q(\mathcal{R}_U(x))\|^2}{\langle \mathcal{R}_U(x), Q(\mathcal{R}_U(x)) \rangle^2} \right] - 1 = \mathbb{E}\left[ \frac{\|x\|^2 \cdot d \cdot \psi^2}{\langle \mathcal{R}_U(x) \parallel \psi \rangle^2} \right] - 1 = \mathbb{E}\left[ \frac{\|x\|^2 \cdot d}{\|\mathcal{R}_U(x)\|^2} \right] - 1 = \mathbb{E}\left[ \frac{d}{\|T\|^2} \right] - 1,
\]

where \( T \in S^{d-1} \) is a uniformly distributed random point on the unit sphere. That is, as pointed out in \[1\], the vNMSE is independent of the input vector \( x \). Further, assuming that the clients sample the rotation matrices in an independent manner and using Theorem \[1\], we get that the NMSE equals the vNMSE divided by \( n \). That is:
\[
\frac{1}{n} \cdot \left( \mathbb{E}\left[ \frac{d}{\|T\|^2} \right] - 1 \right),
\]

which is the NMSE of DRIVE \[1\].

Finally, for the special case where there is only \( n = 1 \) client, and the estimate is not required to be unbiased, the goal is to minimize the vNMSE. We refer to this problem as \( b \)-Vector Estimation (b-VE). In this case, to minimize the SSE we require
\[
\frac{d}{\mathcal{S}} \left( \|x\|^2 - 2S \langle \mathcal{R}(x), Q(\mathcal{R}(x)) \rangle + S^2 \|Q(\mathcal{R}(x))\|^2 \right) = -2 \cdot \langle \mathcal{R}(x), Q(\mathcal{R}(x)) \rangle + 2 \cdot \|Q(\mathcal{R}(x))\|^2 \cdot S = 0,
\]
leading to \( S = \frac{\langle \mathcal{R}(x), Q(\mathcal{R}(x)) \rangle}{\|Q(\mathcal{R}(x))\|^2} \). Then, the SSE of DRIVE becomes:
\[
\|x - \hat{x}\|^2 = \|x\|^2 - 2S \langle \mathcal{R}(x), Q(\mathcal{R}(x)) \rangle + S^2 \|Q(\mathcal{R}(x))\|^2 = \|x\|^2 \frac{2 \langle \mathcal{R}(x), Q(\mathcal{R}(x)) \rangle^2}{\|Q(\mathcal{R}(x))\|^2} + \frac{\mathcal{R}(x), Q(\mathcal{R}(x)) \rangle^2}{\|Q(\mathcal{R}(x))\|^2} = \|x\|^2 - \frac{\langle \mathcal{R}(x), Q(\mathcal{R}(x)) \rangle^2}{\|Q(\mathcal{R}(x))\|^2}.
\]

7
Therefore, the vNMSE becomes:

$$\mathbb{E} \left[ \frac{\|x - \bar{x}\|_2^2}{\|x\|_2^2} \right] = 1 - \mathbb{E} \left[ \frac{\langle \mathcal{R}(x), Q(\mathcal{R}(x)) \rangle^2}{\|x\|_2^2 \cdot \|Q(\mathcal{R}(x))\|_2^2} \right].$$

Similarly to the unbiased case, it is of our future interest to investigate the above expression with respect to choices of $\mathcal{R}$ and $Q$. Note that once again by using $\mathcal{R}_U$ and and $Q = \{-\psi, +\psi\}$ (which meets the $b = 1 + o(1)$ bandwidth requirement), we obtain DRIVE’s vNMSE $= (1 - \frac{2}{3})(1 - \frac{1}{3})$.

### 4.6 Preliminary Evaluation

In this section, we evaluate DRIVE’s NMSE with the different extensions suggested above. Throughout this section, we use the Hadamard random rotation $\mathcal{R}_H$. As shown in [1], while the rotated vectors $\{\mathcal{R}_H(x)\}$ do not follow a uniform distribution as in $\mathcal{R}_U$, reasonable assumptions on the input vector $x$ imply that the distribution of $\mathcal{R}_H(x)$ tends to uniform as the dimension $d$ increases. Therefore, we expect our DRIVE extension to work well with $\mathcal{R}_H$ for large vectors.

We follow the setting used by [1], where $n = 10$ clients have the same input vector, where each coordinate is drawn i.i.d. from a LogNormal(0,1) distribution. Intuitively, this setting tests the encoding bias of the algorithm as the biased will be accumulated for the different clients instead of being hidden due to differences in the input vectors. Each experiment was repeated 100 times, and the average result is reported.

#### Arbitrary bandwidth constraints:

We start by evaluating the NMSE of DRIVE without the $b = 1 + o(1)$ bits limitation as in [1]. First, we explore the $b \in (0,1]$ regime to understand its behavior in the sub-bit domain. The result, shown in Figure 2a, shows that DRIVE gracefully degrades when $b$ decreases. Here, DRIVE sends a $b$ fraction of the coordinates using $Q_2$ while the others are not sent (i.e., quantized to zero using $Q_1$). The figure also shows the behavior for different vector dimensions $d$. As expected, the NMSE is higher for small dimensions as the coordinates distribution, which is shifted Beta as analyzed in [1], is not close enough to the normal distribution. Nonetheless, for $d = 128$, the NMSE almost matches that of $d = 2^{20}$, which means that choosing a quantization levels set $Q$ that approximation $\mathcal{N}(0,1)$ works well unless the dimension is quite small. Another explanation is that while the distribution of $\mathcal{R}_H(x)$ coordinates tends to that of $\mathcal{R}_U(x)$ for large dimensions (assuming the distribution of $x$ admits finite moments, as is the case in our experiment), its distribution for $d = 16$ varies substantially, and thus the estimates may have some bias.

Next, we empirically measure the NMSE for $b > 1$ bits. As shown in Figure 2b, DRIVE effectively uses additional bandwidth compared to the $\approx 0.0571$ NMSE of the original $b = 1 + o(1)$-bit DRIVE. Entropy compression allows further error reduction at the cost of additional computation.

#### Packet loss:

We measured the error of DRIVE when facing packet loss. In our simulation, the PS collects all the packets that reached him without waiting for retransmission of lost packets, as explained in Section 4.4. The results, shown in Figure 3, indicate that DRIVE is barely affected by reasonable ($\leq 10^{-3}$) loss levels when using up to $b = 5$ bits per coordinate. When the network loss is exceedingly high (e.g., 50% of the packets are lost), the improvement of quantizing each coordinate to multiple bits quickly diminishes.
Nonetheless, the DRIVE with $Q_2$ (one bit) degrades by less than $4 \times$ factor compared to lossless delivery. This implies that for a high loss rate, it may be beneficial to use fewer bits per coordinate with redundancy to minimize the number of lost coordinates.

**Compression speed:** Our measurements were performed on the same Intel Core i7-7700 CPU (8 cores, 3.60 GHz, and 8 MB cache), 64 GB RAM, NVIDIA GeForce GTX 1060 (6GB) GPU, Windows 10 (build 18363.1556) operating system, and CUDA release 10.2, V10.2.89 as in [1]. We have computed the quantization level sets $\{Q_n\}$ offline and use `torch.bucketize` to map vector to its closest quantization (speed evaluation for the entropy encoded quantization sets were not conducted). Our measurements indicate that for $b = 2, 3, 4, 5$, DRIVE is able to compress vectors of size $2^{25}$ (the same setting as in [1]) within 160ms. This indicates that `torch.bucketize` is not optimized for a small number of quantization levels, and further optimizations are needed.

## 5 Conclusions and Future Work

In this technical report, we studied the extension of DRIVE [1], a state of the art algorithm for the one-bit distributed mean estimation problem, to arbitrary bandwidth constraints. We explored several strategies, including ones oriented towards compression speed and others that optimize the bandwidth to error tradeoff. Our results indicate that DRIVE gracefully degrades when reducing the bandwidth constraint to less than one bit per coordinate and effectively leverages any additional bandwidth to reduce the error rates. We have also proved that DRIVE can produce unbiased estimates for any sign-symmetric quantization level set $Q$.

We now identify several gaps that are interesting for future research. First, it is interesting to analyze how the vNMSE of DRIVE behaves for different rotations $R$ and quantization levels $Q$. Second, our current evaluation does not compare to other works in the $b > 1$ parameter range, with or without entropy encoding. Finally, our scheme does not determine what the right way to pack the rotated coordinates is (e.g., how many to put in each packet, how many bits to encode each with, etc.) to optimize the training time over lossy networks.

## References

[1] Shay Vargaftik, Ran Ben Basat, Amit Portnoy, Gal Mendelson, Yaniv Ben-Itzhak, and Michael Mitzenmacher. DRIVE: One-bit Distributed Mean Estimation. *arXiv preprint arXiv:2105.08339*, 2021.

[2] 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.

[3] 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.

[4] 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.

[5] Yanping Huang, Youlong Cheng, Ankur Bapna, Orhan Firat, Dehao Chen, Mia Chen, HyoukJoong Lee, Jiqian Ngiam, Quoc V Le, Yonghui Wu, and zhiheng 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.

[6] Youhui Bai, Cheng Li, Quan Zhou, Jun Yi, Ping Gong, Feng Yan, Ruichuan Chen, and Yinlong Xu. Gradient Compression Supercharged High-Performance Data Parallel DNN Training. In *ACM SOSP*, 2021.

[7] 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.

[8] 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.

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

[10] 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.

[11] 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.

[12] 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.

[13] Jeremy Bernstein, Yu-Xiang Wang, Kamyar Azizzadenesheli, and Animashree Anandkumar. signSGD: Compressed Optimisation for Non-Convex Problems. In *International Conference on Machine Learning*, pages 560–569, 2018.

[14] 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.

[15] Wei Wen, Cong Xu, Feng Yan, Chunpeng Wu, Yandang 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.

[16] Yimin Jiang, Yibo Zhu, Chang Lan, Bairen Yi, Yong Cui, and Chuanxiong Guo. A unified architecture for accelerating distributed DNN training in heterogeneous gpu/cpu clusters. In *14th USENIX Symposium on Operating Systems Design and Implementation (OSDI 20)*, pages 463–479. USENIX Association, November 2020.

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

[18] 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.

[19] 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](https://github.com/uber-research/intrinsic-dimension).
[20] Stuart Lloyd. Least squares quantization in pcm. *IEEE transactions on information theory*, 28(2):129–137, 1982.

[21] Joel Max. Quantizing for minimum distortion. *IRE Transactions on Information Theory*, 6(1):7–12, 1960.

[22] David A Huffman. A method for the construction of minimum-redundancy codes. *Proceedings of the IRE*, 40(9):1098–1101, 1952.

[23] Richard Clark Pasco. *Source coding algorithms for fast data compression*. PhD thesis, Stanford University CA, 1976.

[24] Jorma J Rissanen. Generalized kraft inequality and arithmetic coding. *IBM Journal of research and development*, 20(3):198–203, 1976.

[25] Philip A Chou, Tom Lookabaugh, and Robert M Gray. Entropy-constrained vector quantization. *IEEE Transactions on acoustics, speech, and signal processing*, 37(1):31–42, 1989.

[26] Thomas M Cover. *Elements of information theory*. John Wiley & Sons, 1999.

[27] Michael Mitzenmacher and Eli Upfal. *Probability and computing: Randomization and probabilistic techniques in algorithms and data analysis*. Cambridge university press, 2017.

[28] Hao Ye, Le Liang, and Geoffrey Li. Decentralized Federated Learning with Unreliable Communications. *arXiv preprint arXiv:2108.02397*, 2021.

[29] Takayuki Nishio and Ryo Yonetani. Client selection for federated learning with heterogeneous resources in mobile edge. In *ICC 2019-2019 IEEE International Conference on Communications (ICC)*, pages 1–7. IEEE, 2019.
A Unbiasedness proof

**Theorem 1.** Consider a set of quantization levels $Q$ that satisfies $Q(-y) = -Q(y)$ for any $y \in \mathbb{R}^d$. Then, for any $x \in \mathbb{R}^d$, using DRIVE with a scale $S = \frac{\|x\|^2}{(\langle R_U(x),Q(R_U(x)) \rangle)}$ results in $\mathbb{E}[\hat{x}] = x$.

**Proof.** For any $x \in \mathbb{R}^d$ denote $x' = (\|x\|_2,0,\ldots,0)^T$ and let $R_{x-x'} \in \mathbb{R}^{d \times d}$ be a rotation matrix such that $R_{x-x'} \cdot x = x'$. Further, denote $R_x = R_U R_{x-x'}$. Using these definitions we have that,

$$\hat{x} = R_{x-x'}^{-1} R_{x-x'}' \hat{x} = S \cdot R_{x-x'}^{-1} R_{x-x'} \cdot R_U^{-1} \cdot Q(R_U \cdot x) = S \cdot R_{x-x'}^{-1} \cdot R_x^{-1} \cdot Q(R_x \cdot R_{x-x'} \cdot x) = S \cdot R_{x-x'}^{-1} \cdot R_x^{-1} \cdot Q(R_x \cdot x') .$$

Let $C_i$ be a vector containing the values of the $i$'th column of $R_x$. Then, $R_x \cdot x' = \|x\|_2 \cdot C_0$ and we obtain,

$$R_x^{-1} \cdot Q(R_x \cdot x') = \left( \langle C_0, Q(\|x\|_2 \cdot C_0) \rangle, \langle C_1, Q(\|x\|_2 \cdot C_0) \rangle, \ldots, \langle C_{d-1}, Q(\|x\|_2 \cdot C_0) \rangle \right)^T .$$

Now, observe that $\langle R_U \cdot x, Q(R_U \cdot x) \rangle = \langle R_x \cdot x', Q(R_x \cdot x') \rangle = \|x\|_2 \cdot \langle C_0, Q(\|x\|_2 \cdot C_0) \rangle$. This yields,

$$\hat{x} = R_{x-x'}^{-1} \cdot \|x\|_2 \cdot \left( \begin{array}{c} \langle C_1, Q(\|x\|_2 \cdot C_0) \rangle \\ \langle C_0, Q(\|x\|_2 \cdot C_0) \rangle \\ \vdots \\ \langle C_{d-1}, Q(\|x\|_2 \cdot C_0) \rangle \\ \end{array} \right) .$$

(1)

Consider an algorithm DRIVE' that operates exactly as DRIVE but, instead of directly using the sampled rotation matrix $R_U = R_x \cdot R_{x-x'}^{-1}$, it calculates and uses the rotation matrix $R_U' = R_x \cdot I' \cdot R_{x-x'}^{-1}$ where $I'$ is identical to the $d$-dimensional identity matrix with the exception that $I'_{10} = -1$ instead of 1. Since $R_U \sim R_U'$ and $R_{x-x'}$ is a fixed rotation matrix, we have that $R_x \sim R_U$. In turn, this also means that $R_x \cdot I' \sim R_U$ since $I'$ is a fixed rotation matrix.

Consider a run of both algorithm where $\hat{x}$ is the reconstruction of DRIVE for $x$ with a sampled rotation $R_U$ and $\hat{x}'$ is the corresponding reconstruction of DRIVE' for $x$ with the rotation $R_U'$.

According to (1) it holds that: $\hat{x} + \hat{x}' = R_{x-x'}^{-1} \cdot \|x\|_2 \cdot (2,0,\ldots,0)^T = 2 \cdot x$. This is because both runs are identical except that the first column of $R_x$ and $R_x \cdot I'$ have opposite signs and thus for all $i \in \{1,2,\ldots,d-1\}$:

$$\langle C_i, Q(\|x\|_2 \cdot C_0) \rangle - \langle -C_i, Q(\|x\|_2 \cdot -C_0) \rangle = \langle C_i, Q(\|x\|_2 \cdot C_0) \rangle - \langle -C_i, Q(\|x\|_2 \cdot -C_0) \rangle = 0 .$$

Note that we used the fact that $Q(-y) = -Q(y)$ for any $y \in \mathbb{R}^d$.

Finally, it holds that $\mathbb{E}[\hat{x} + \hat{x}'] = 2 \cdot x$. But, since $R_x \sim R_U$ and $R_x \cdot I' \sim R_U$, both algorithms have the same expected value. This yields $\mathbb{E}[\hat{x}] = \mathbb{E}[\hat{x}'] = x$. This concludes the proof. \qed