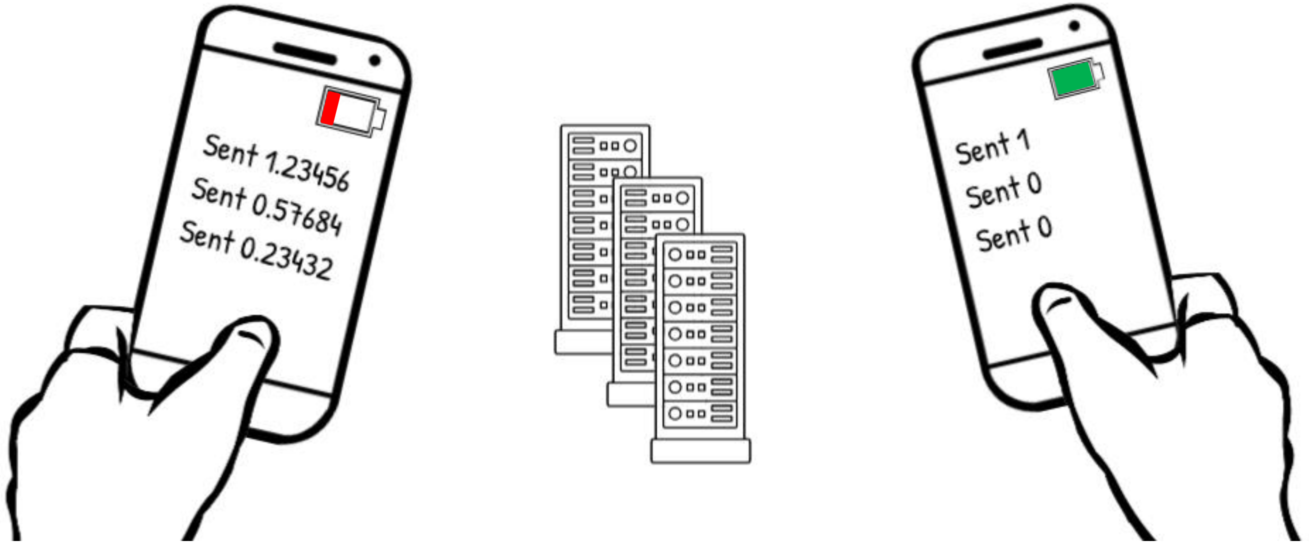
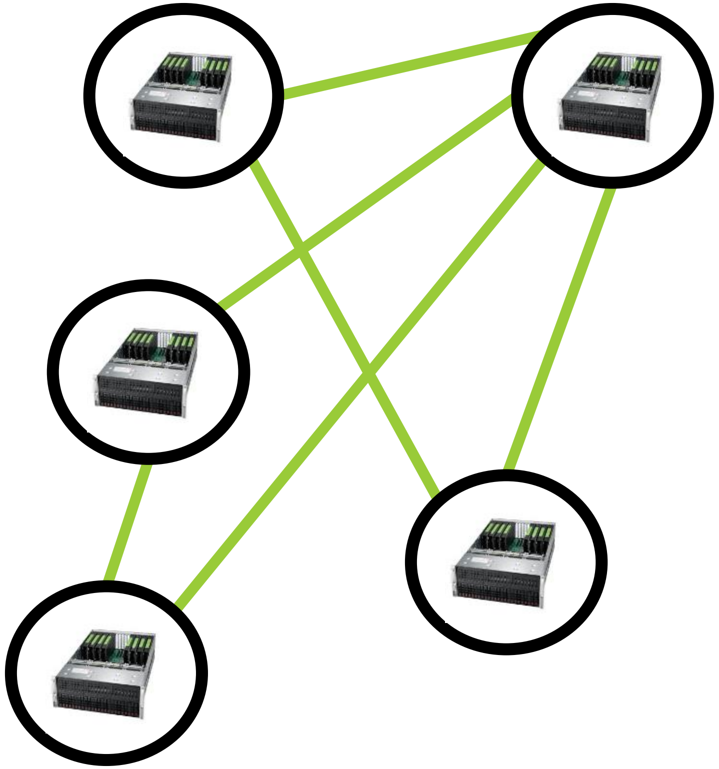
DRIVE: One-bit Distributed Mean Estimation

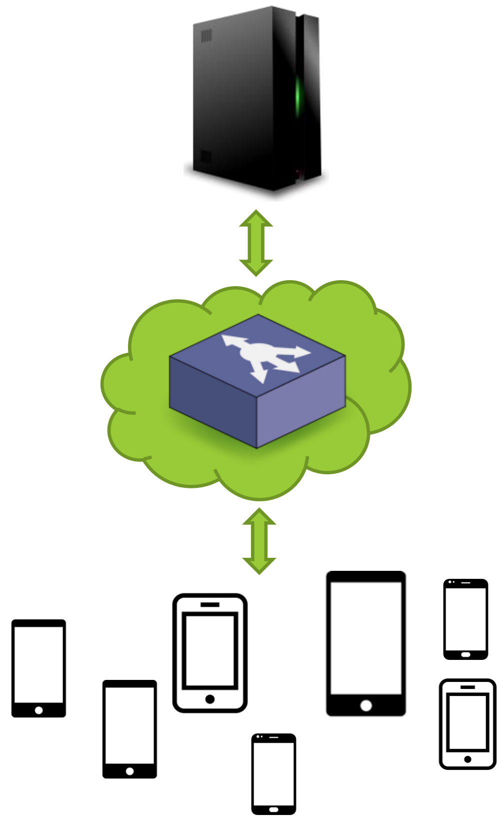
Shay Vargaftik (VMware Research)
Ran Ben-Basat (University College London)
Amit Portnoy (Ben-Gurion University)
Gal Mendelson (Stanford University)
Yaniv Ben-Itzhak (VMware Research)
Michael Mitzenmacher (Harvard University)



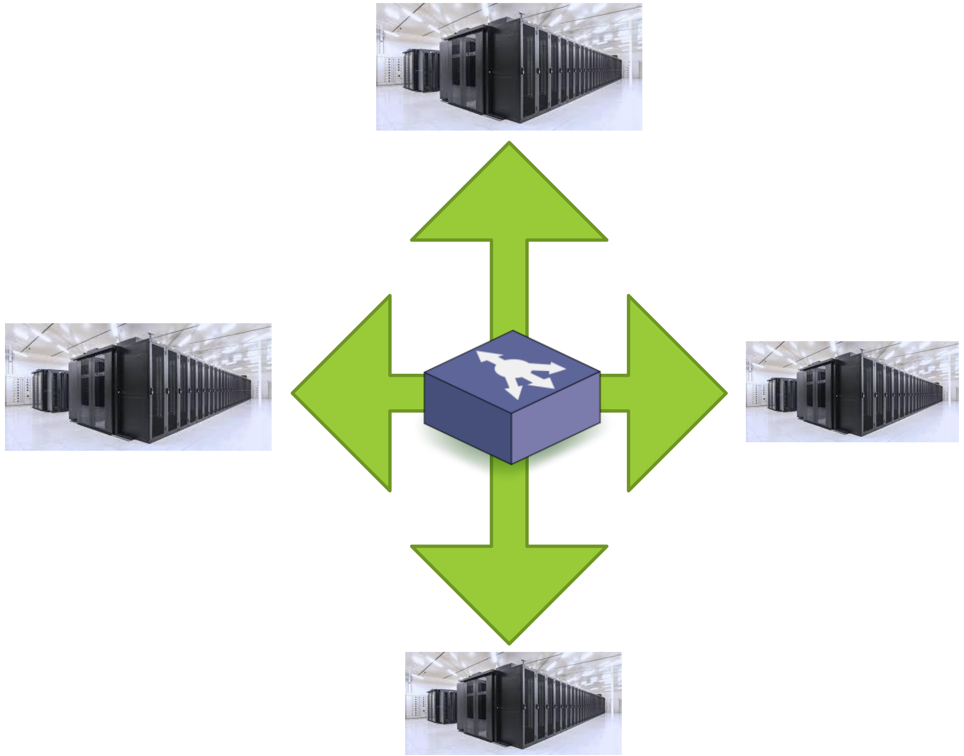
Distributed/Federated Learning



Distributed

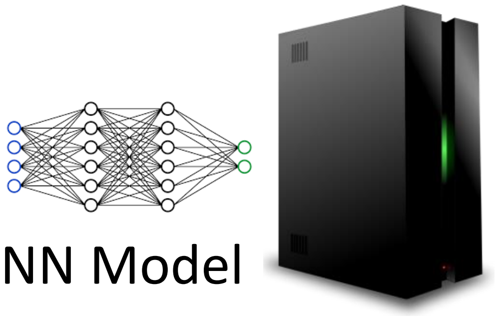


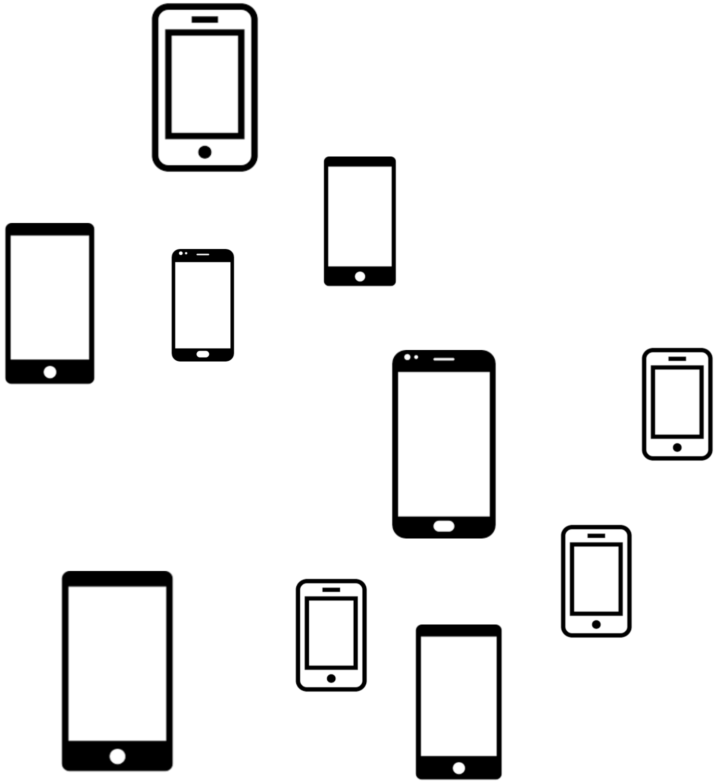
Federated – “cross device”



Federated – “cross silo”

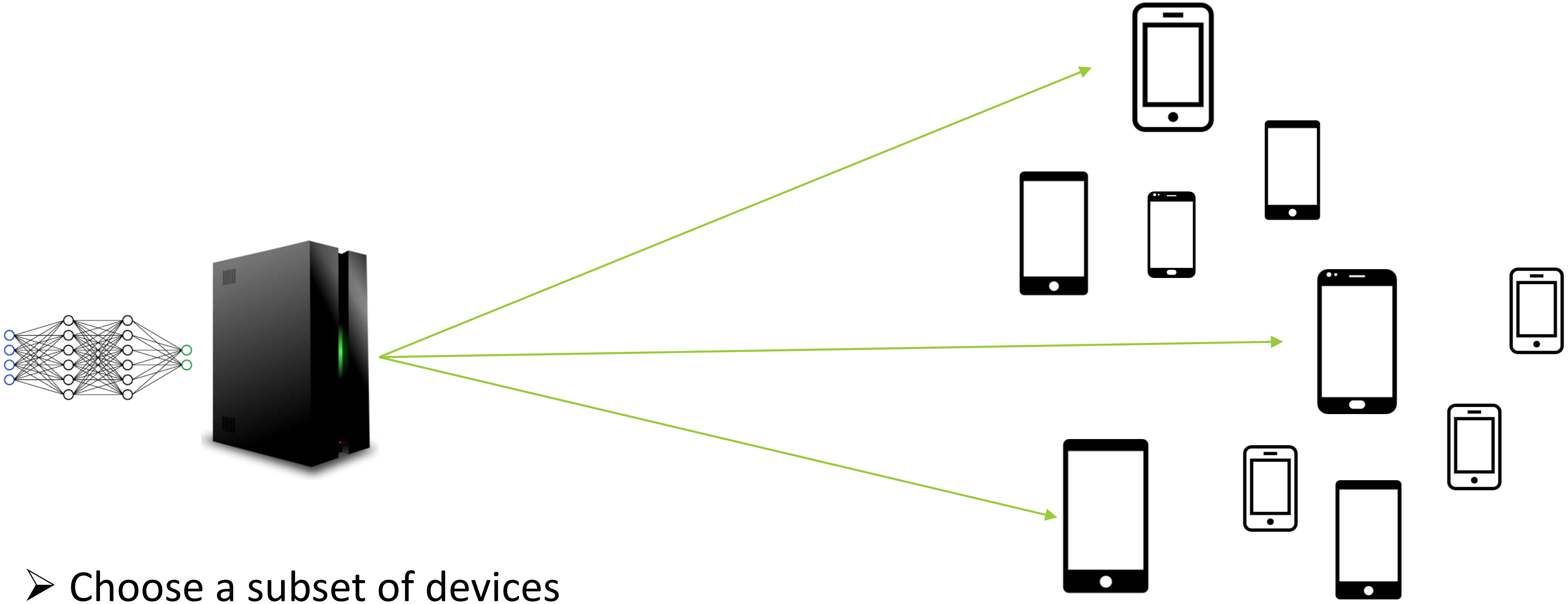
Communication Is a Bottleneck





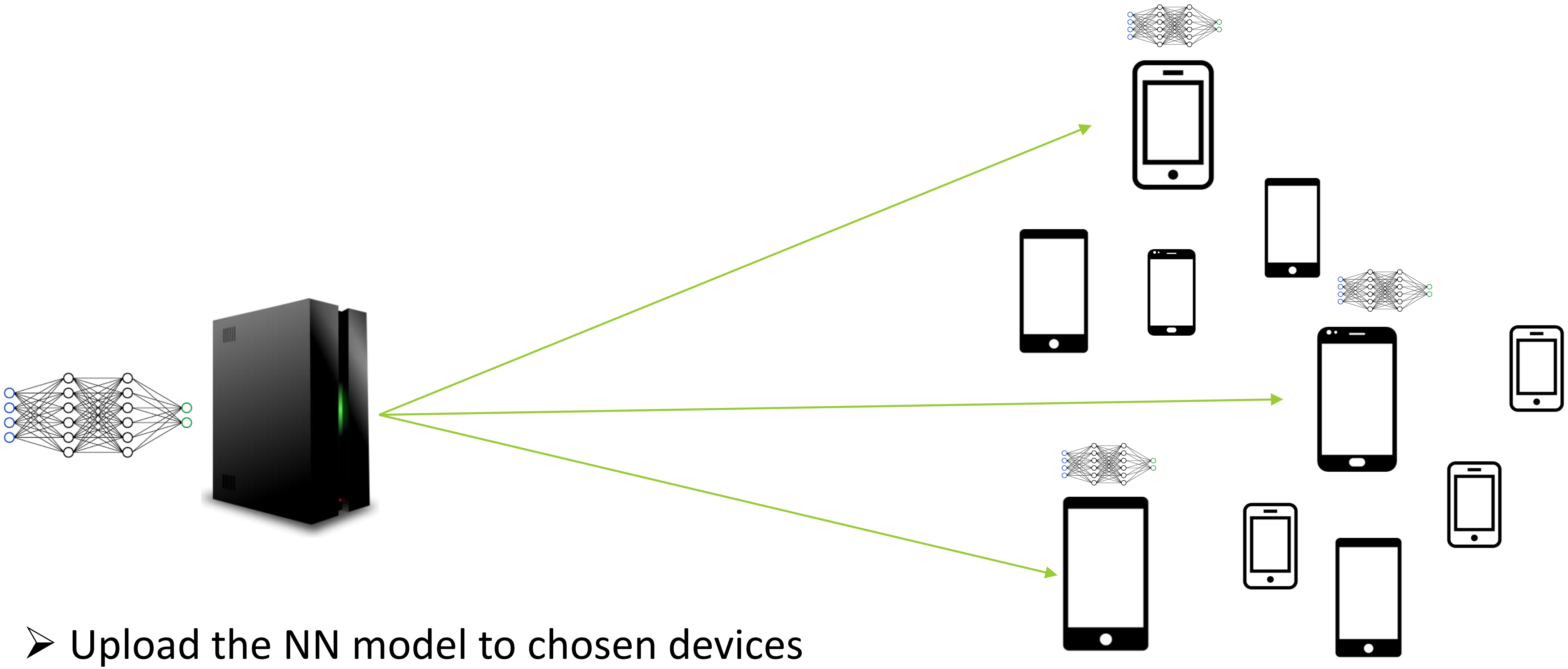
* McMaham, et al. "Communication-Efficient Learning of Deep Networks from Decentralized Data." AISTATS, 2017.

Communication Is a Bottleneck



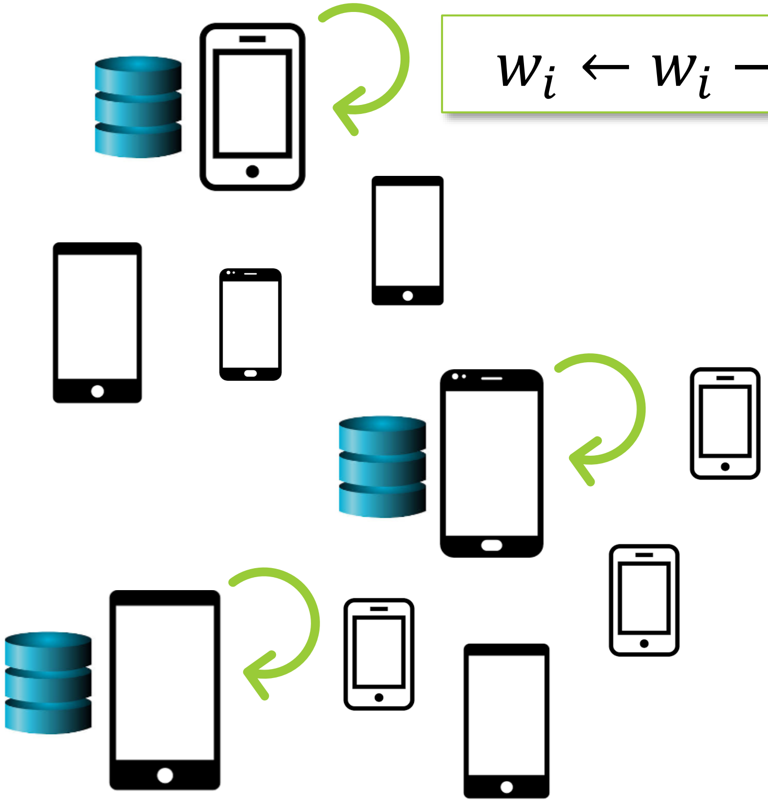
➢ Choose a subset of devices

Communication Is a Bottleneck



➢ Upload the NN model to chosen devices

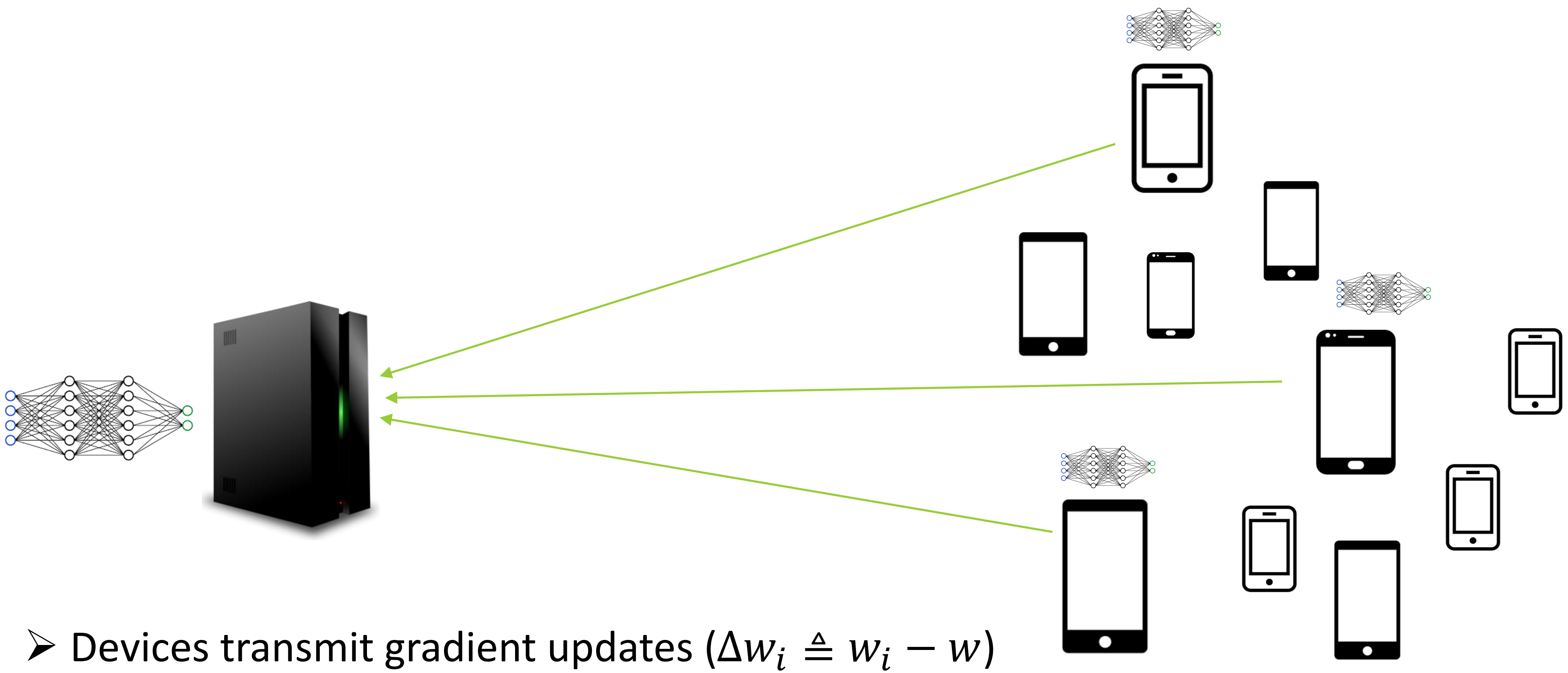
Communication Is a Bottleneck



$$w_i \leftarrow w_i - \eta \nabla \ell(w_i; b)$$

- Devices perform local training

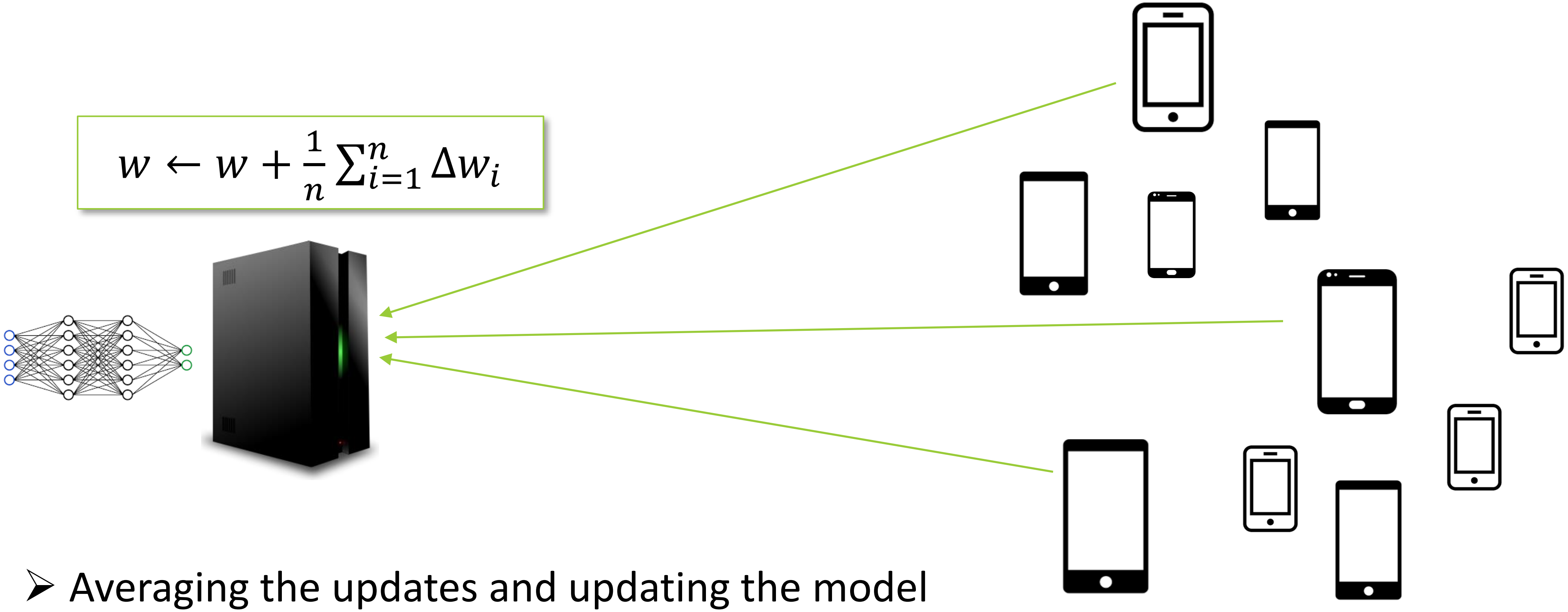
Communication Is a Bottleneck



- Devices transmit gradient updates ($\Delta w_i \triangleq w_i - w$)

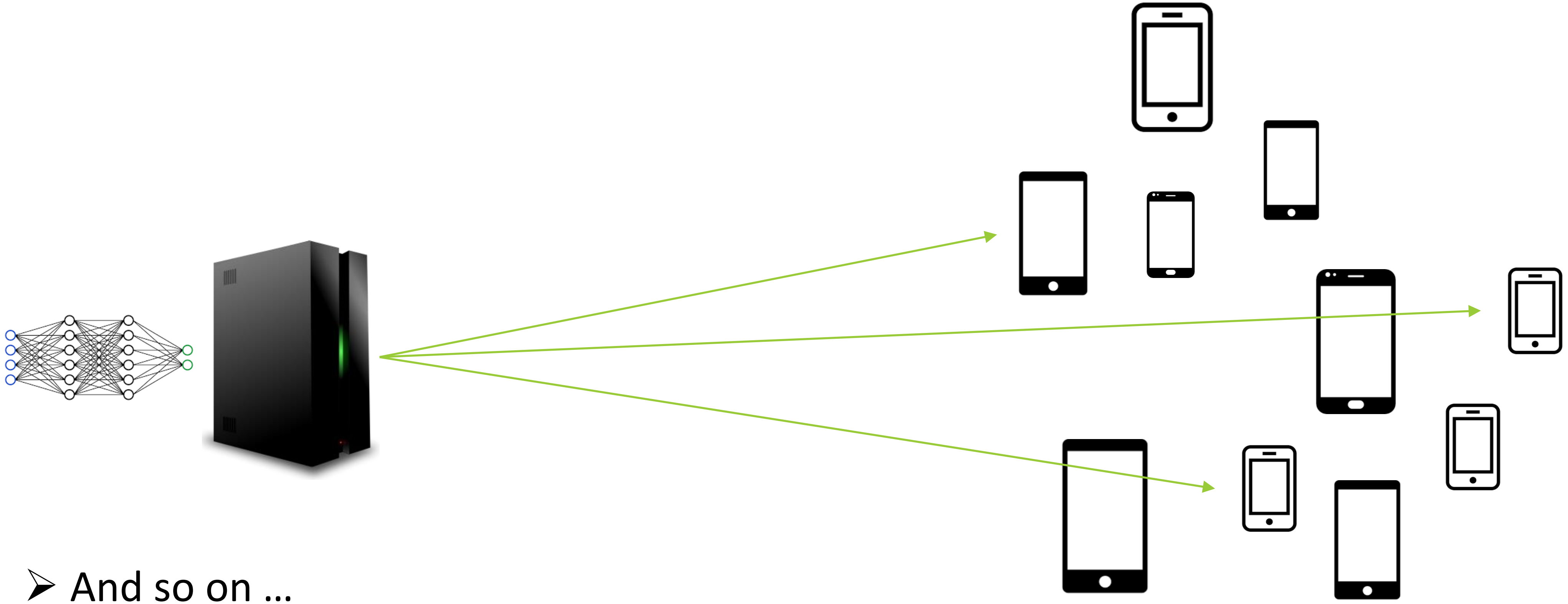
Communication Is a Bottleneck

$$w \leftarrow w + \frac{1}{n} \sum_{i=1}^{n} \Delta w_i$$



- Averaging the updates and updating the model

Communication Is a Bottleneck



➤ And so on …

Reducing Communication

- **Compression (reducing message size)**
- Increasing computation to communication ratio

Vector Estimation

$$\textbf{Minimize}: \textit{vector Normilized Mean Squred Error (vNMSE)} \triangleq \frac{\mathbb{E}\|x-\hat{x}\|_2^2}{\|x\|_2^2}$$

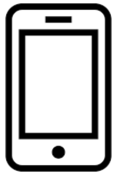
3

<u>Decompress:</u>

$\hat{x} \in \mathbb{R}^d$

2

Compressed message M



1

<u>Compress (lossy):</u>

$x \in \mathbb{R}^d$

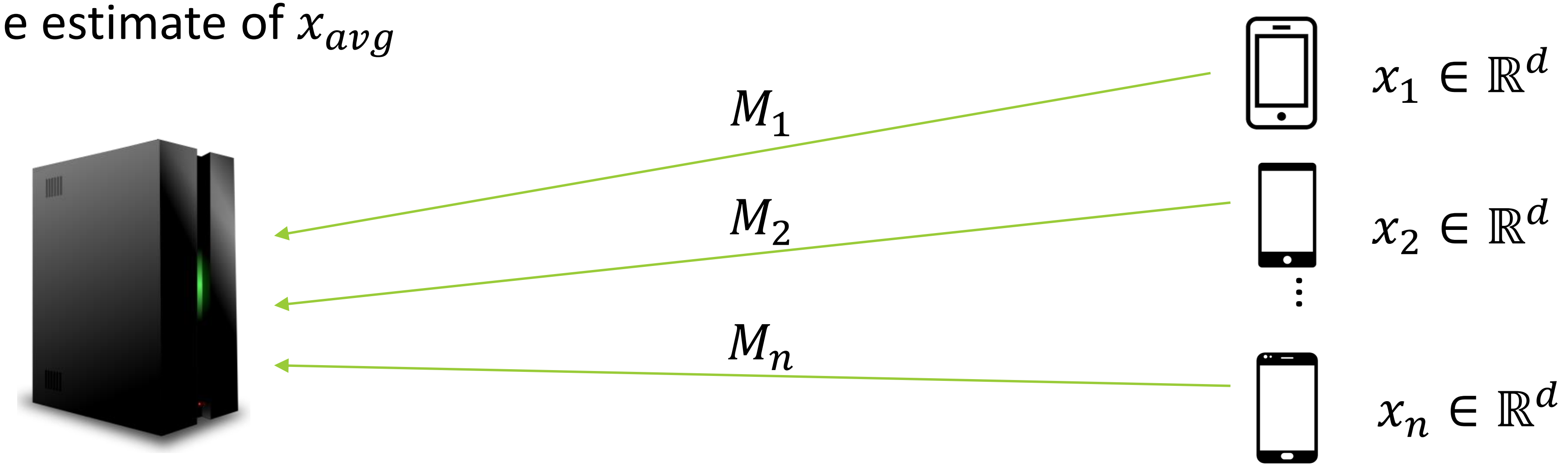
(e.g., NN gradient)

Distributed Mean Estimation

$\boldsymbol{Minimize}$: $\boldsymbol{Normilized\ Mean\ Squred\ Error\ (NMSE)} \triangleq \dfrac{\mathbb{E}\left\|x_{avg} - \widehat{x_{avg}}\right\|_2^2}{\frac{1}{n}\sum_{i=1}^{n}\|x_i\|_2^2}$

- $x_{avg} = \frac{1}{n}\sum_{i=1}^{n} x_i$
- $\widehat{x_{avg}}$ is the estimate of x_{avg}

$\widehat{x_{avg}} \in \mathbb{R}^d$

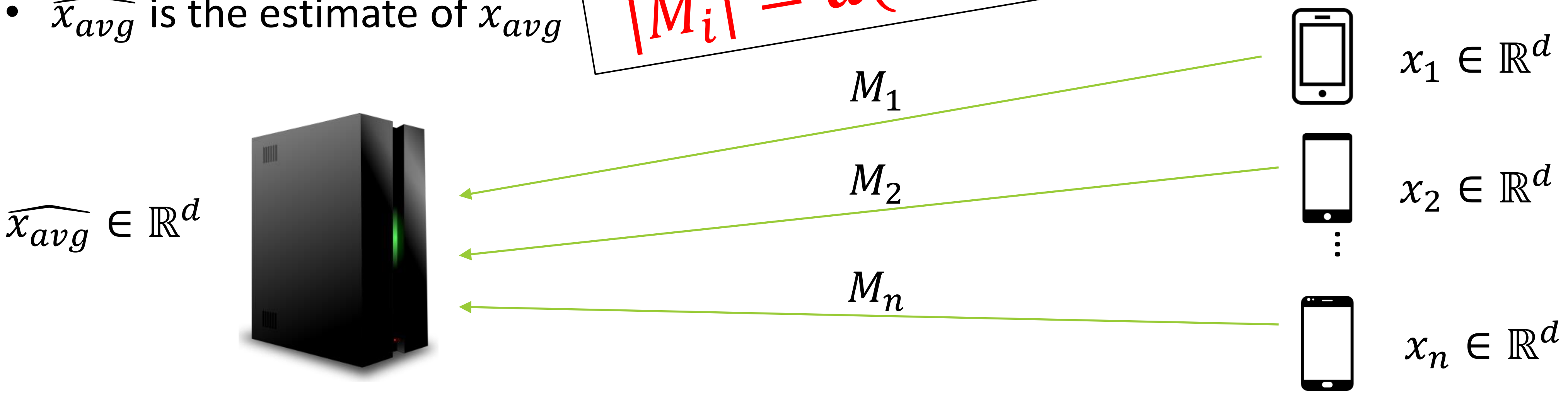


One-bit Distributed Mean Estimation

$$\boldsymbol{Minimize}\text{: } \textbf{Normilized Mean Squred Error (NMSE)} \triangleq \frac{\mathbb{E}\left\|x_{avg} - \widehat{x_{avg}}\right\|_2^2}{\frac{1}{n}\sum_{i=1}^{n}\|x_i\|_2^2}$$

- $x_{avg} = \frac{1}{n}\sum_{i=1}^{n} x_i$
- $\widehat{x_{avg}}$ is the estimate of x_{avg}

$$|M_i| = \boldsymbol{d(1 + o(1))}$$



One-bit Distributed Mean Estimation

Previous Works

	bits/client ($\boldsymbol{x_i} \in \mathbb{R}^{\boldsymbol{d}}$)	**NMSE**
[1]	$O(d)$	$O\left(\frac{\log d}{n}\right)^{(1)}, O\left(\frac{1}{n}\right)^{(2)}$
[2]	$d(1 + o(1))$	$O\left(\frac{r \cdot R}{n}\right)$
[3] (also see [4, 5])	$\lambda \cdot d\big(1 + o(1)\big),\ \lambda > 1$	$O\left(\frac{\lambda^2}{\left(\sqrt{\lambda}-1\right)^4 \cdot n}\right)$
DRIVE (this work)	$d(1 + o(1))$	$O\left(\frac{1}{n}\right)$; for $d \gg 1$: $\frac{\frac{\pi}{2} - 1}{n} \approx \frac{0.571}{n}$

[1] Suresh, et al. "Distributed mean estimation with limited communication." *ICML*, 2017.
[2] Konečný, et al. "Randomized distributed mean estimation: Accuracy vs. communication." *Frontiers in Applied Mathematics and Statistics*, 2018.
[3] Safaryan, et al. "Uncertainty principle for communication compression in distributed and federated learning and the search for an optimal compressor." *arXiv,* 2020.
[4] Caldas, et al. "Expanding the reach of federated learning by reducing client resource requirements." *arXiv,* 2018.
[5] Lyubarskii, et al. "Uncertainty principles and vector quantization." *IEEE Transactions on Information Theory*, 2010.

One-bit/coordinate in DNN training

Previous Works

- 1 bit/coordinate:
  - 1-bit SGD [INTERSPEECH, 2014]
  - SignSGD [ICML, 2018-9][ICLR, 2019]
  - SIGNUM [ICLR, 2019]
  - …
- Few bits/coordinate:
  - TernGrad [NeurIPS, 2017]
  - QSGD [NeurIPS, 2017]
  - Sketched-SGD [NeurIPS, 2019]
  - FetchSGD [ICML, 2020]
  - …

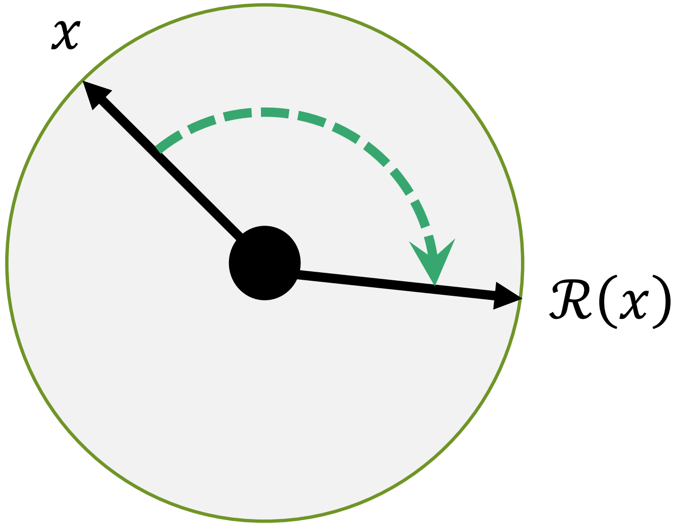
up to **32X** savings in parameter-update communication compared to non-compressed solution

Notations and Definitions
Rotation

- R is a rotation matrix ($R^{-1}R = R^T R = I$)

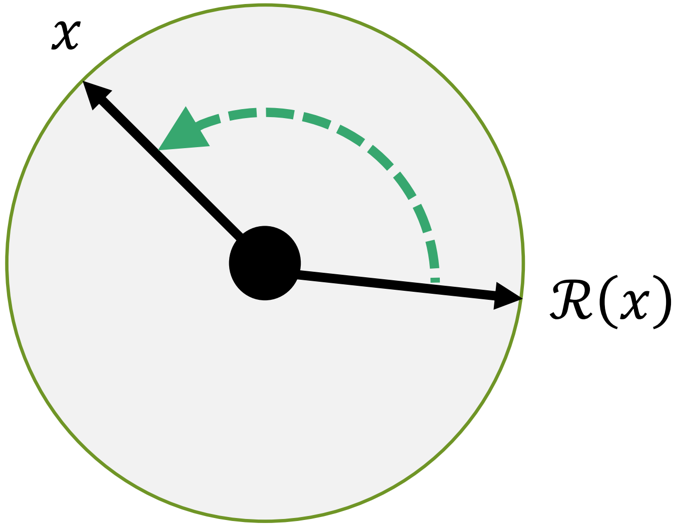
Rotation:

$$\mathcal{R}(x) \triangleq R \cdot x \in \mathbb{R}^d$$



Inverse rotation:

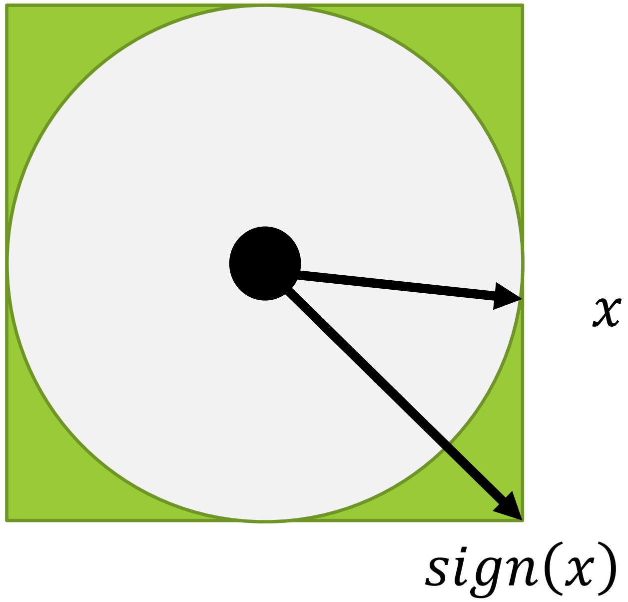
$$\mathcal{R}^{-1}\big(\mathcal{R}(x)\big) \triangleq R^T R \cdot x = x \in \mathbb{R}^d$$



Notations and Definitions

Sign

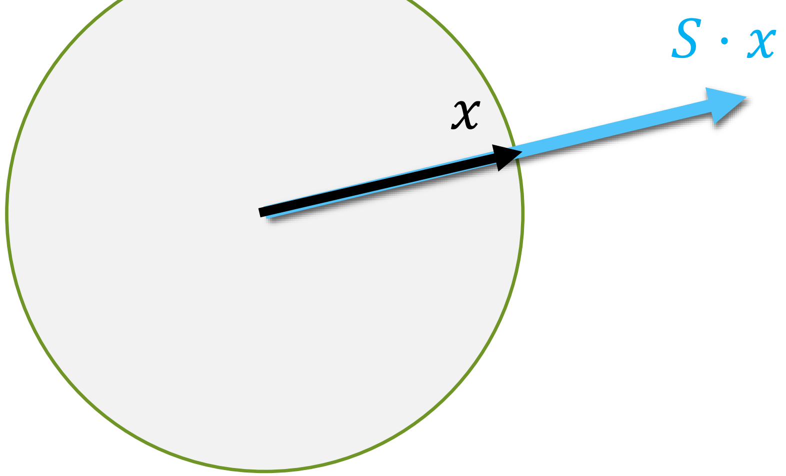
- $sign(x) \in \{-1,1\}^d$



Notations and Definitions

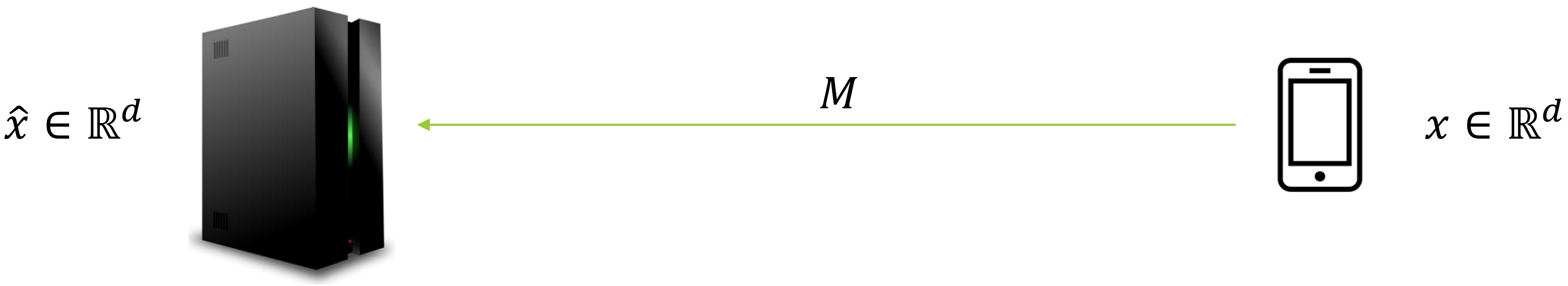
Scale

- $S \in \mathbb{R}$ is a scale



DRIVE - Deterministically RoundIng randomly rotated VEctors

- R is a rotation matrix
- $\mathcal{R}(x) \triangleq R \cdot x \in \mathbb{R}^d$
- $S \in \mathbb{R}$ is a scale
- $sign(\mathcal{R}(x)) \in \{-1,1\}^d$



DRIVE - Deterministically RoundIng randomly rotated VEctors

- R is a rotation matrix
- $\mathcal{R}(x) \triangleq R \cdot x \in \mathbb{R}^d$
- $S \in \mathbb{R}$ is a scale
- $sign(\mathcal{R}(x)) \in \{-1,1\}^d$

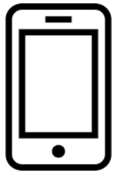
Compress:

1. $Compute: \mathcal{R}(x), S$
2. $Send: M = (sign(\mathcal{R}(x)), S)$

$\hat{x} \in \mathbb{R}^d$

M

$|M| = d(1 + o(1))$



$x \in \mathbb{R}^d$

DRIVE - Deterministically RoundIng randomly rotated VEctors

- R is a rotation matrix
- $\mathcal{R}(x) \triangleq R \cdot x \in \mathbb{R}^d$
- $S \in \mathbb{R}$ is a scale
- $sign(\mathcal{R}(x)) \in \{-1,1\}^d$

<u>Decompress:</u>

1. $Compute$: $\widehat{\mathcal{R}(x)} = S \cdot sign(\mathcal{R}(x))$
2. $Estimate$: $\hat{x} = \mathcal{R}^{-1}(\widehat{\mathcal{R}(x)})$

<u>Compute:</u>

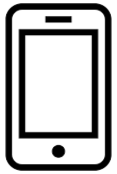
<u>Compress:</u>

1. $Compute$: $\mathcal{R}(x), S$
2. $Send$: $M = (sign(\mathcal{R}(x)), S)$

$\hat{x} \in \mathbb{R}^d$

M

$$|M| = \boldsymbol{d(1 + o(1))}$$



$x \in \mathbb{R}^d$

DRIVE's Properties

- Given $x \in \mathbb{R}^d$:
  - How to chose the rotation matrix R?
  - How to set the scale S?

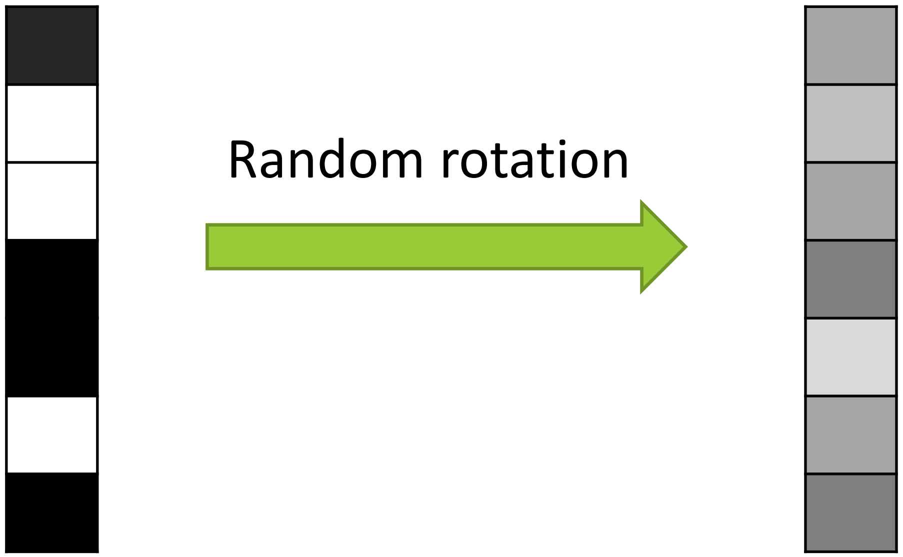
- Considerations:
  - Guarantees
  - Complexity

Intuition Behind DRIVE

Random rotation

Quantization leads to large error for unbalanced coordinates

✓ All coordinates follow the **same distribution** ($\approx \mathcal{N}\left(0, \frac{\|x\|_2^2}{d}\right)$ $for\ d \gg 1$)

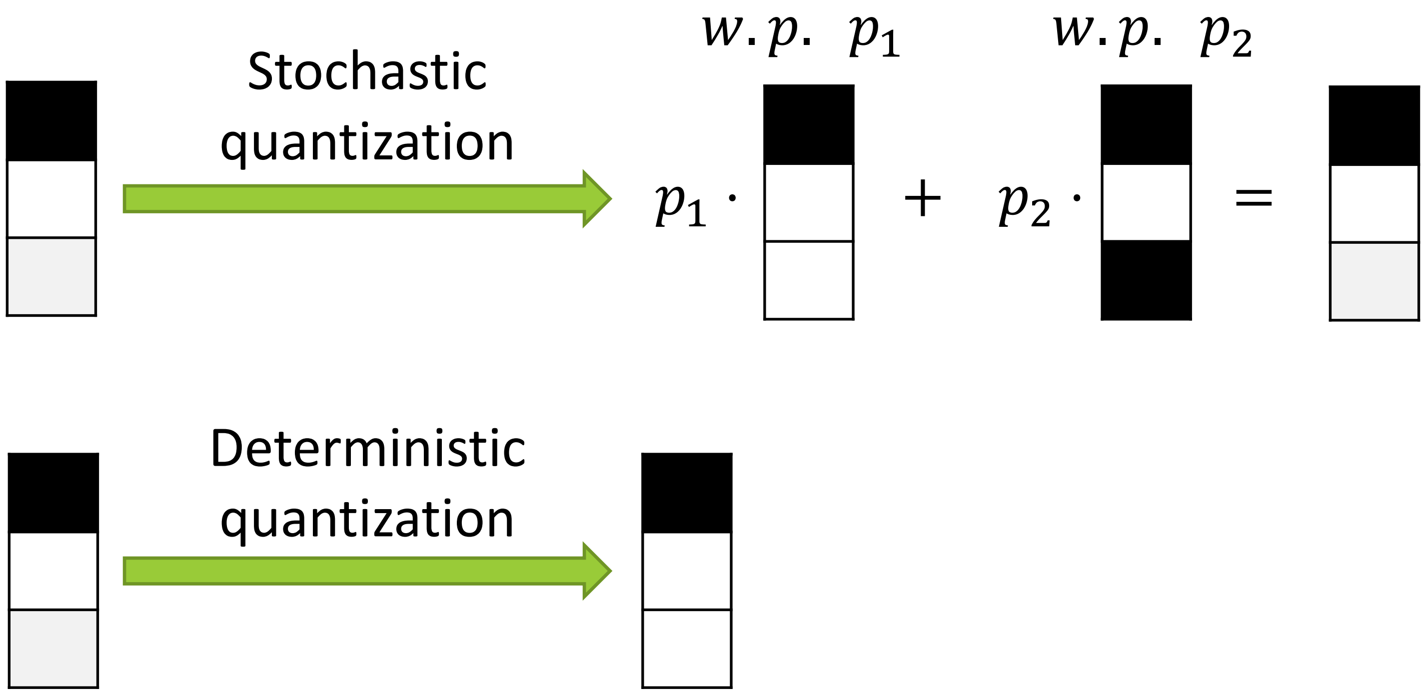


Intuition Behind DRIVE

Deterministic rounding + Rescaling

Stochastic Quantization is unbiased but leads to larger errors

- ✓ Proper rescaling can minimize error and/or make the **estimation unbiased**



DRIVE With a Uniform Random Rotation

- $\mathcal{R}_U(x)$ is uniformly distributed on a $d-1$ dimensional sphere of radius $\|x\|_2$

- Minimize $\boldsymbol{vNMSE}$. Set $\boldsymbol{S} = \frac{\|\boldsymbol{\mathcal{R}_U(x)}\|_1}{\boldsymbol{d}}$, then:
  - $\frac{\mathbb{E}\|x-\hat{x}\|_2^2}{\|x\|_2^2} = \left(1-\frac{\pi}{2}\right)\left(1-\frac{1}{\mathrm{d}}\right) < 0.3634$

DRIVE With a Uniform Random Rotation Is Unbiased With Proper Scaling

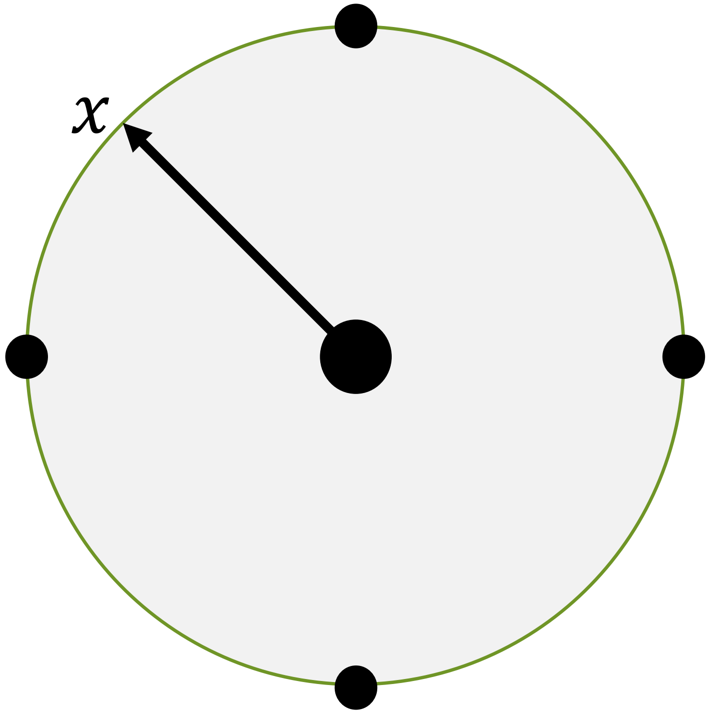
- $\mathcal{R}_U(x)$ is uniformly distributed on a $d-1$ dimensional sphere of radius $\|x\|_2$

- Minimize $\boldsymbol{NMSE}$. Set $\boldsymbol{S} = \frac{\|\boldsymbol{x}\|_2^2}{\|\boldsymbol{\mathcal{R}_U(x)}\|_1}$, then $\mathbb{E}[\hat{x}] = x$ and:
  - $\frac{\mathbb{E}\|x-\hat{x}\|_2^2}{\|x\|_2^2} \xrightarrow{(*)} \frac{\pi}{2} - 1 \approx 0.571$
  - $\frac{\mathbb{E}\left\|x_{avg}-\widehat{x_{avg}}\right\|_2^2}{\frac{1}{n}\sum_{i=1}^{n}\|x_i\|_2^2} \to \frac{0.571}{n}$

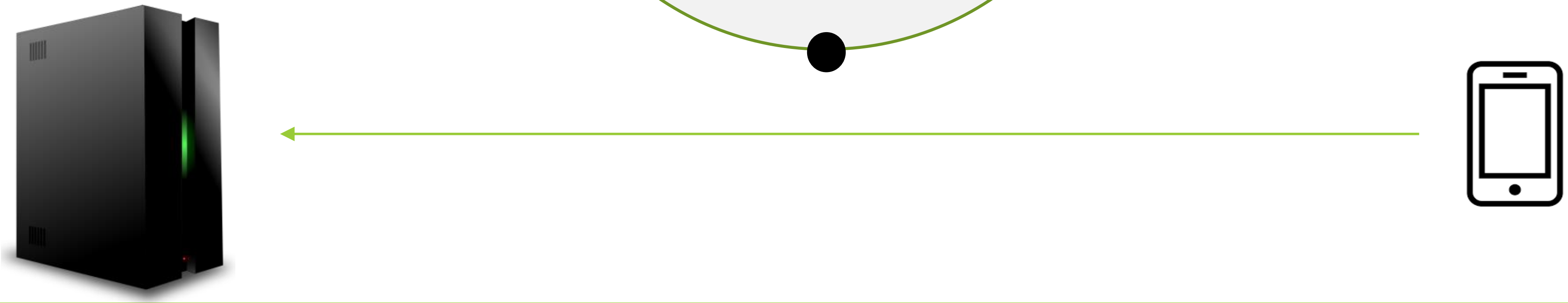
$(*)$ for $d \geq 135$, $\frac{\mathbb{E}\|x-\hat{x}\|_2^2}{\|x\|_2^2} \leq \frac{\pi}{2} - 1 + \frac{\sqrt{(6\pi^3-12\pi^2)\cdot\ln(d)+1}}{d}$

DRIVE With a Uniform Random Rotation Is Unbiased With Proper Scaling

- For ease of exposition:
  - $d = 2$
  - $sign(\mathcal{R}(x)) \in \{-1,0,1\}^d$

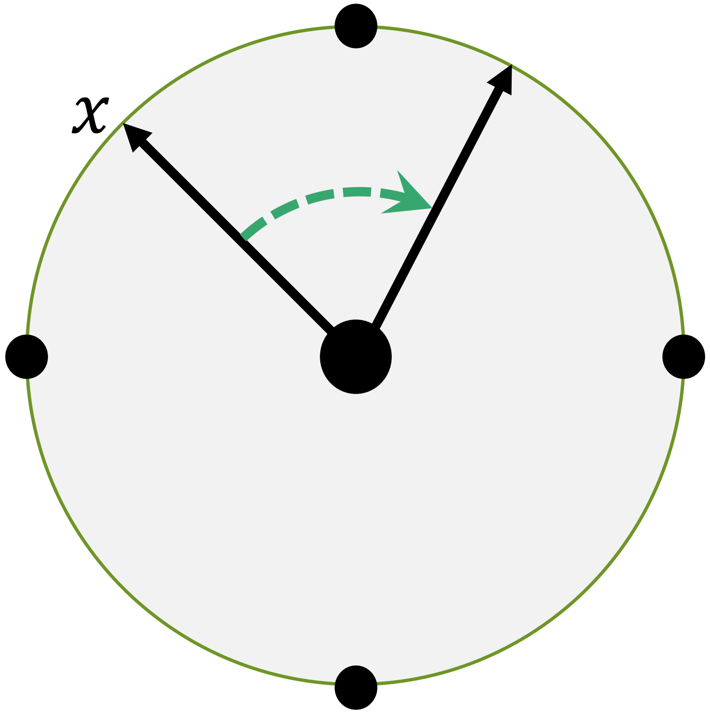


$\hat{x} \in \mathbb{R}^d$



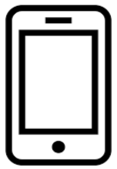
$x \in \mathbb{R}^d$

DRIVE With a Uniform Random Rotation Is Unbiased With Proper Scaling



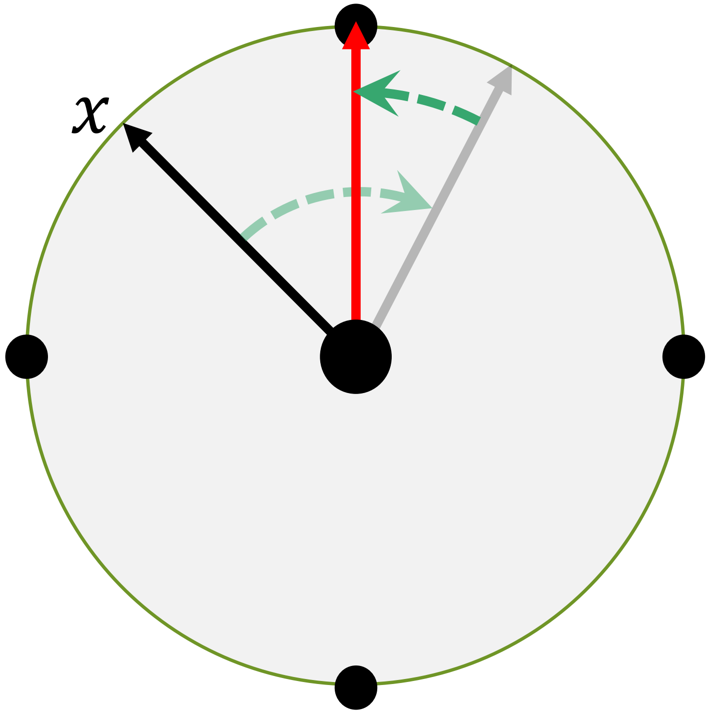
- Random rotation

$\hat{x} \in \mathbb{R}^d$



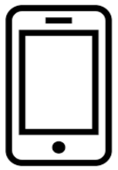
$x \in \mathbb{R}^d$

DRIVE With a Uniform Random Rotation Is Unbiased With Proper Scaling



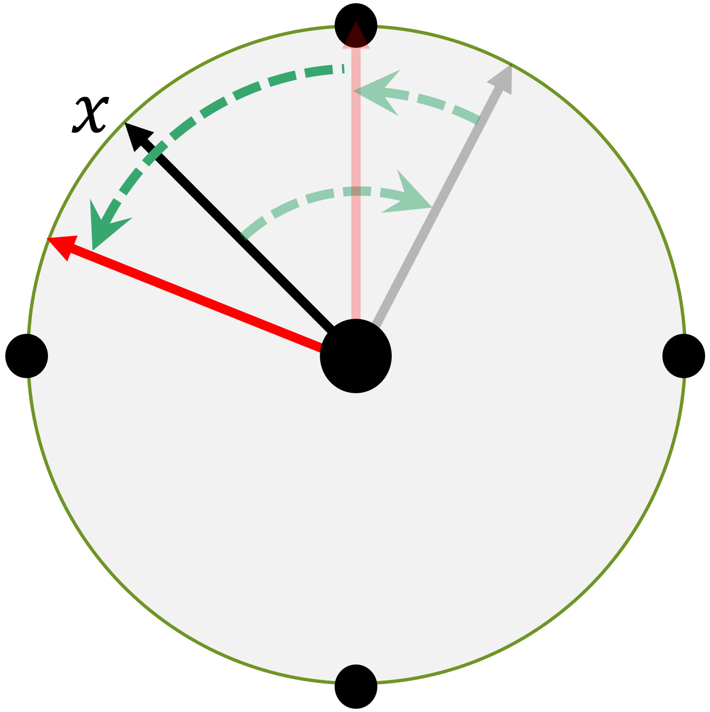
- Random rotation
- Quantization

$\hat{x} \in \mathbb{R}^d$



$x \in \mathbb{R}^d$

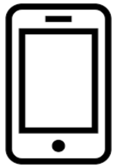
DRIVE With a Uniform Random Rotation Is Unbiased With Proper Scaling



- Inverse rotation

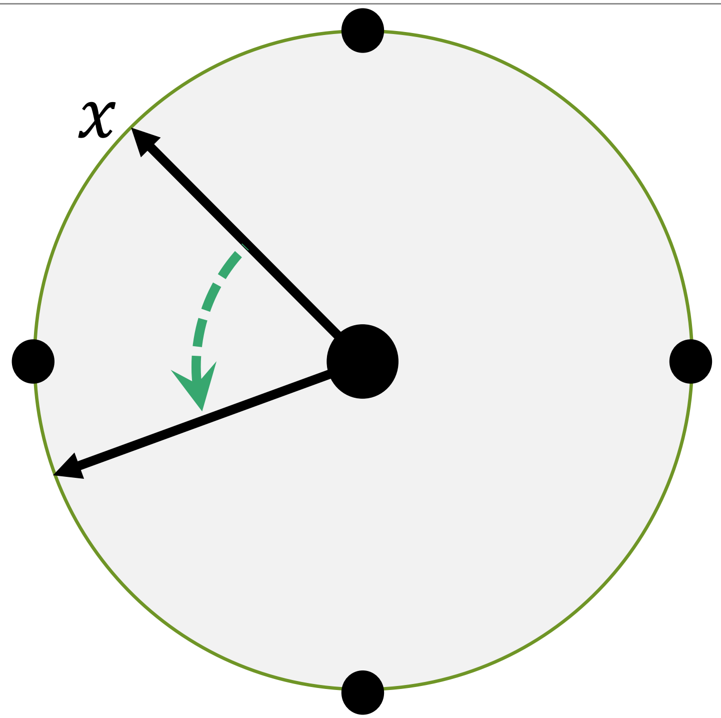
- Random rotation
- Quantization

$\hat{x} \in \mathbb{R}^d$

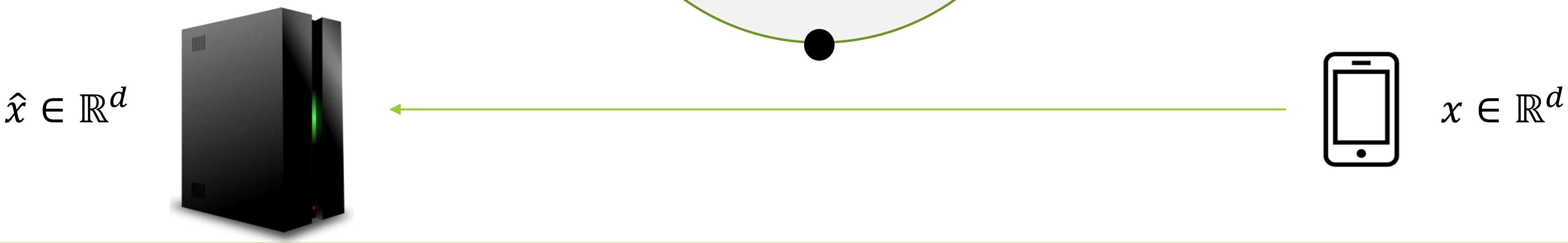


$x \in \mathbb{R}^d$

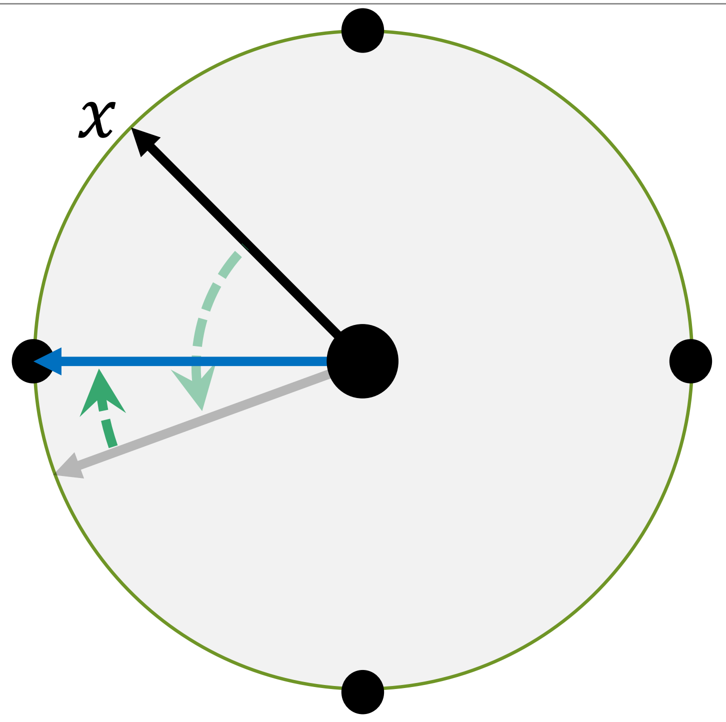
DRIVE With a Uniform Random Rotation Is Unbiased With Proper Scaling



- Random rotation

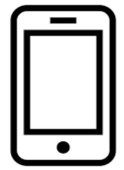


DRIVE With a Uniform Random Rotation Is Unbiased With Proper Scaling



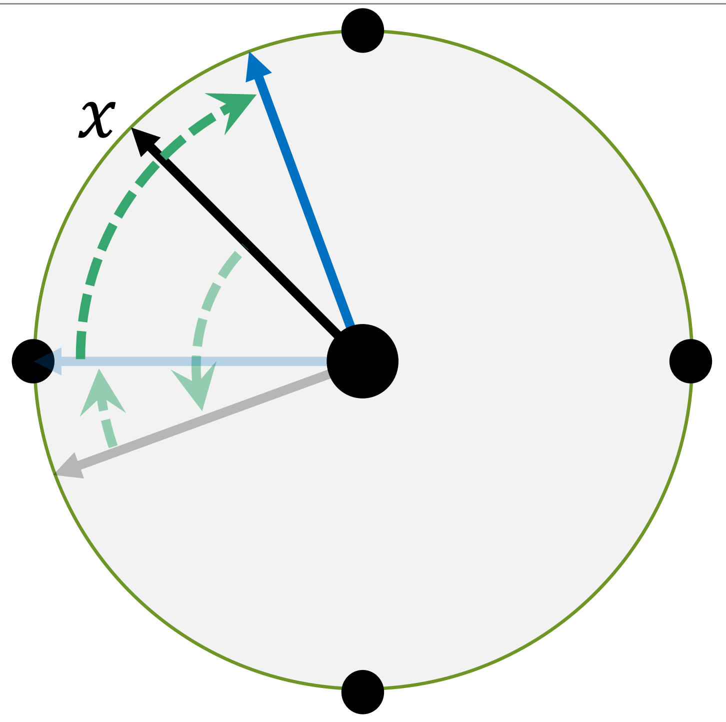
- Random rotation
- Quantization

$\hat{x} \in \mathbb{R}^d$



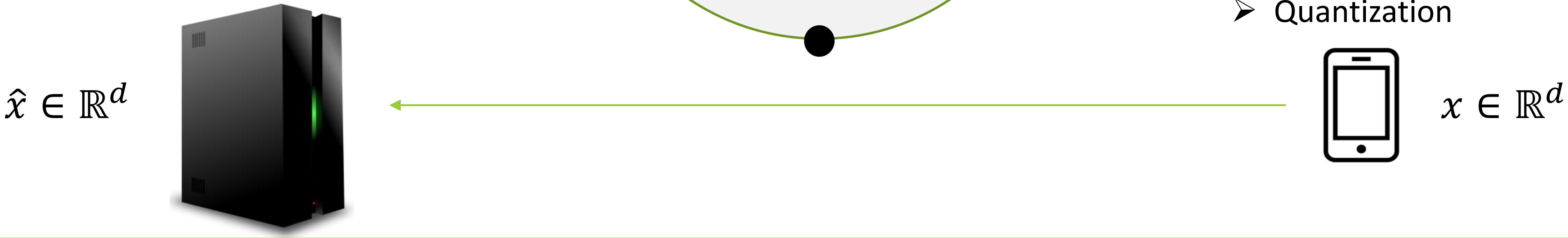
$x \in \mathbb{R}^d$

DRIVE With a Uniform Random Rotation Is Unbiased With Proper Scaling

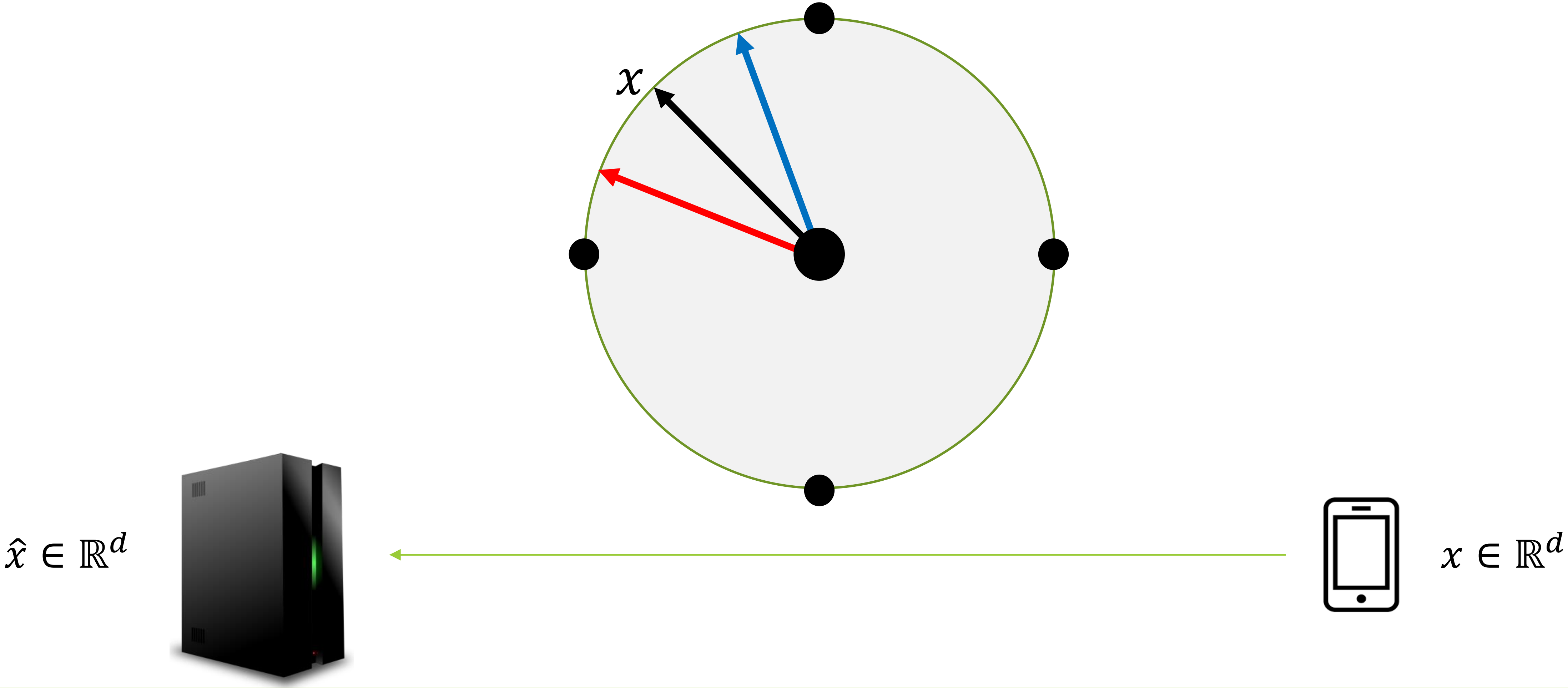


- Inverse rotation

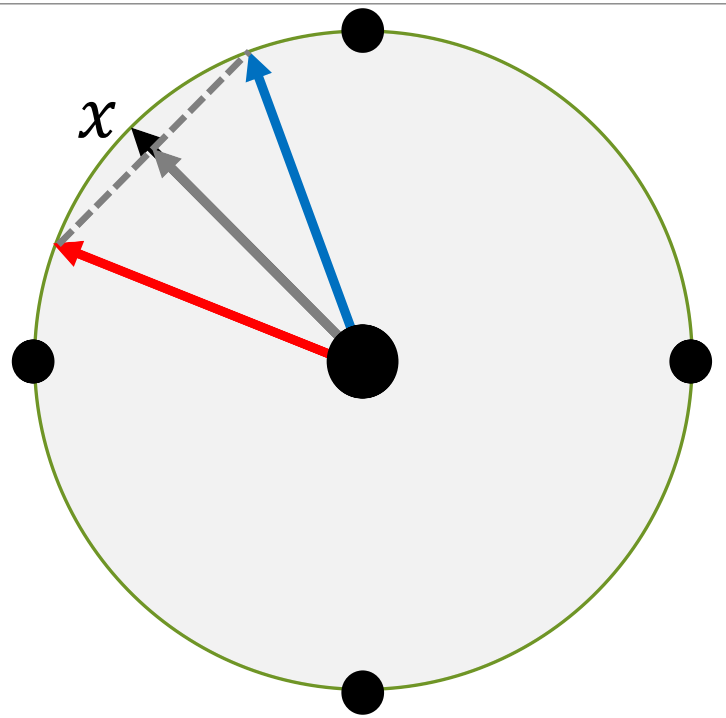
- Random rotation
- Quantization

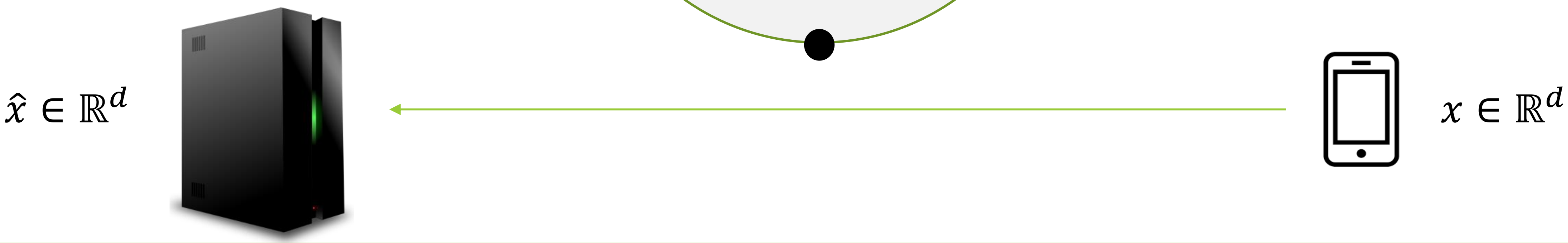


DRIVE With a Uniform Random Rotation Is Unbiased With Proper Scaling

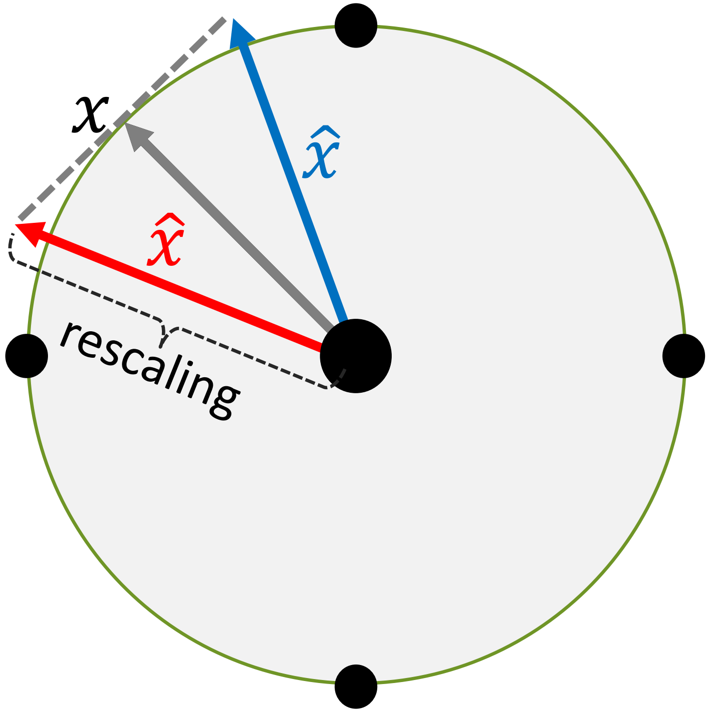


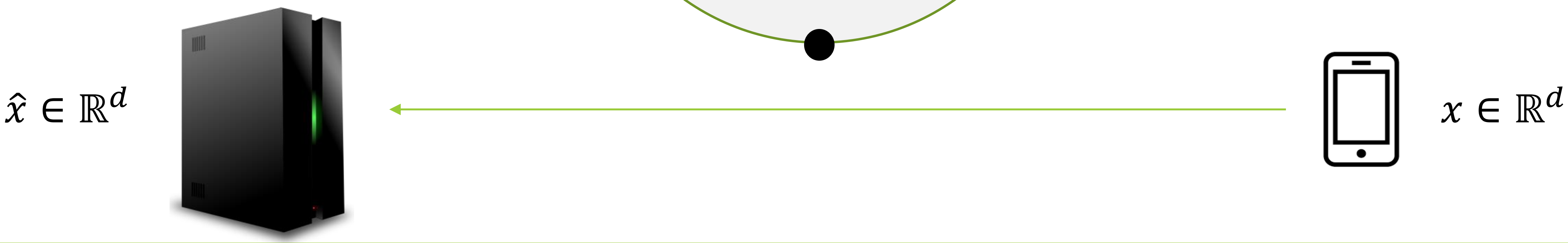
DRIVE With a Uniform Random Rotation Is Unbiased With Proper Scaling





DRIVE With a Uniform Random Rotation Is Unbiased With Proper Scaling





DRIVE With a Structured Random Rotation

Challenge: Uniform random rotation may not be sufficiently fast ($O(d^3)$)

Solution: Randomized Hadamard transform

- ✓ $O\left(d \cdot log(d)\right)$ time complexity
- ✓ GPU friendly in-place implementation
- ✓ ≈ 27 ms for $d = 33.5$ M

```python
def hadamard(vec):

  h = 2

  while h <= vec.numel():

    hf = h // 2
    vec = vec.view(vec.numel() // h, h)

    vec[:, :hf] = vec[:, :hf] + vec[:, hf:2 * hf]
    vec[:, hf:2 * hf] = vec[:, :hf] - 2 * vec[:, hf:2 * hf]

    h *= 2

  vec /= np.sqrt(vec.numel())
```

* Suresh, et al. "Distributed mean estimation with limited communication." *ICML*, 2017.

DRIVE With a Structured Random Rotation

- $\mathcal{R}_H(x)$ depends on x
- Defines a grid on a $d-1$ dimensional sphere of radius $\|x\|_2$

- Minimize $\boldsymbol{vNMSE}$. Set $S = \frac{\|\mathcal{R}_H(x)\|_1}{\mathrm{d}}$, then:
  - $\frac{\mathbb{E}\|\boldsymbol{x}-\hat{\boldsymbol{x}}\|_2^2}{\|x\|_2^2} \leq 0.5$ (instead of ≈ 0.3634 for $\mathcal{R}_U(x)$)

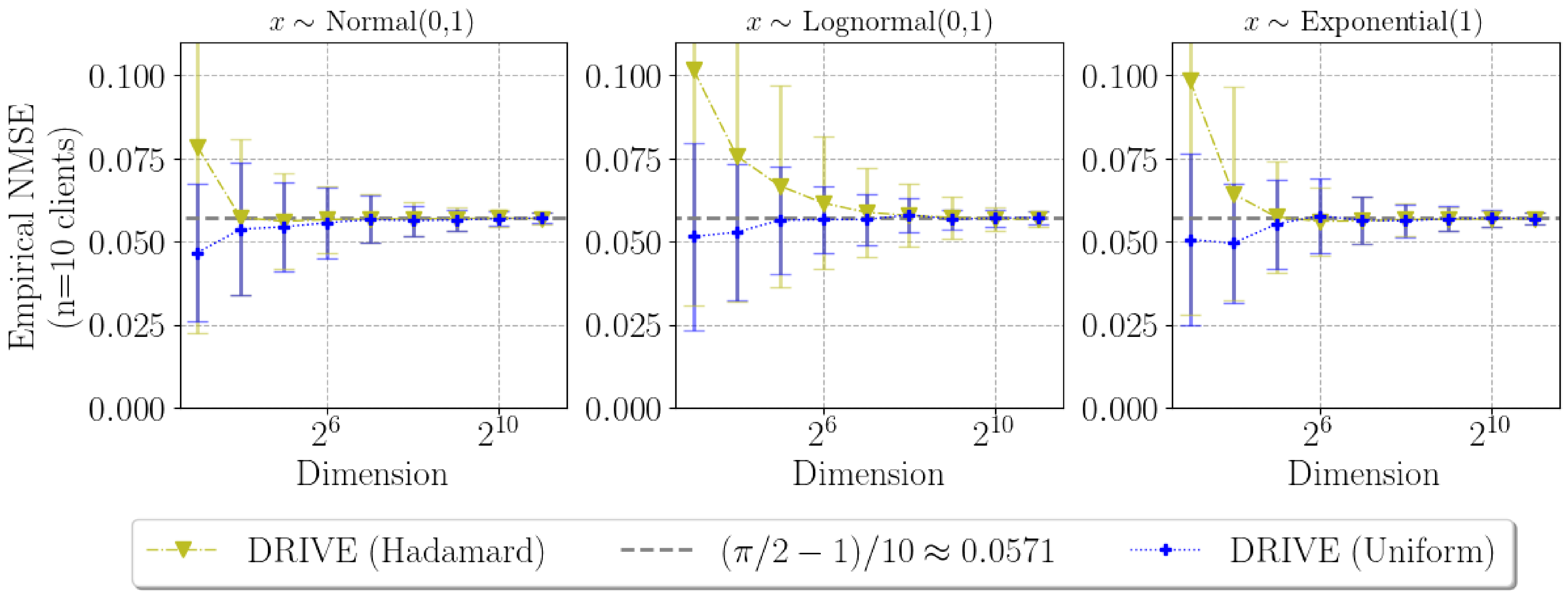
DRIVE With a Structured Random Rotation

- $\mathcal{R}_H(x)$ depends on x
- Defines a grid on a $d-1$ dimensional sphere of radius $\|x\|_2$

- Minimize $\boldsymbol{NMSE}$. Set $S = \frac{\|x\|_2^2}{\|\mathcal{R}_H(x)\|_1}$, then, if $\boldsymbol{x}$ admits finite moments:
  - For $d \gg 1$: $\mathcal{R}_H(x) \approx \mathcal{R}_U(x)$ (converge to the same moments)
  - For $d \gg 1$: $\mathbb{E}[\hat{x}] \approx x \rightarrow \frac{\mathbb{E}\left\|x_{avg} - \widehat{x_{avg}}\right\|_2^2}{\frac{1}{n}\sum_{i=1}^{n}\|x_i\|_2^2} \approx \frac{0.571}{n}$

* Chmiel, et al. "Neural gradients are near-lognormal: improved quantized and sparse training." ICLR, 2021.

DRIVE With a Structured Random Rotation



DRIVE With a Structured Random Rotation

Overhead?
PRNG seed!

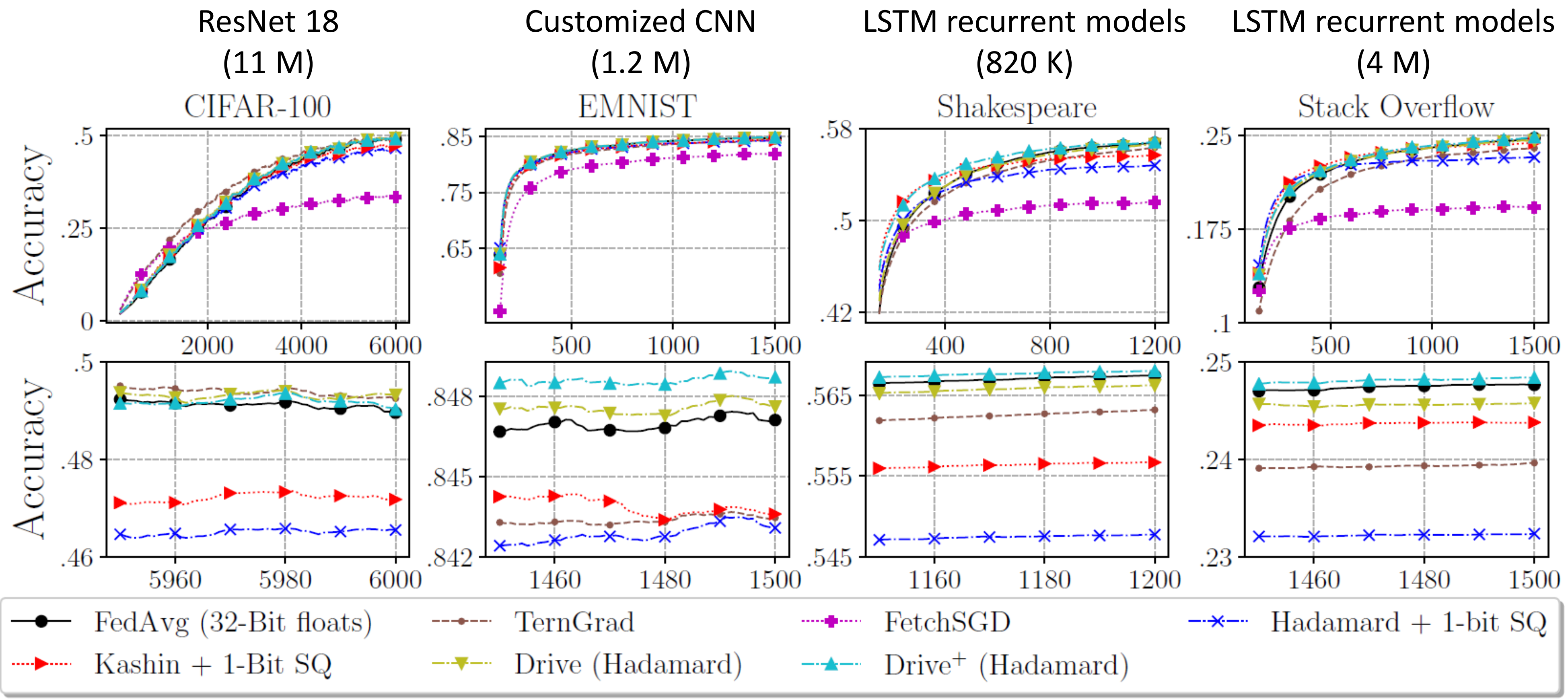
```
def drive_compress(vec, prng):

    ### randomize vec signs
    radamacher_diagonal = 2 * torch.bernoulli(torch.ones(vec.numel(), device=vec.device) / 2, generator=prng) - 1
    vec = vec * radamacher_diagonal

    ### in-place Hadamard transform
    hadamard_rotate(vec)

    ### compute the scale
    scale = torch.norm(vec,2)**2 / torch.norm(vec,1)

    ### compute the sign of the rotated vector
    sign_rvec = 1.0-2*(vec<0)

    #### send the sign vector of the rotated vector and its scale
    return sign_rvec, scale

def drive_decompress(compressed_vec, scale, prng):

    ### in-place Hadamard transform (inverse)
    hadamard_rotate(compressed_vec)

    ### restore vec signs using a mathcing radamacher diagonal - uses the same PRNG seed as the sender
    radamacher_diagonal = 2 * torch.bernoulli(torch.ones(vec.numel(), device=vec.device) / 2, generator=prng) - 1
    compressed_vec = compressed_vec * radamacher_diagonal

    ##### scale and return
    return scale * compressed_vec
```

More in the Paper

- DRIVE^{+} - further reduces vMNSE (especially for low dimensions)
- More evaluation vs. SOTA techniques :
  - NMSE and encoding speeds over different GPUs
  - Distributed Learning (CNNs)
  - Distributed K-means (Lloyd's algorithm)
  - Distributed Power-iteration (e.g., PCA)
- Compatibility with EF
- Entropy encoding

Evaluation

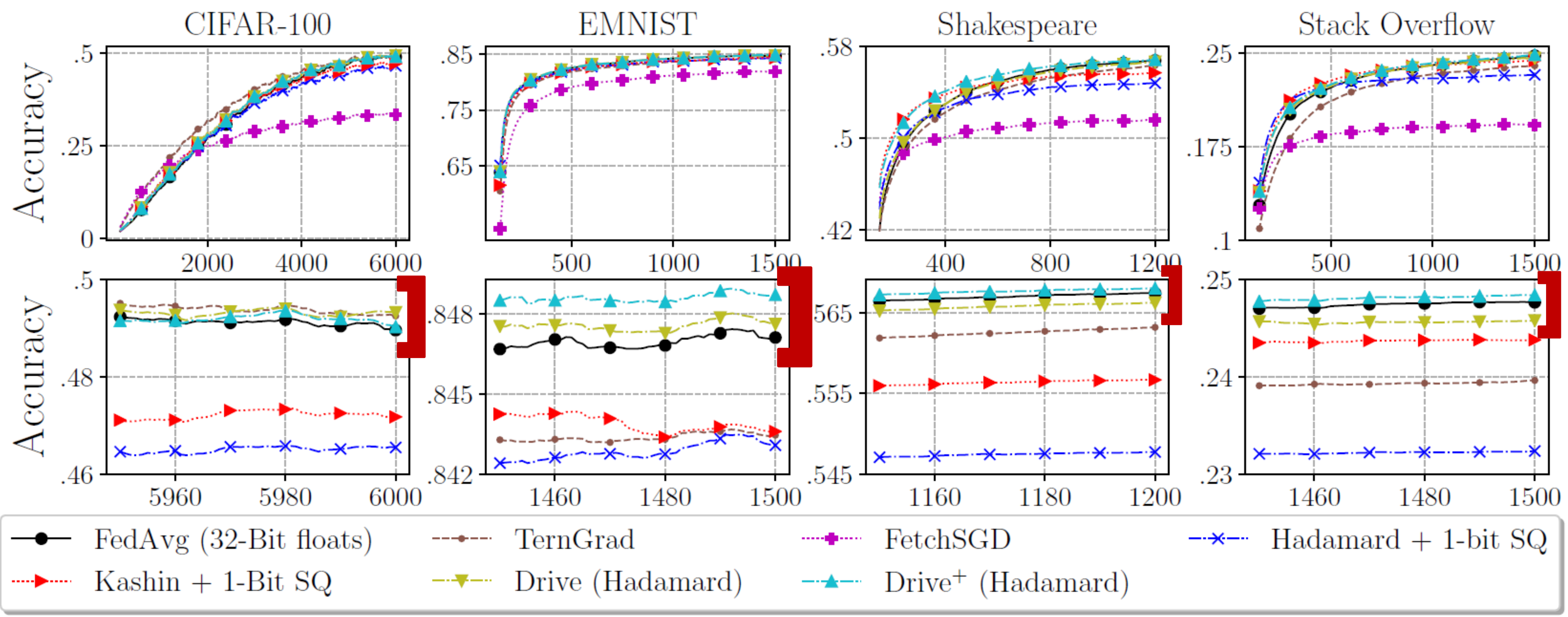
Federated learning



* Evaluation inspired by: Reddi, et al. "Adaptive Federated Optimization." ICLR, 2021.

Evaluation

Federated learning



Our Results Are Reproducible

- DRIVE's code is available in:
  - https://github.com/amitport/DRIVE-One-bit-Distributed-Mean-Estimation

- All simulations in the paper
- Stand-alone PyTorch implementation
- Stand-alone TensorFlow implementation

Future Work

[arXiv] Extend DRIVE to other settings:

- https://arxiv.org/pdf/2108.08842.pdf

Communication-Efficient Federated Learning via Robust Distributed Mean Estimation

Shay Vargaftik* — VMware Research
Ran Ben Basat* — University College London
Amit Portnoy* — Ben-Gurion University
Gal Mendelson — Stanford University
Yaniv Ben-Itzhak — VMware Research
Michael Mitzenmacher — Harvard University

[ICALP21'] Push the boundary of shared randomness:

- https://drops.dagstuhl.de/opus/volltexte/2021/14094/pdf/LIPIcs-ICALP-2021-25.pdf

How to Send a Real Number Using a Single Bit (and Some Shared Randomness)

Ran Ben Basat
University College London
Michael Mitzenmacher
Harvard University
Shay Vargaftik
VMware Research

Thank You!

And enjoy your DRIVE!