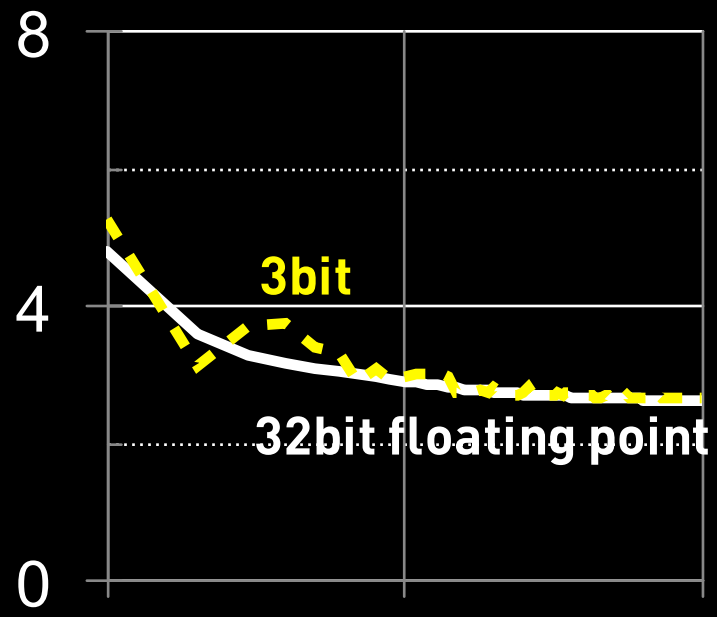
ZipML

Faster Machine Learning via Low-Precision Communication & Computation

Dan Alistarh (IST Austria & ETH Zurich),
Hantian Zhang (ETH Zurich)

How many bits do you need to represent
a single number in machine learning systems?

Takeaways



Training Neural Networks
4 bits is enough for **communication**

Training Linear Models
4 bits is enough **end-to-end**

Beyond Empirical
Rigorous theoretical guarantees

First Example: GPUs

- GPUs have plenty of compute
- Yet, **bandwidth relatively limited**
- **PCIe** or (newer) **NVLINK**

Trend towards large models and datasets

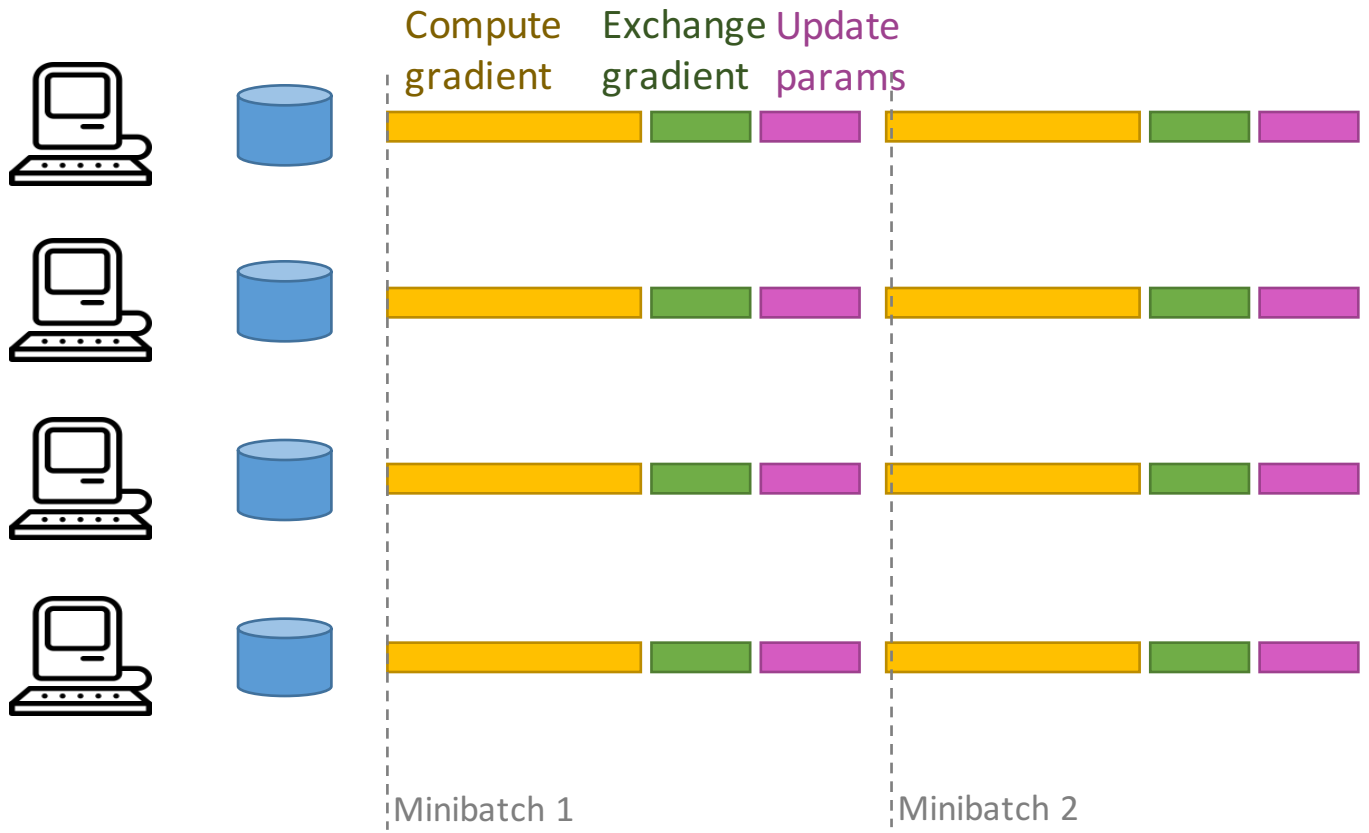
- **Vision**: ImageNet (**1.8M images)**
- ResNet-152 model [He+15]:
  60M parameters (~240 MB)

- **Speech**: NIST2000 **2000 hours**
- LACEA [Yu+16]:
  65M parameters (~300 MB)

Gradient transmission is **expensive**.

What happens in practice?

Regular model



First Example: GPUs

- GPUs have plenty of compute
- Yet, **bandwidth relatively limited**
- **PCIe** or (newer) **NVLINK**

General trend towards large models

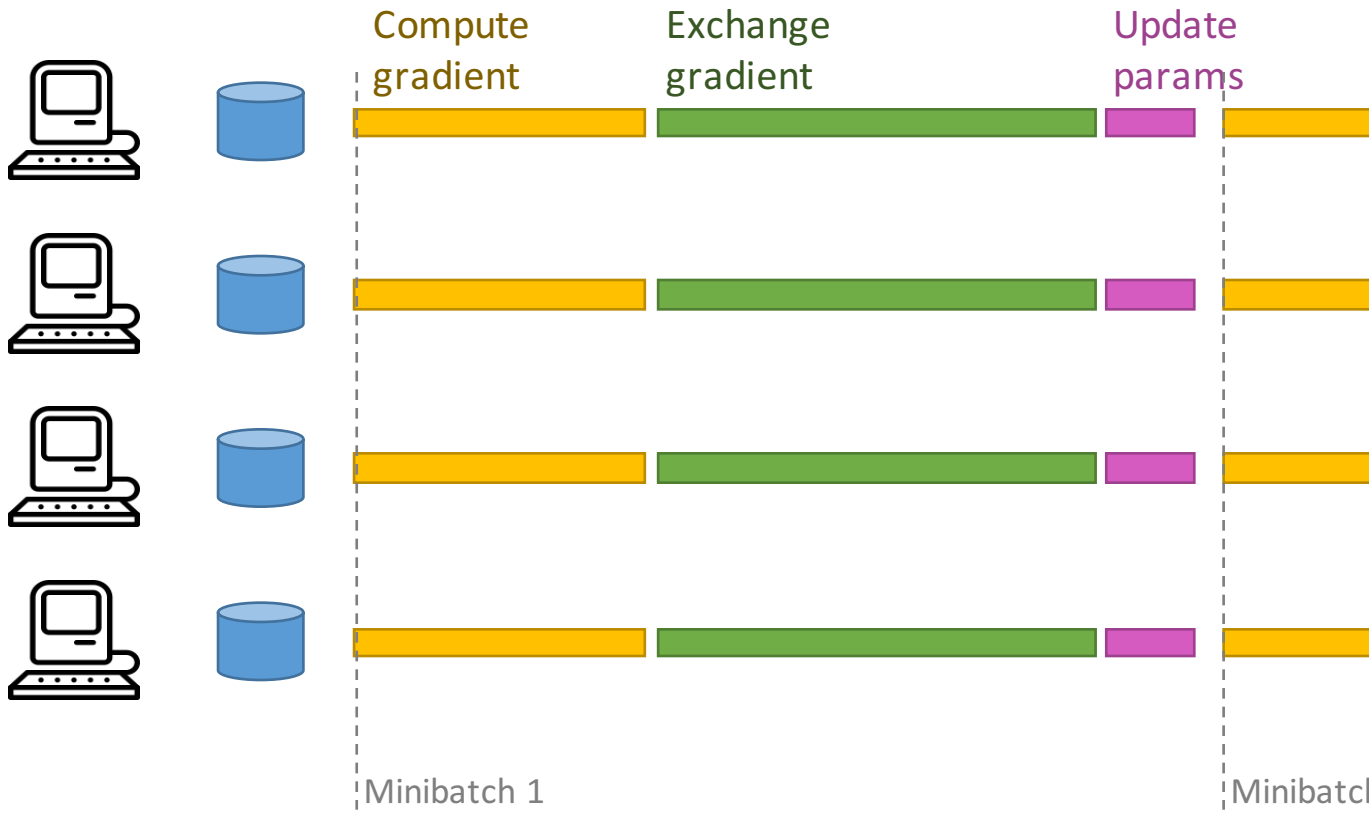
- **Vision**: ImageNet (**1.8M images)**
- ResNet-152 model [He+15]:
  60M parameters (~240 MB)

- **Speech**: NIST2000 **2000 hours**
- LACEA [Yu+16]:
  65M parameters (~300 MB)

Gradient transmission is **expensive**.

What happens in practice?

Bigger model



First Example: GPUs

- GPUs have plenty of compute
- Yet, **bandwidth relatively limited**
- **PCIe** or (newer) **NVLINK**

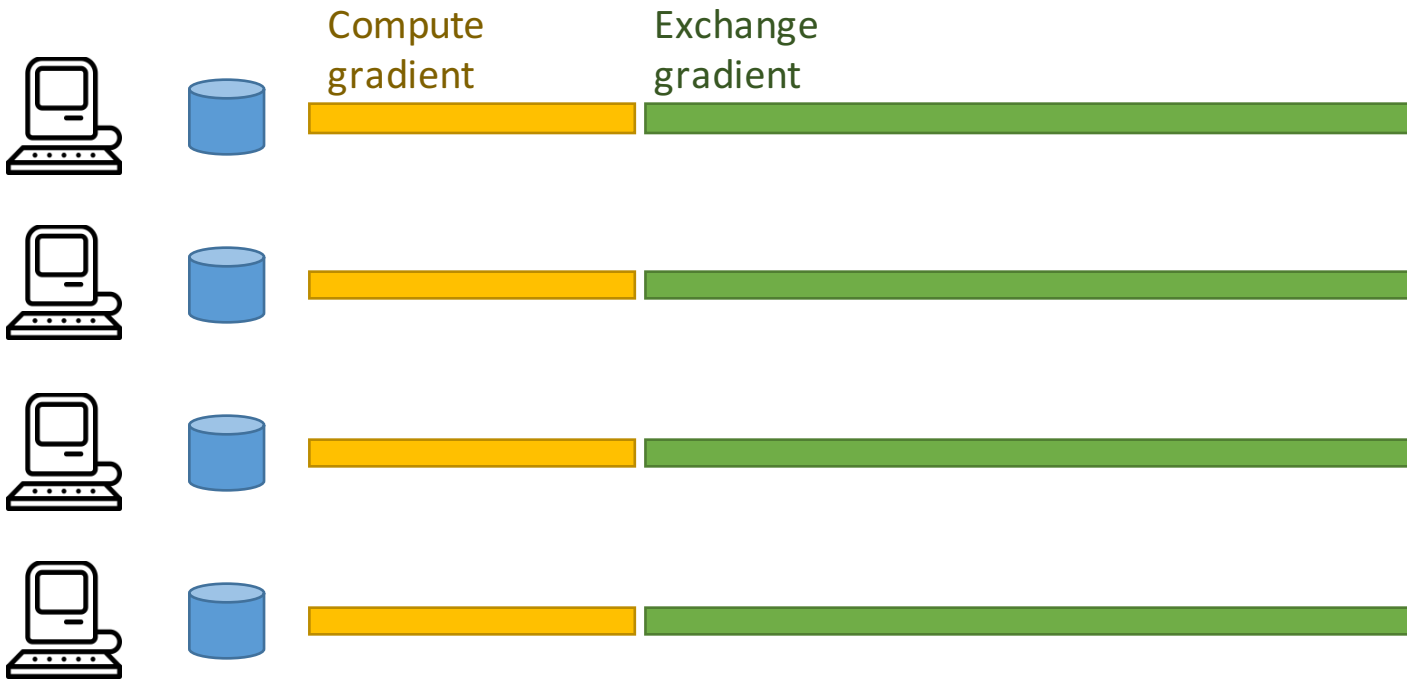
General trend towards large models

- **Vision**: ImageNet (**1.8M images)**
- ResNet-152 model [He+15]:
  60M parameters (~240 MB)

- **Speech**: NIST2000 **2000 hours**
- LACEA [Yu+16]:
  65M parameters (~300 MB)

Gradient transmission is **expensive**.

What happens in practice?
Biggerer model



First Example: GPUs

- GPUs have plenty of compute
- Yet, **bandwidth relatively limited**
- **PCIe** or (newer) **NVLINK**

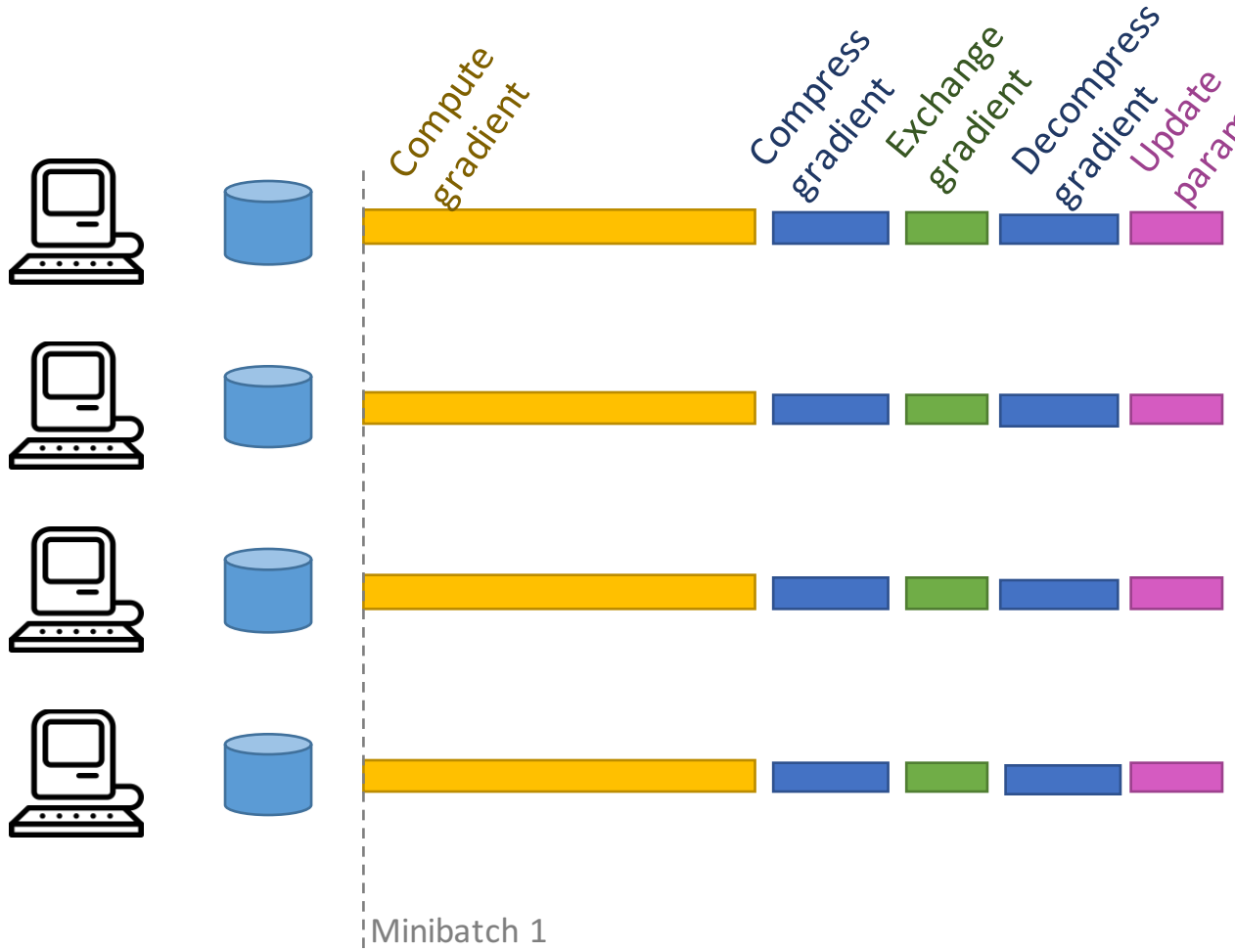
General trend towards large models

- **Vision**: ImageNet (**1.8M images)**
- ResNet-152 model [He+15]:
  60M parameters (~240 MB)

- **Speech**: NIST2000 **2000 hours**
- LACEA [Yu+16]:
  65M parameters (~300 MB)

Gradient transmission is **expensive**.

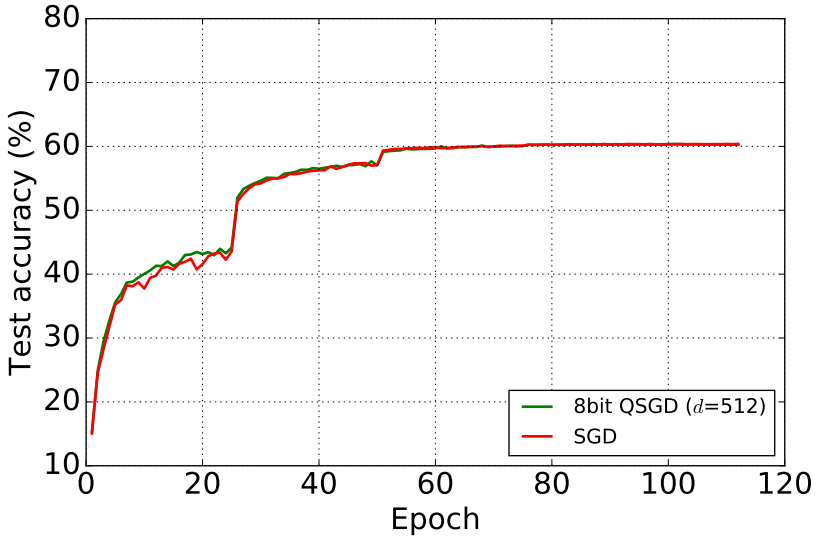
Compression [Seide et al., Microsoft CNTK]



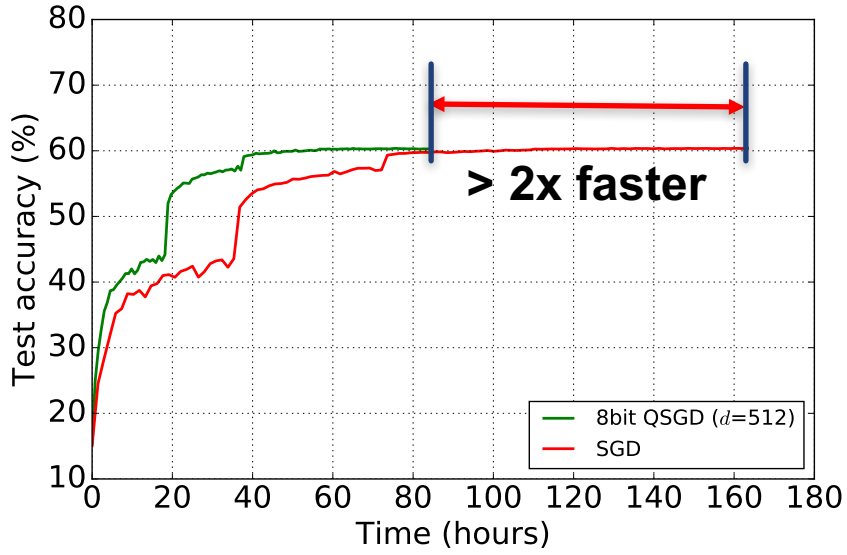
The Key Question

Can lossy compression provide *speedup*, while preserving *convergence*?

Yes. *Quantized* SGD (QSGD) can converge as fast as SGD, with considerably less bandwidth.



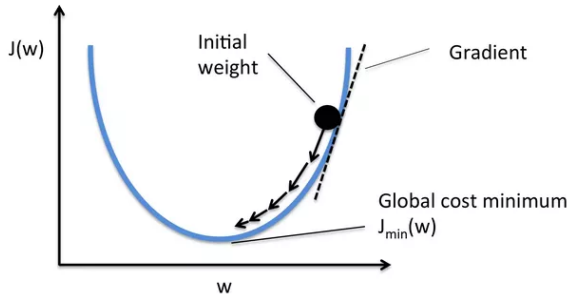
Top-1 accuracy for AlexNet (ImageNet).



Top-1 accuracy vs Time for AlexNet (ImageNet).

Why does QSGD work?

Notation in One Slide

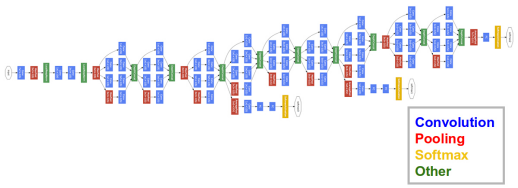


$$\text{argmin}_x f(x)$$

Solved via optimization procedure.

$$f(x) = (1/M) \sum_{i=1}^{M} loss(x, ei)$$

Notion of “quality”



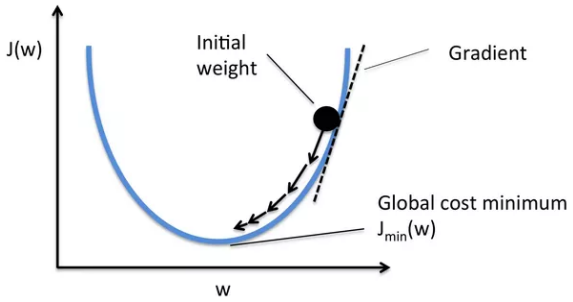
Model x

Task

E.g., image classification

Data
(M examples)

Background on Stochastic Gradient Descent



- *Stochastic* Gradient Descent:

  Goal: find $\text{argmin}_x f(x)$.

  Let $\tilde{g}(x)$ = x's gradient at a *randomly chosen data point*.

  Iteration:

$$x_{t+1} = x_t - \eta_t \tilde{g}(xt), \quad \text{where } E[\tilde{g}(xt)] = \nabla f(x_t).$$

Let $E\left[\left|\left|\tilde{g}(x) - \nabla f(x)\right|\right|^2\right] \leq \sigma^2$ **(variance bound)**

Theorem [**Informal**]: Given f **nice** (e.g., **convex** and **smooth)**, and $R^2 = ||x_0 - x^*||^2$.

To **converge within** ε of optimal it is sufficient to run for

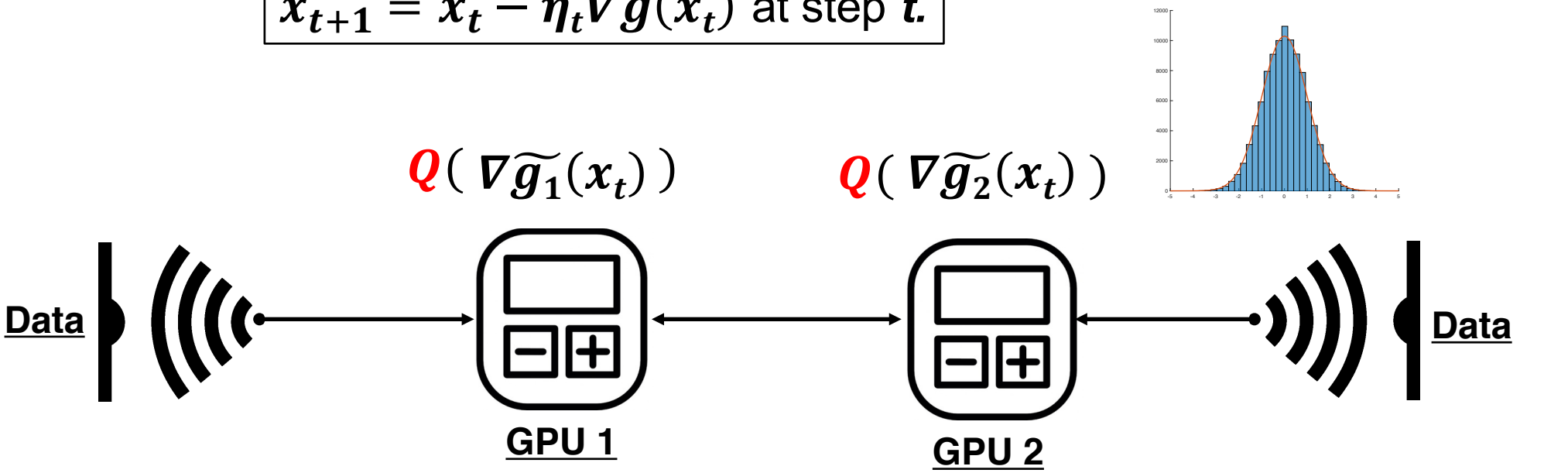
$$T = \mathcal{O}\left(R^2 \frac{\sigma^2}{\varepsilon^2}\right) \text{ iterations.}$$

Higher variance = **more iterations** to convergence.

Data Flow: Data-Parallel Training (e.g. GPUs)

Standard SGD step:

$$x_{t+1} = x_t - \eta_t \nabla \widetilde{g}(x_t) \text{ at step } t.$$



Quantized SGD step:

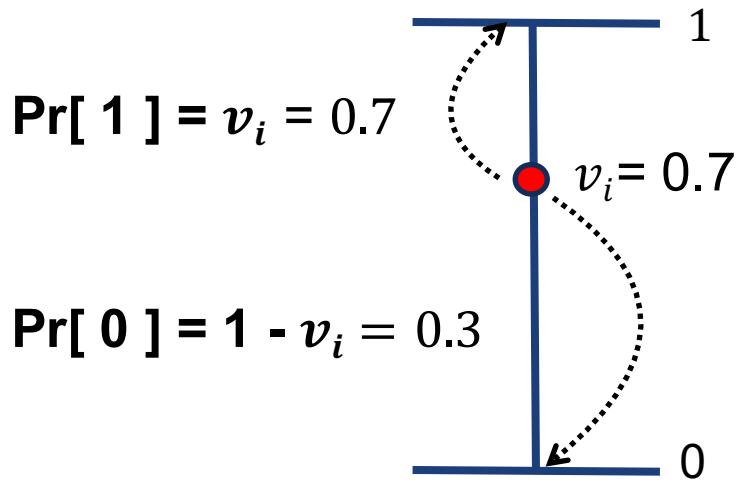
$$x_{t+1} = x_t - \eta_t Q(\nabla \widetilde{g}(x_t)) \text{ at step } t.$$

How Do We Quantize?

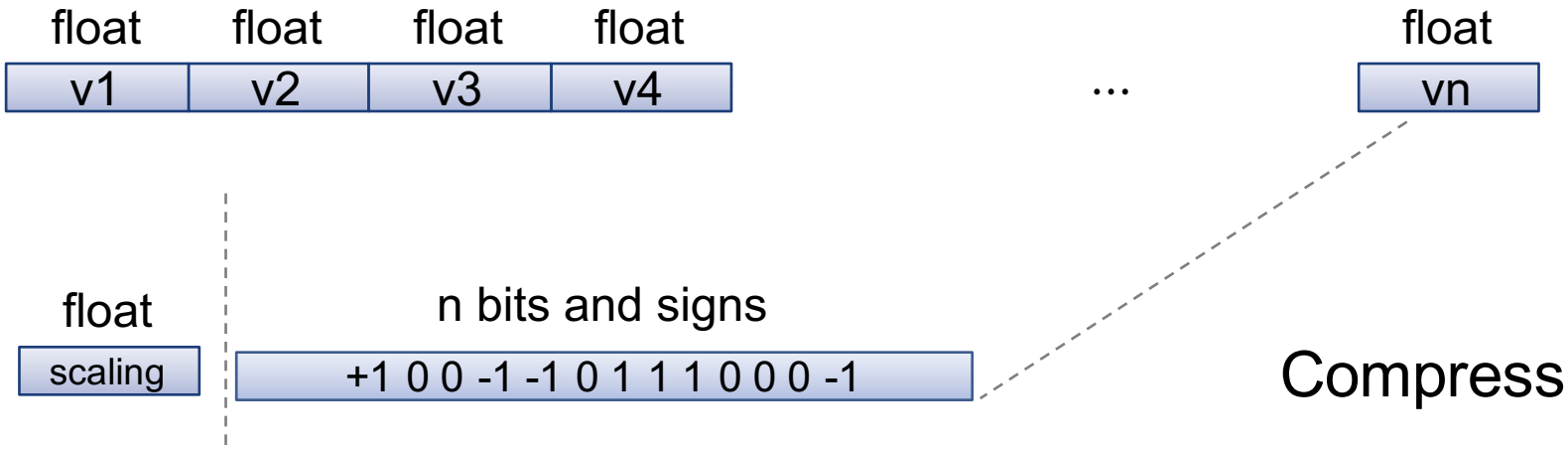
- Gradient = **vector v of dimension n, normalized**
- Quantization function

$$Q[v_i] = \xi_i(v_i) \cdot \mathrm{sgn}(v_i)$$

where $\xi_i(v_i) = \mathbf{1}$ with probability $|\boldsymbol{v_i}|$, and **0**, otherwise.



- Quantization is an **unbiased estimator: $E[Q[v]] = v$.**
- Why do this?

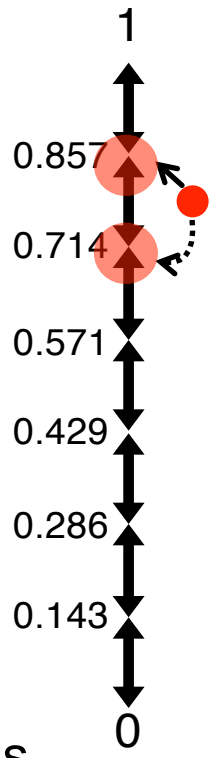


Compression rate > 15x

Gradient Compression

- We apply stochastic rounding to ***gradients***
- The SGD iteration becomes:

$$x_{t+1} = x_t - \eta_t Q(\tilde{g}(x_t)) \text{ at step } t.$$



Theorem [QSGD: Alistarh, Grubic, Li, Tomioka, Vojnovic, 2016]
Given dimension ***n***, QSGD guarantees the following:

1. **Convergence**:
   If **SGD** converges, then **QSGD converges**.
2. **Convergence speed:**
   If **SGD converges** in ***T* iterations, QSGD converges in** $\leq \sqrt{n}\ T$ iterations.
3. **Bandwidth cost**:

The Gamble: The **benefit** of **reduced communication** will outweigh the **performance hit** because of **extra iterations/variance** and **coding/decoding**.

Does it *actually* work?

Experimental Setup

Where?

- Amazon p16xLarge (16 x NVIDIA K80 GPUs)
- Microsoft **CNTK v2.0**, with **MPI-based communication (no NVIDIA NCCL)**

What?

- **Tasks: image classification (ImageNet)** and **speech recognition (CMU AN4)**
- **Nets: ResNet, VGG, Inception, AlexNet,** respectively **LSTM**
- With default parameters

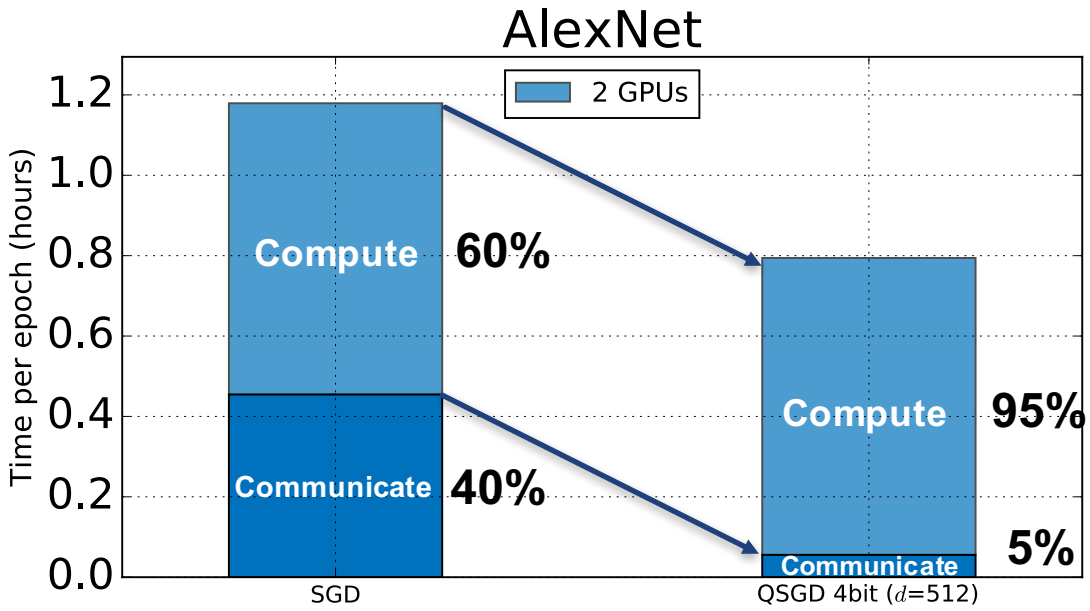
Why?

- Accuracy vs. Speed/Scalability

Open-source implementation, as well as docker containers.

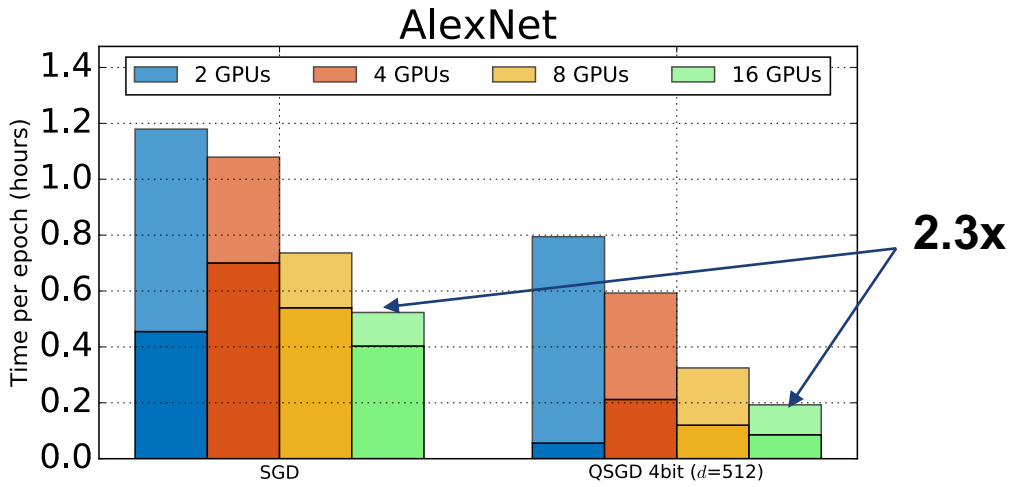
Experiments: Communication Cost

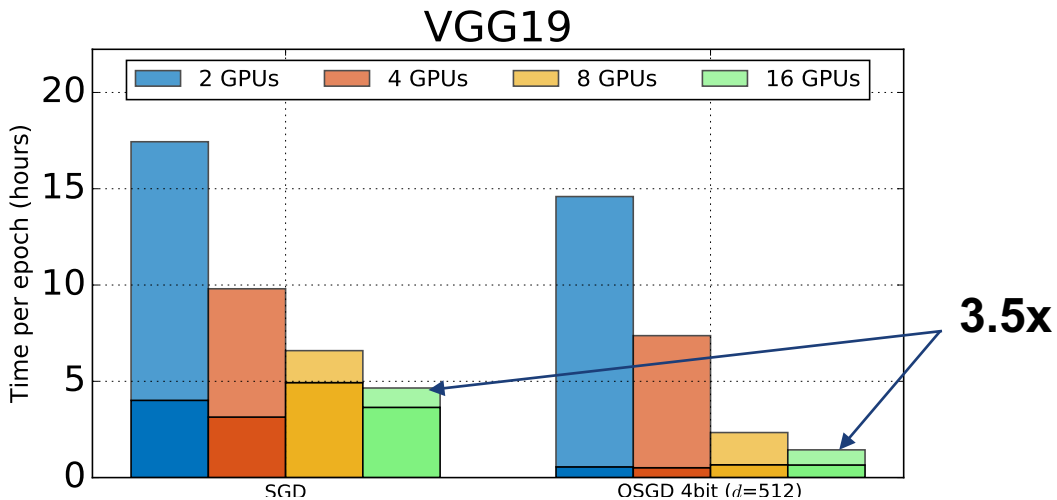
- AlexNet x ImageNet-1K x 2 GPUs

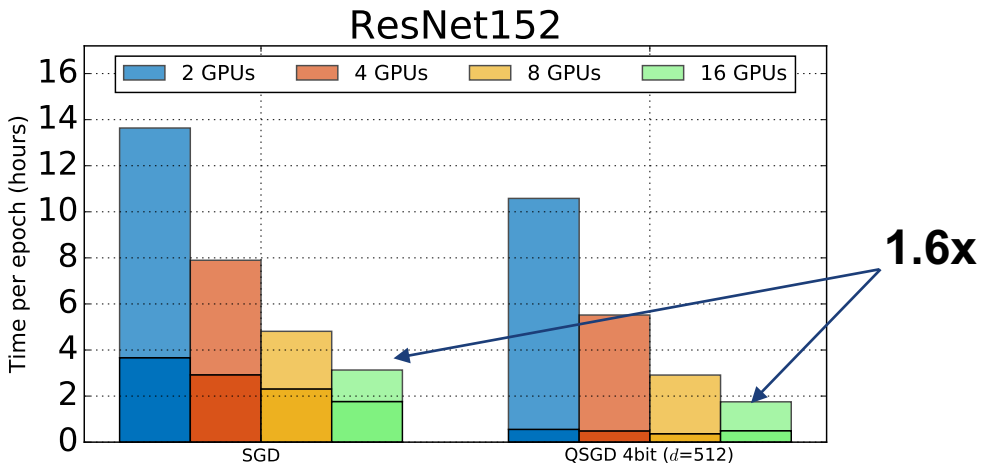


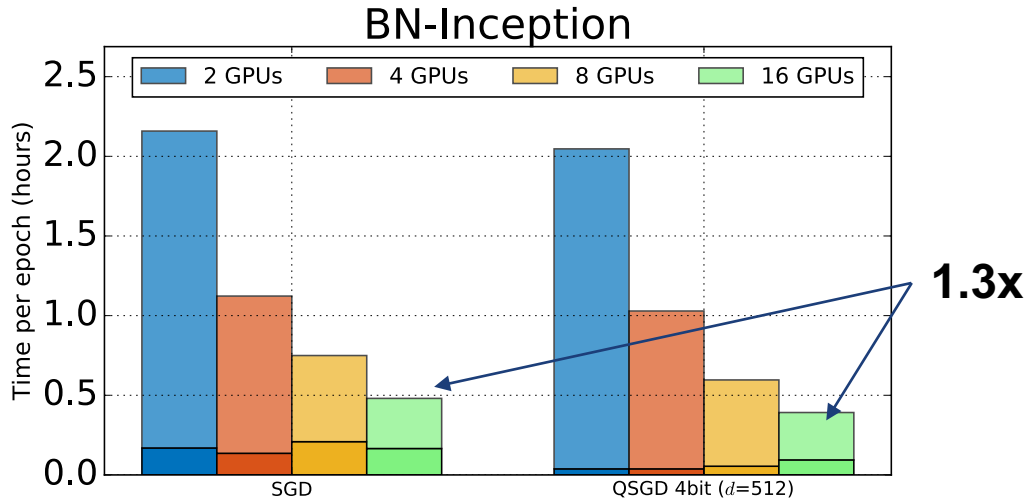
SGD vs QSGD on AlexNet.

Experiments: "Strong" Scaling

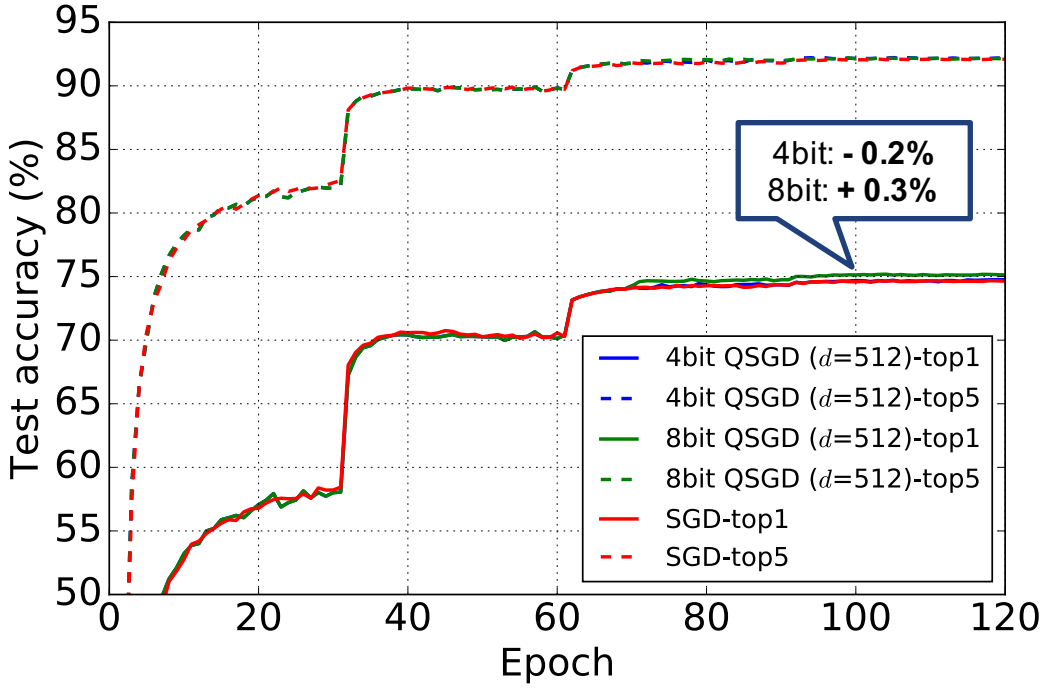




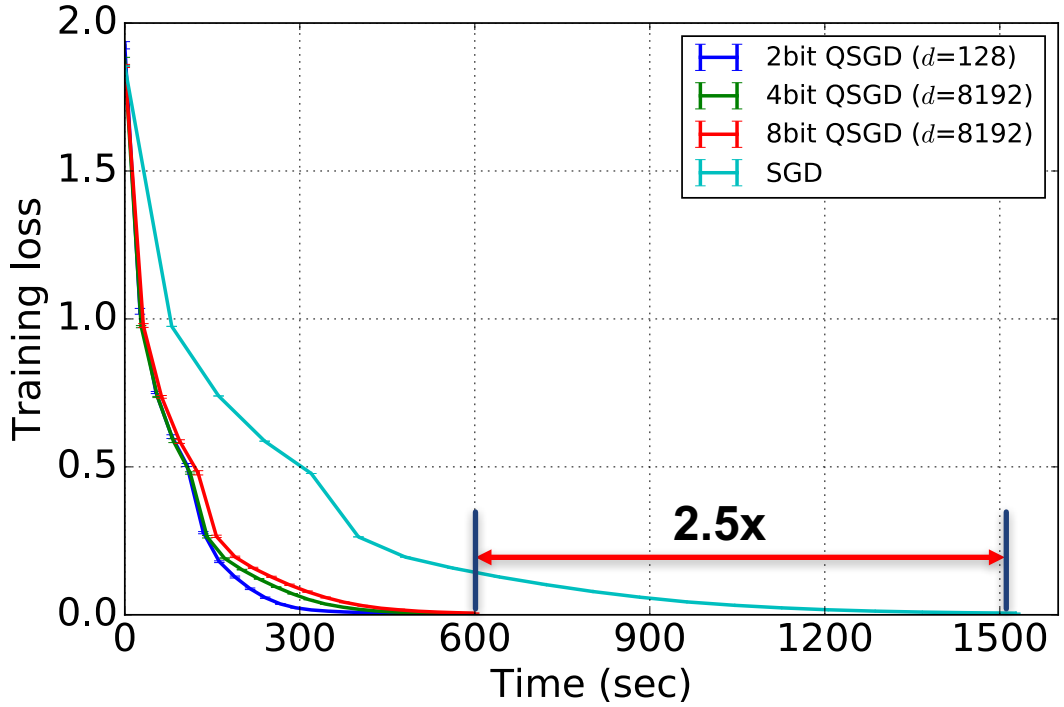




Experiments: A Closer Look at Accuracy



ResNet50 on ImageNet

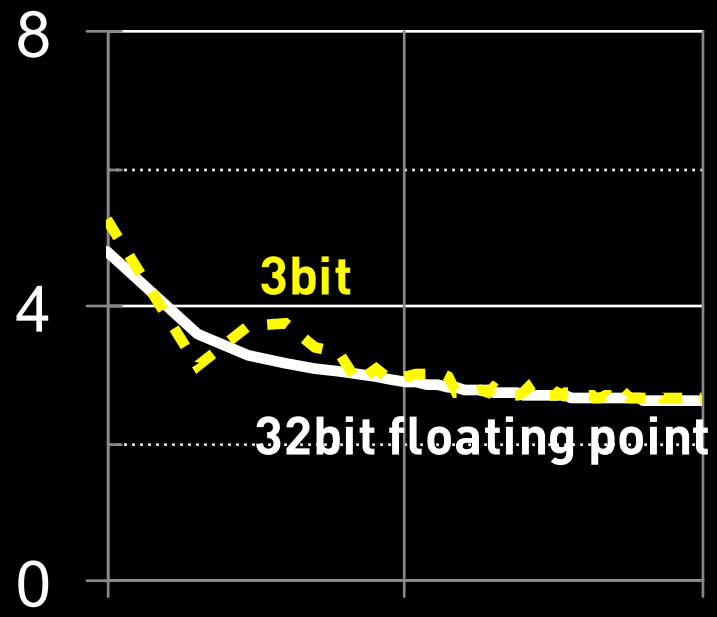


3-Layer LSTM on CMU AN4 (Speech)

Across all networks we tried, 4 bits are sufficient for full accuracy. (QSGD arxiv tech report contains full numbers and comparisons.)

How many bits do you need to represent
a single number in machine learning systems?

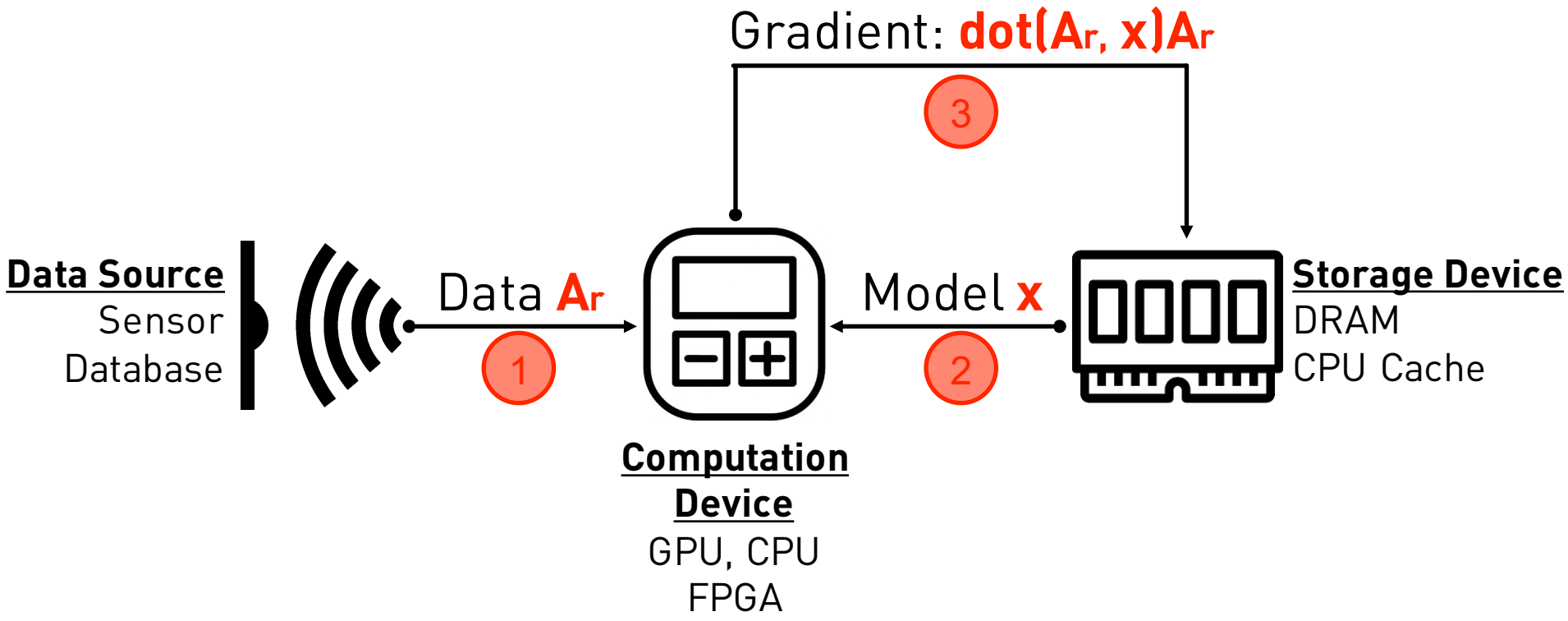
Takeaways



Training Neural Networks
4 bits is enough for **communication**

Training Linear Models
4 bits is enough **end-to-end**

Data Flow in Machine Learning Systems



ZipML

$$\min_x \frac{1}{2} \sum_r (A_r \cdot x - b_r)^2$$

[.... 0.7 ....]

1

0

Naive solution: nearest rounding (=1)
=> Converge to a different solution

Stochastic rounding:
0 with prob 0.3, 1 with prob 0.7

Gradient: **dot(A_r, x)A_r**

3

Expectation matches
=> OK!
(Over-simplified, need to be careful about variance!)

Data Source
Sensor
Database

Data **A_r**

1

Model **x**

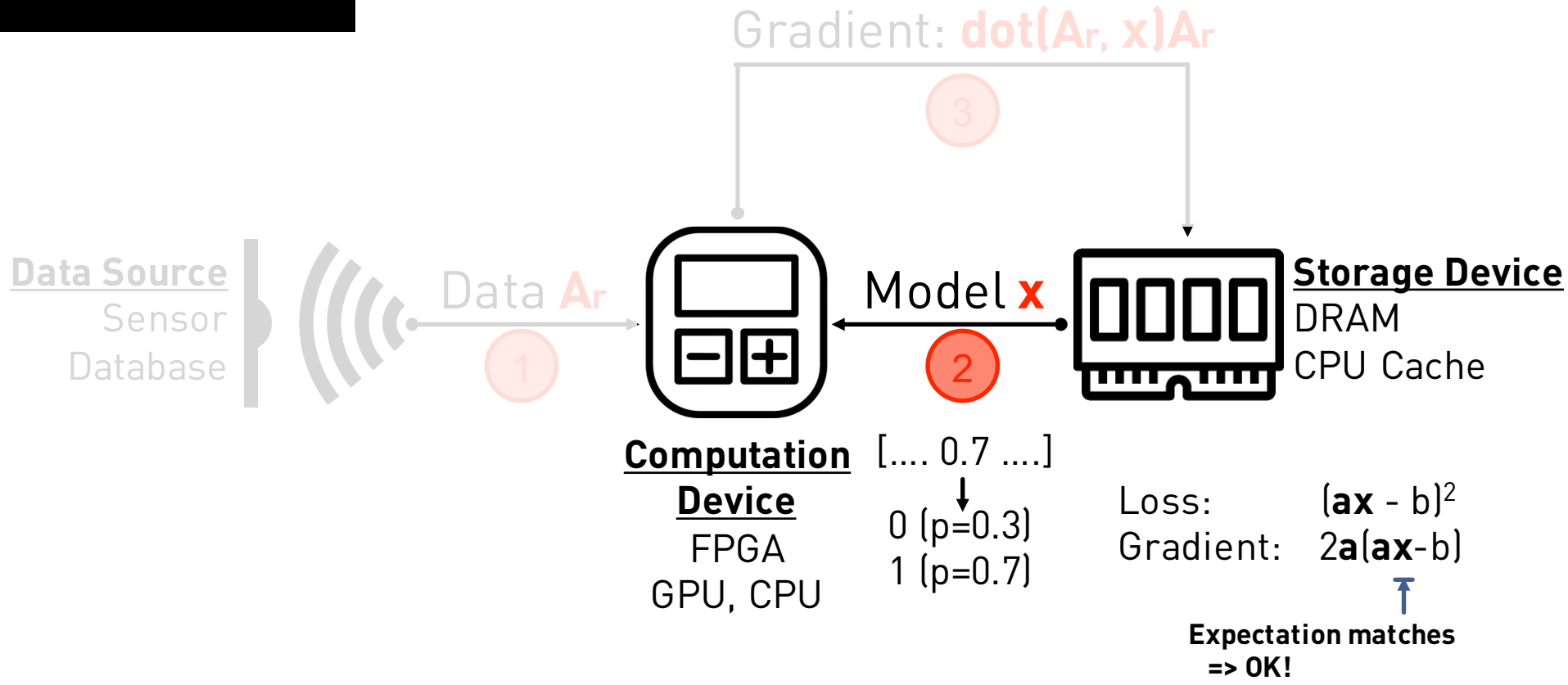
2

Storage Device
DRAM
CPU Cache

Computation Device
FPGA
GPU, CPU

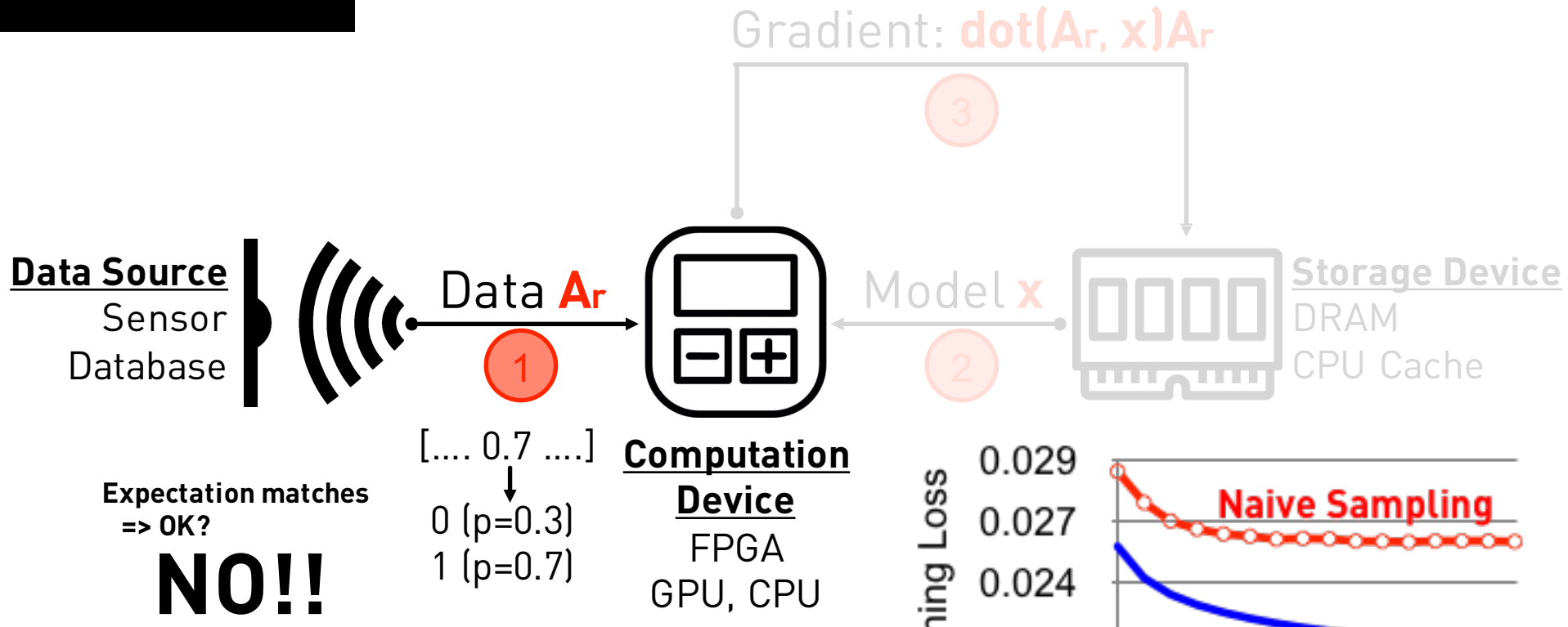
ZipML

$$\min_x \frac{1}{2} \sum_r (A_r \cdot x - b_r)^2$$



ZipML

$$\min_x \frac{1}{2}\sum_r (A_r \cdot x - b_r)^2$$



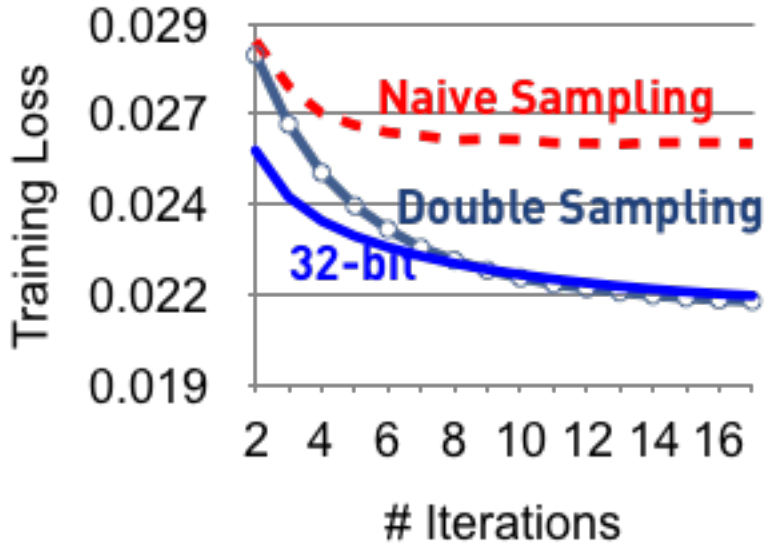
ZipML: "Double Sampling"

How to generate samples for **a** to get an unbiased estimator for $2\mathbf{a}(\mathbf{a}x-b)$?

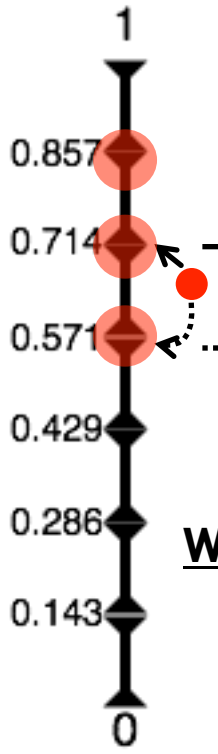
TWO Independent Samples!

$$2\mathbf{a}_1(\mathbf{a}_2x-b)$$

First Sample, Second Sample



How many more bits do we need to store the second sample?



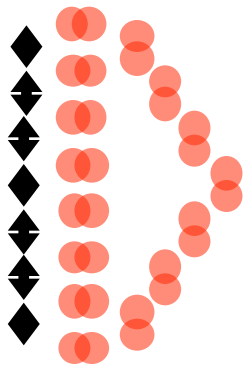
Not 2x Overhead!

3bits to store the first sample

2nd sample only have 3 choices:
- up, down, same

=> 2bits to store

We can do even better—Samples are symmetric!



15 different possibilities

=> 4bits to store 2 samples

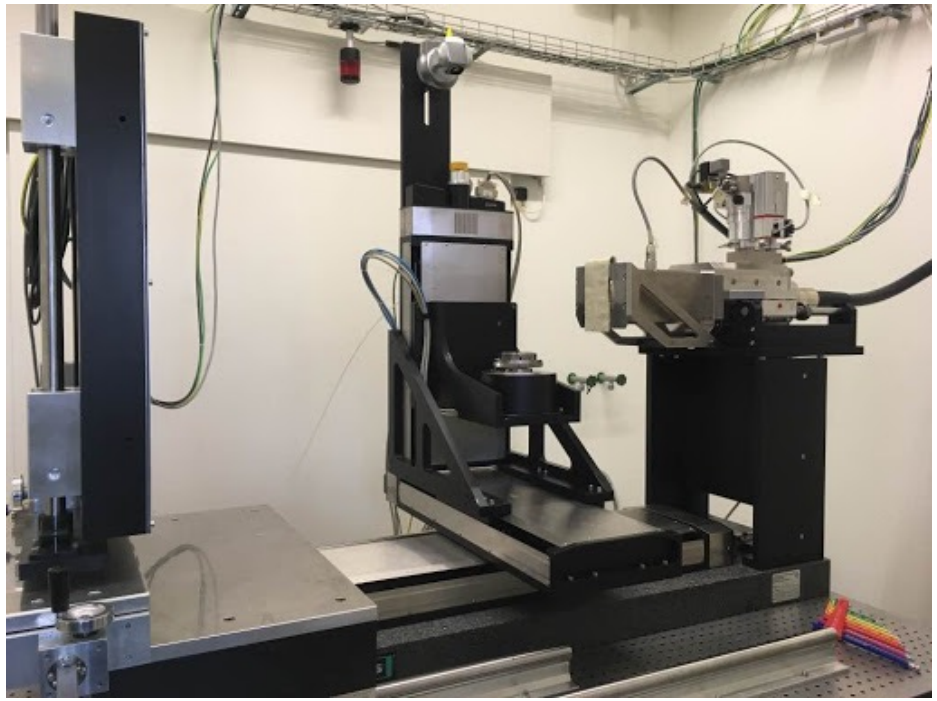
1bit Overhead

arXiv'16

It works!

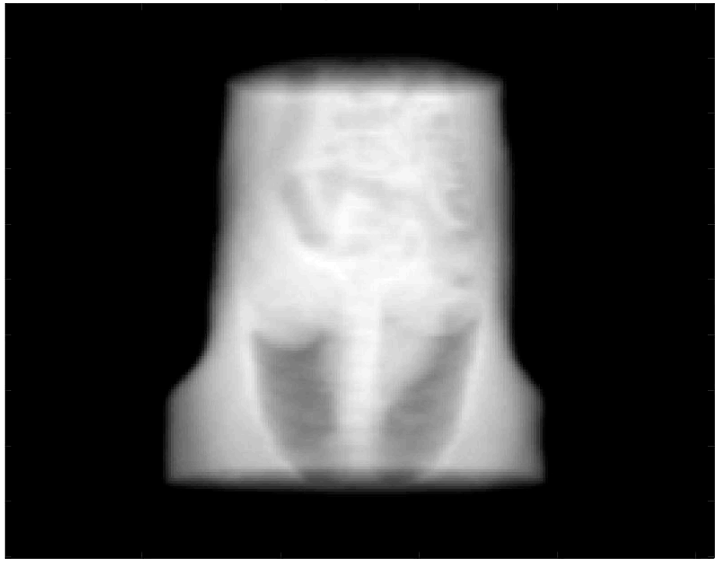
Experiments

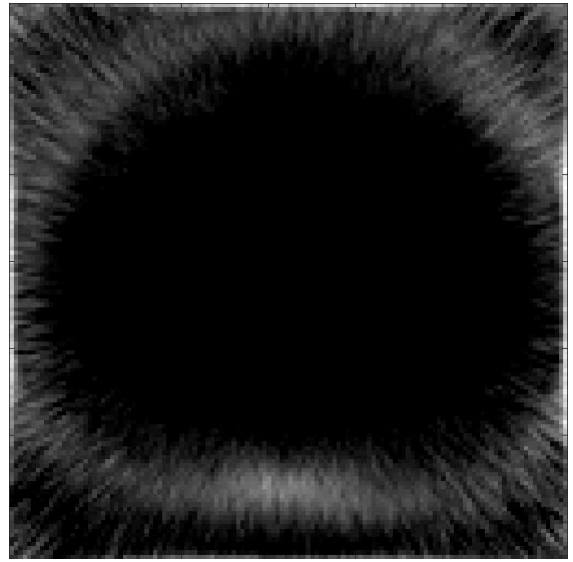
EMPA Tomographic Reconstruction



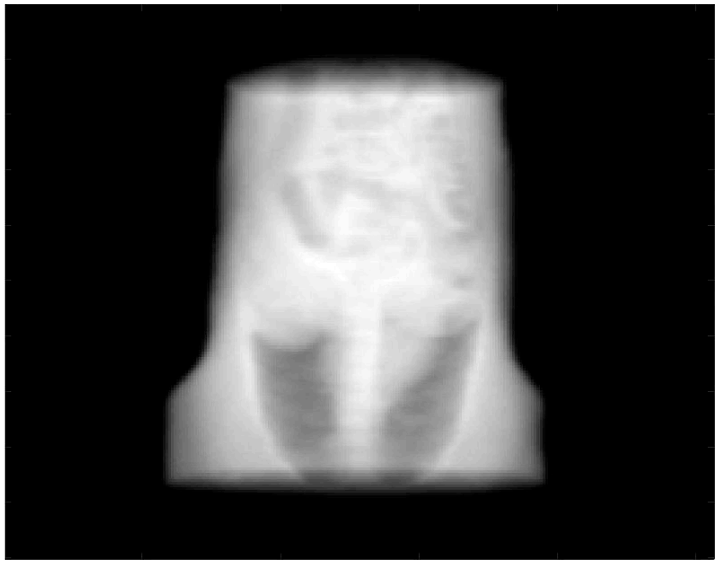
Linear regression with fancy regularization
(but a 240GB model)

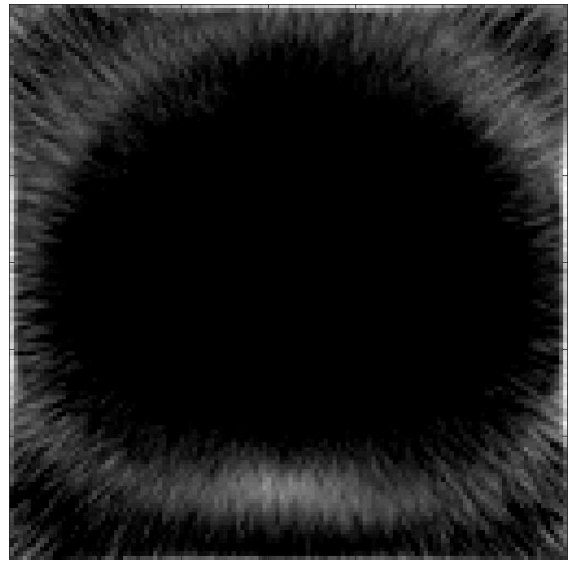
32bit Floating Points





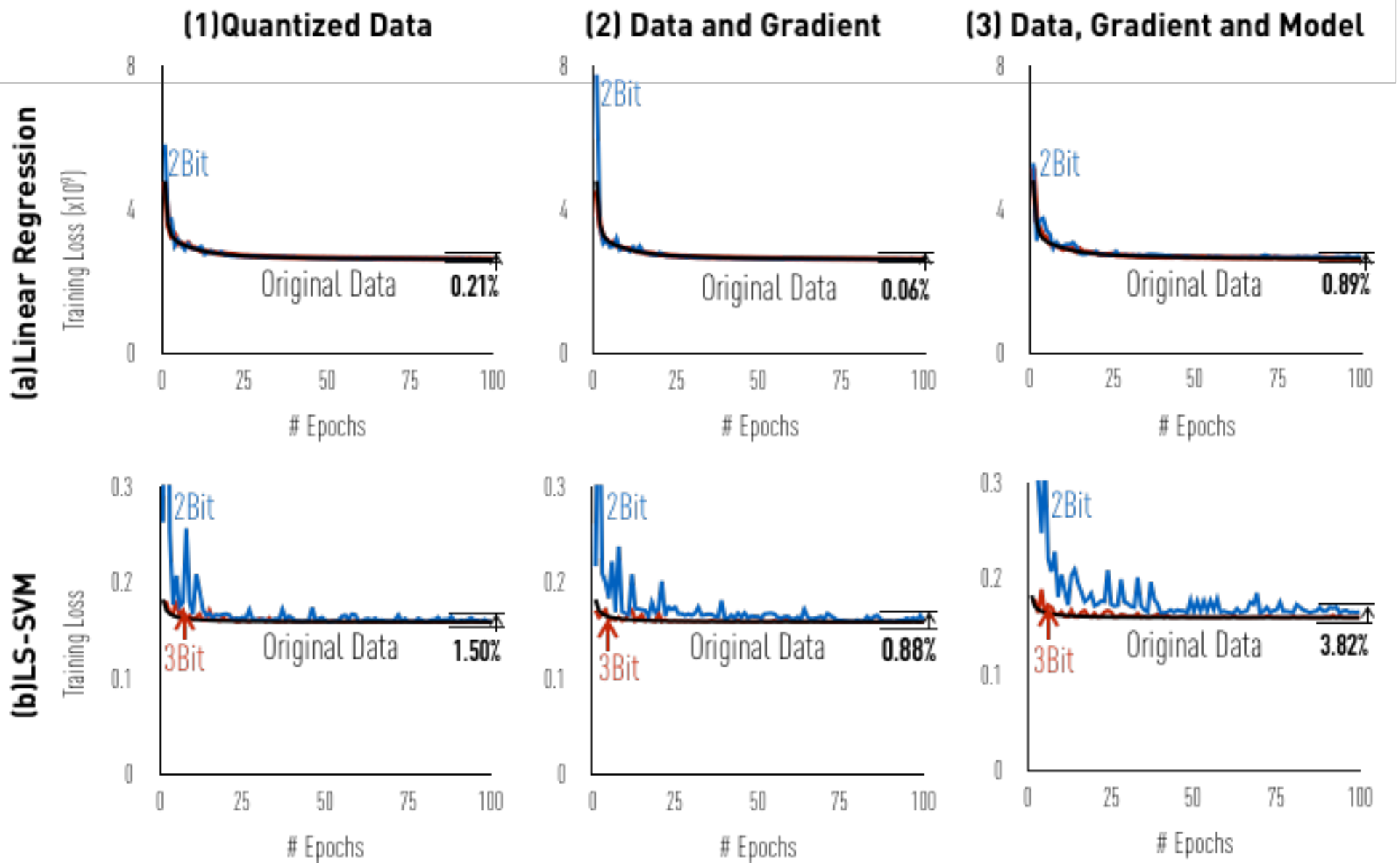
12bit Fixed Points





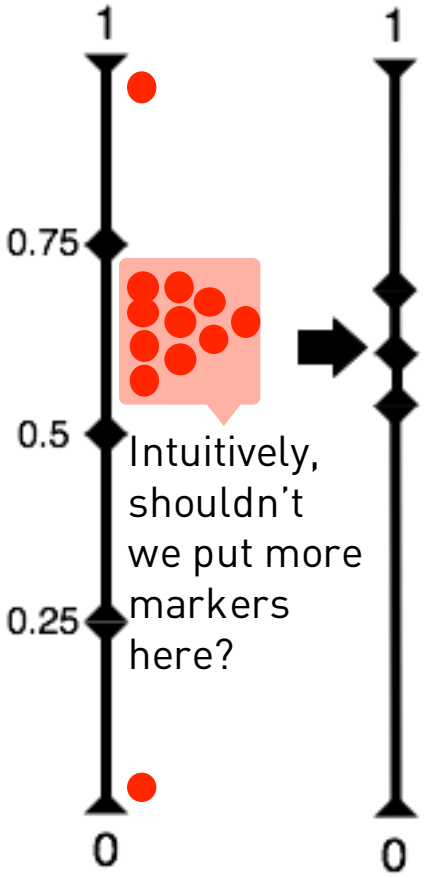
Experiments

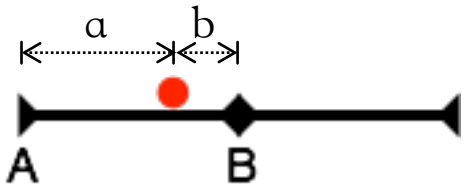
Enterprise Setting



It works, but is what we are doing optimal?

Not Really : Data-optimal Quantization Strategy

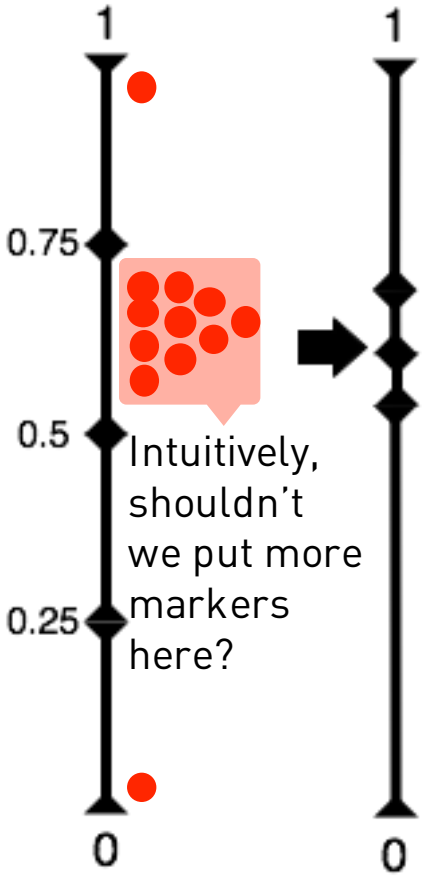


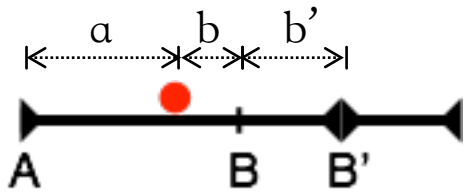


Probability of quantizing to A: $P_A = b / (a+b)$
Probability of quantizing to B: $P_B = a / (a+b)$

Expectation of quantization error for A, B (variance)
$= a P_A + b P_B = 2ab / (a+b)$

Not Really : Data-optimal Quantization Strategy





Probability of quantizing to A: $P_A = b / (a+b)$
Probability of quantizing to B: $P_B = a / (a+b)$

Expectation of quantization error for A, B (variance)
$= a P_A + b P_B = 2ab / (a+b)$

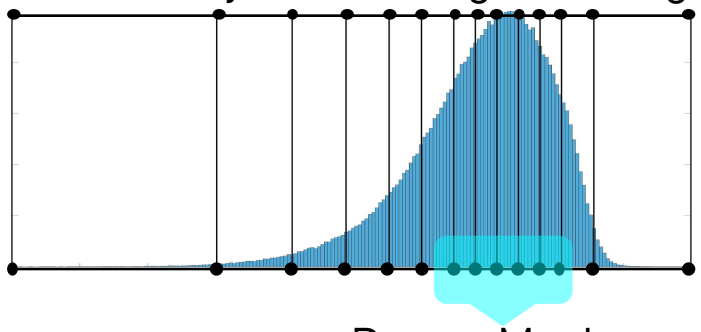
Expectation of quantization error for A, B' (variance)
$= 2a(b+b') / (a+b+b')$

Find a set of markers $c_1 < \cdots < c_s$,

$$\min_{c_1,\ldots,c_s} \sum_j \sum_{a \in [c_j, c_{j+1}]} \frac{(a-c_j)(c_{j+1}-a)}{c_{j+1}-c_j}$$

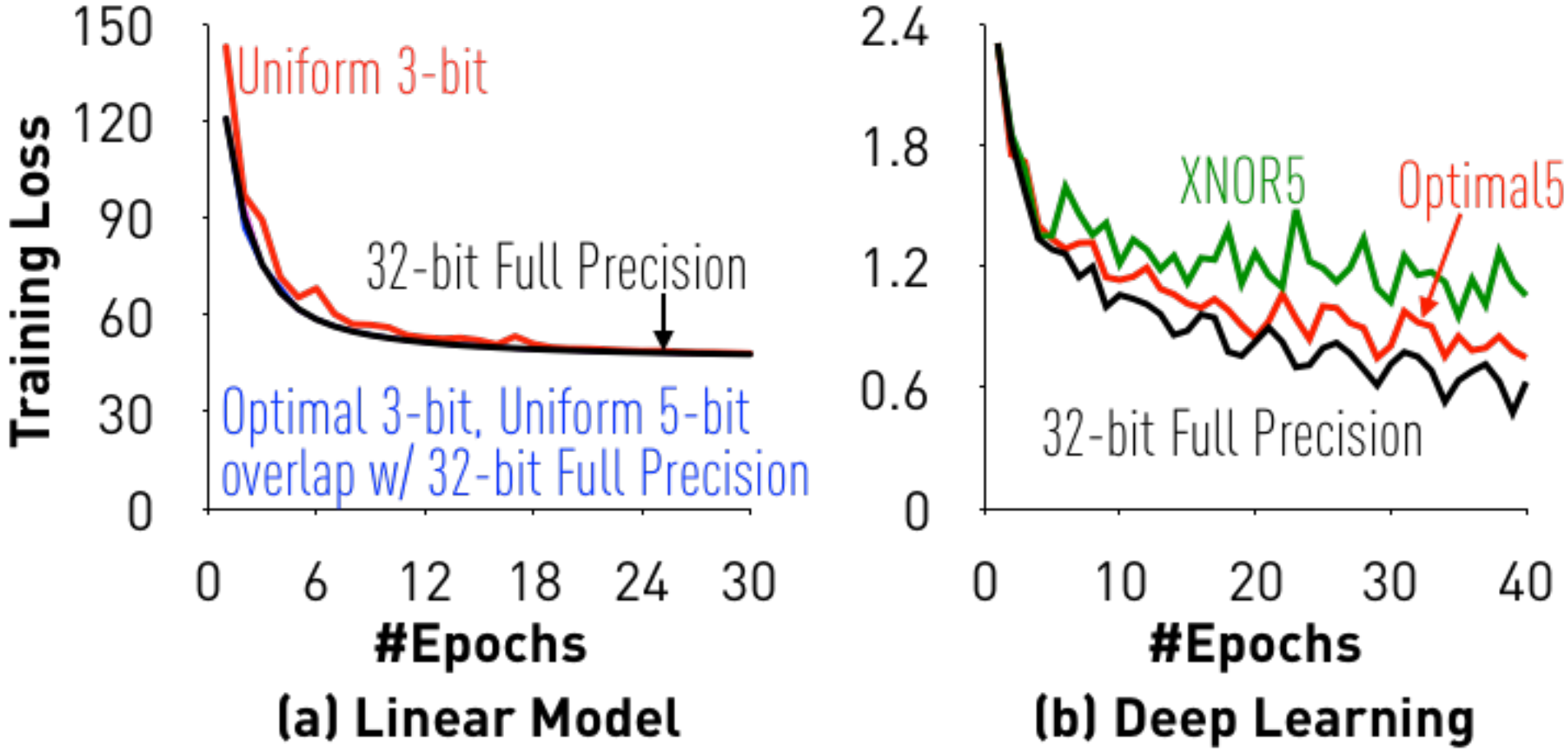
All data points falling into an interval

P-TIME Dynamic Programming



Dense Markers

Experiments



Beyond Linear Regression

General Loss Function

Linear Regression or LS-SVM

$$\min_x \frac{1}{2} \sum_r (A_r \cdot x - b_r)^2$$

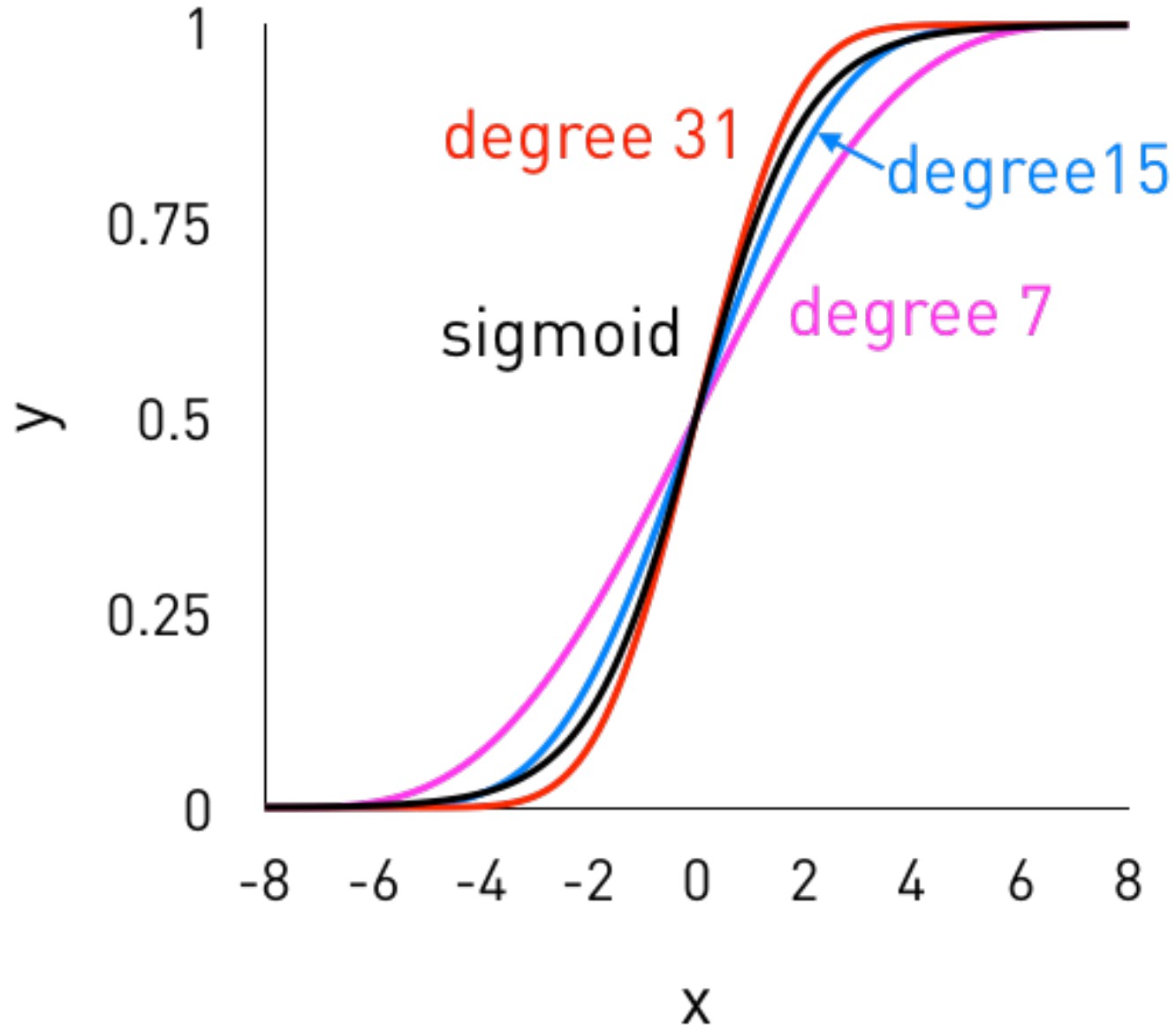
$$\min_x \sum_r \frac{1}{1+\exp\{-A_r \cdot x\}}$$

Logistic Regression

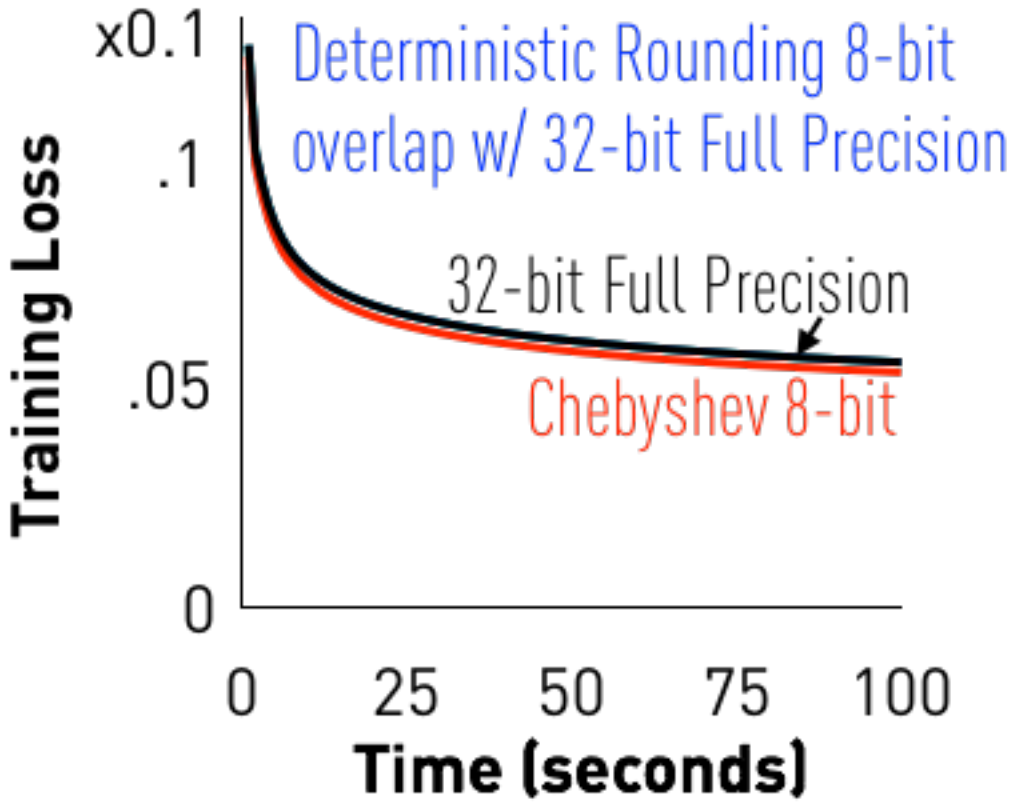
Challenge: Non-Linear terms

Approximation via polynomials

Chebyshev Polynomials



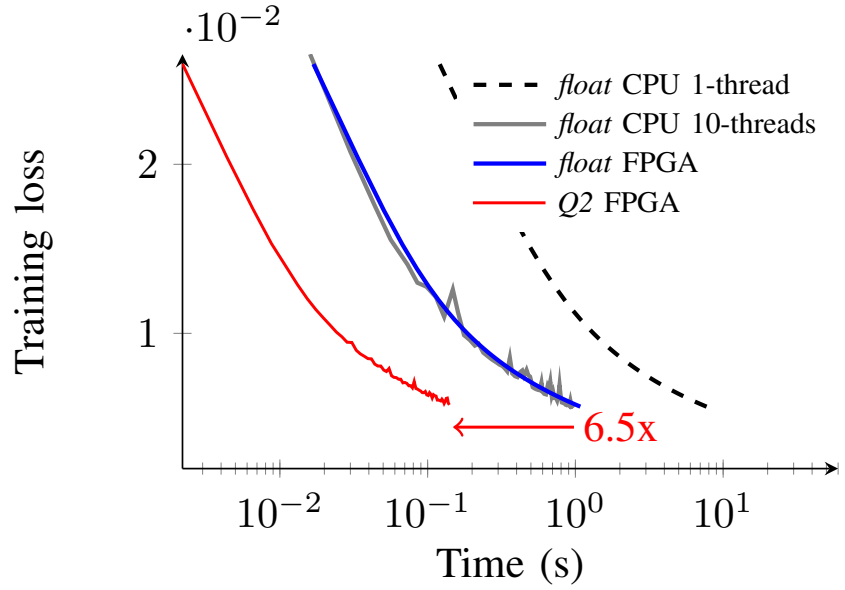
Experiments



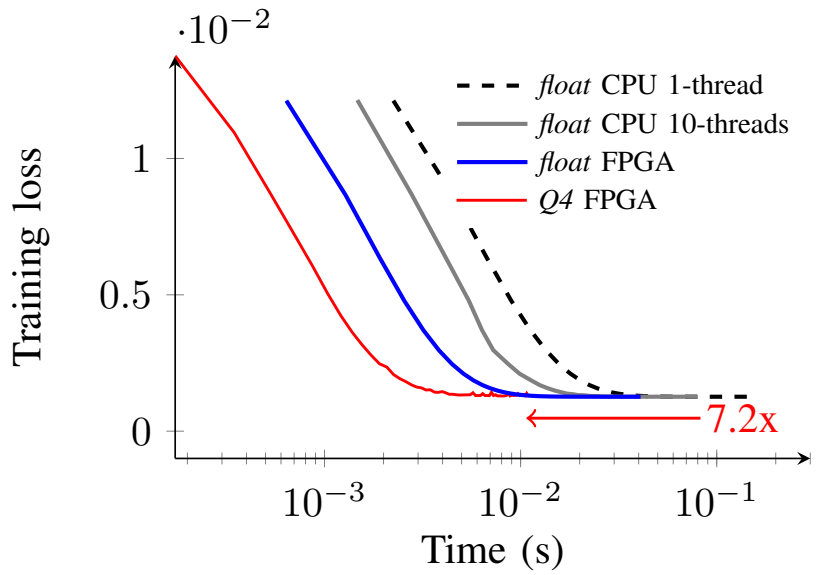
Even for non-linear models, we can do end-to-end quantization at 8 bits, with guaranteed convergence.

Summary & Implementation

- We can do **end-to-end quantized linear regression** with **guarantees**
- We can do arbitrary **classification loss** via **approximation**
- Implemented on an **Xeon + FPGA platform (Intel-Altera HARP)**
  - **Quantized data -> 8-16x more data per transfer**

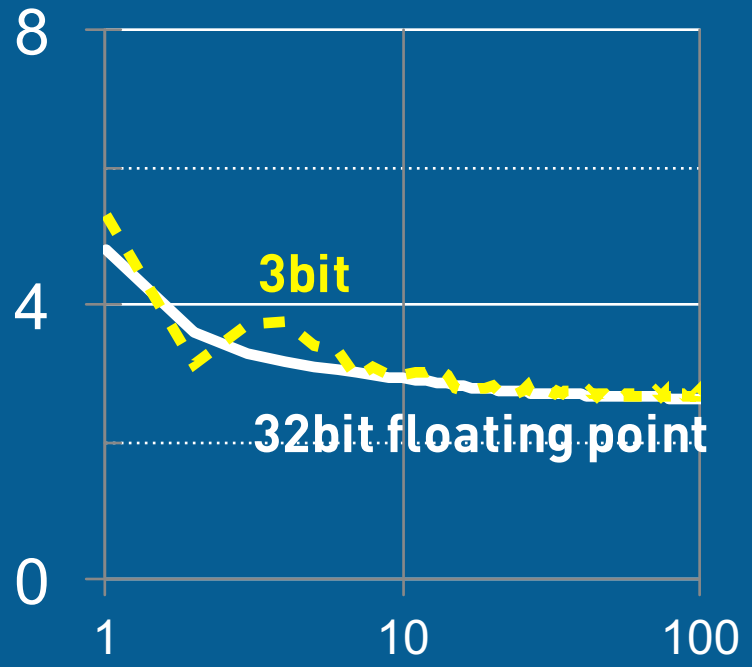


Classification (gisette dataset, n = 5K).



Regression (synthetic dataset, n = 100).

Takeaways



Training Neural Networks
4 bits is enough for **communication**

Training Linear Models
4 bits is enough **end-to-end**

Beyond Empirical
Rigorous theoretical guarantees

Questions?