ZipML: Training Linear Models with End-to-End Low Precision, and a Little Bit of Deep Learning

Hantian Zhang [1] **Jerry Li** [2] **Kaan Kara** [1] **Dan Alistarh** [1 3] **Ji Liu** [4] **Ce Zhang** [1]

Abstract

Recently there has been significant interest in training machine-learning models at low precision: by reducing precision, one can reduce computation and communication by one order of magnitude. We examine training at reduced precision, both from a theoretical and practical perspective, and ask: *is it possible to* train *models at end-to-end low precision with* provable *guarantees? Can this lead to consistent order-of-magnitude speedups?* We mainly focus on linear models, and the answer is yes for linear models. We develop a simple framework called ZipML based on one simple but novel strategy called double sampling. Our ZipML framework is able to execute training at low precision with no bias, guaranteeing convergence, whereas naive quantization would introduce significant bias. We validate our framework across a range of applications, and show that it enables an FPGA prototype that is up to $6.5\times$ faster than an implementation using full 32-bit precision. We further develop a variance-optimal stochastic quantization strategy and show that it can make a significant difference in a variety of settings. When applied to linear models together with double sampling, we save up to another $1.7\times$ in data movement compared with uniform quantization. When training deep networks with quantized models, we achieve higher accuracy than the state-of-the-art XNOR-Net.

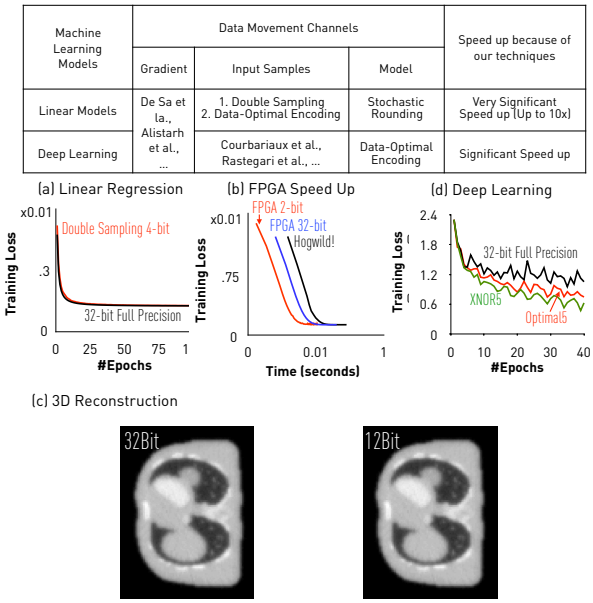


Figure 1. Overview of theoretical results and highlights of empirical results. See Introduction for details.

1. Introduction

The computational cost and power consumption of today's machine learning systems are often driven by data movement, and by the precision of computation. In our experience, in applications such as tomographic reconstruction, anomaly detection in mobile sensor networks, and compressive sensing, the overhead of transmitting the data samples can be massive, and hence performance can hinge on reducing the precision of data representation and associated computation. A similar trend is observed in deep learning, where impressive progress has been reported with systems using end-to-end reduced-precision representations (Hubara et al., 2016; Rastegari et al., 2016; Zhou et al., 2016; Miyashita et al., 2016). The empirical success of these works inspired this paper, in which we try to provide a *theoretical* understanding of end-to-end low-precision training for machine learning models. In this context, the motivating question behind our work is: *When*

[1]ETH Zurich, Switzerland [2]Massachusetts Institute of Technology, USA [3]IST Austria, Austria [4]University of Rochester, USA. Correspondence to: Hantian Zhang <hantian.zhang@inf.ethz.ch>, Ce Zhang <ce.zhang@inf.ethz.ch>.

Proceedings of the 34^{th} International Conference on Machine Learning, Sydney, Australia, PMLR 70, 2017.

training general machine learning models, can we lower the precision of data representation, communication, and computation, while maintaining provable guarantees?

In this paper, we develop ZipML, a general framework to answer this question, and present results obtained in the context of this ZipML framework. Figure 1 encapsulates our results: (a) for linear models, we are able to lower the precision of both computation and communication, including input samples, gradients, and model, by up to 16 times, while still providing rigorous theoretical guarantees; (b) our FPGA implementation of this framework achieves up to $6.5\times$ speedup compared with a 32-bit FPGA implementation, or with a 10-core CPU running Hogwild!; (c) we are able to decrease data movement by $2.7\times$ for tomographic reconstruction, while obtaining a negligible quality decrease. Elements of our framework generalize to (d) model compression for training deep learning models. In the following, we describe our technical contributions in more detail.

1.1. Summary of Technical Contributions

We consider the following problem in training generalized linear models:

$$\min_{\mathbf{x}}: \quad \frac{1}{2K}\sum_{k=1}^{K} l(\mathbf{a}_k^\top \mathbf{x}, b_k)^2 + R(\mathbf{x}), \tag{1}$$

where $l(\cdot,\cdot)$ is a loss function and R is a regularization term that could be ℓ_1 norm, ℓ_2 norm, or even an indicator function representing the constraint. The gradient at the sample $(\mathbf{a}_k, b_k)$ is:

$$\mathbf{g}_k := \mathbf{a}_k \frac{\partial l(\mathbf{a}_k^\top \mathbf{x}, b_k)}{\partial \mathbf{a}_k^\top \mathbf{x}}.$$

We denote the problem dimension by n. We consider the properties of the algorithm when a lossy compression scheme is applied to the data (samples), gradient, and model, to reduce the communication cost of the algorithm—that is, we consider quantization functions Q_g, Q_m, and Q_s for gradient, model, and samples, respectively, in the gradient update:

$$\mathbf{x}_{t+1} \leftarrow \text{prox}_{\gamma R(\cdot)}\left(\mathbf{x}_t - \gamma Q_g(\mathbf{g}_k(Q_m(\mathbf{x}_t), Q_s(\mathbf{a}_t)))\right), \tag{2}$$

where the proximal operator is defined as

$$\text{prox}_{\gamma R(\cdot)}(\mathbf{y}) = \underset{\mathbf{x}}{\text{argmin}} \frac{1}{2}\|\mathbf{x}-\mathbf{y}\|^2 + \gamma R(\mathbf{x}).$$

Our Results. We summarize our results as follows.

Linear Models. When $l(\cdot,\cdot)$ is the least squares loss, we first notice that simply doing stochastic quantization of data samples (i.e., Q_s) introduces bias of the gradient estimator and therefore SGD would converge to a different solution. We propose a simple solution to this problem by introducing a *double sampling* strategy $\tilde{Q}_s$ that uses multiple samples to eliminate the correlation of samples introduced by the non-linearity of the gradient. We analyze the additional variance introduced by double sampling, and find that its impact is *negligible in terms of convergence time* as long as the number of bits used to store a quantized sample is at least $\Theta(\log n/\sigma)$, where σ^2 is the variance of the standard stochastic gradient. This implies that the 32-bit precision may be excessive for many practical scenarios.

We build on this result to obtain an *end-to-end quantization* strategy for linear models, which compresses all data movements. For certain settings of parameters, end-to-end quantization adds as little as a *constant factor* to the variance of the entire process.

Optimal Quantization and Extension to Deep Learning. We then focus on reducing the variance of stochastic quantization. We notice that different methods for setting the quantization points have different variances—the standard uniformly-distributed quantization strategy is far from optimal in many settings. We formulate this as an independent optimization problem, and solve it optimally with an efficient dynamic programming algorithm that only needs to scan the data in a single pass. When applied to linear models, this optimal strategy can save up to $1.6\times$ communication compared with the uniform strategy.

We perform an analysis of the optimal quantizations for various settings, and observe that the uniform quantization approach popularly used by state-of-the-art end-to-end low-precision deep learning training systems when more than 1 bit is used is suboptimal. We apply optimal quantization to models and show that, with one standard neural network, we outperform the uniform quantization used by XNOR-Net and a range of other recent approaches. This is related, but different, to recent work on model compression for inference (Han et al., 2016). To the best of our knowledge, this is the first time such optimal quantization strategies have been applied to training.

2. Linear Models

In this section, we focus on linear models with possibly non-smooth regularization. We have labeled data points $(\mathbf{a}_1, b_1), (\mathbf{a}_2, b_2), \ldots, (\mathbf{a}_K, b_K) \in \mathbb{R}^n \times \mathbb{R}$, and our goal is to minimize the function

$$F(\mathbf{x}) = \underbrace{\frac{1}{K}\sum_{k=1}^{K} \|\mathbf{a}_k^\top \mathbf{x} - b_k\|_2^2}_{=:f(\mathbf{x})} + R(\mathbf{x}), \tag{3}$$

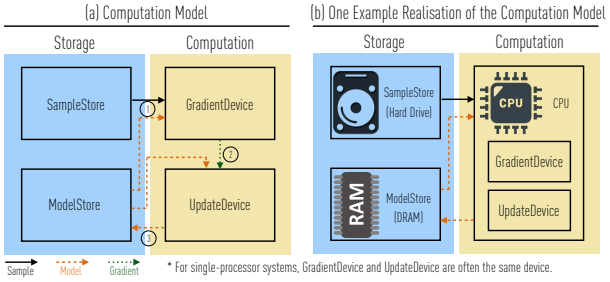


Figure 2. A schematic representation of the computational model.

i.e., minimize the empirical least squares loss plus a non-smooth regularization $R(\cdot)$ (e.g., ℓ_1 norm, ℓ_2 norm, and constraint indicator function). SGD is a popular approach for solving large-scale machine learning problems. It works as follows: at step $\mathbf{x}_t$, given an unbiased gradient estimator $\mathbf{g}_t$, that is, $\mathbb{E}(\mathbf{g}_t) = \nabla f(\mathbf{x}_t)$, we update $\mathbf{x}_{t+1}$ by

$$\mathbf{x}_{t+1} = \text{prox}_{\gamma_t R(\cdot)}\left(\mathbf{x}_t - \gamma_t \mathbf{g}_t\right),$$

where γ_t is the predefined step length. SGD guarantees the following convergence property:

Theorem 1. *[e.g., (Bubeck, 2015), Theorem 6.3] Let the sequence $\{\mathbf{x}_t\}_{t=1}^T$ be bounded. Appropriately choosing the steplength, we have the following convergence rate for* (3)*:*

$$F\left(\frac{1}{T}\sum_{t=0}^{T}\mathbf{x}_t\right) - \min_{\mathbf{x}} F(\mathbf{x}) \leq \Theta\left(\frac{1}{T} + \frac{\sigma}{\sqrt{T}}\right) \qquad (4)$$

where σ is the upper bound of the mean variance

$$\sigma^2 \geq \frac{1}{T}\sum_{t=1}^{T}\mathbb{E}\|\mathbf{g}_t - \nabla f(\mathbf{x}_t)\|^2.$$

There are three key requirements for SGD to converge:

1. Computing stochastic gradient $\mathbf{g}_t$ is cheap;
2. The stochastic gradient $\mathbf{g}_t$ should be unbiased;
3. The stochastic gradient variance σ dominates the convergence efficiency, so it needs to be controlled appropriately.

The common choice is to uniformly select one sample:

$$\mathbf{g}_t = \mathbf{g}_t^{(full)} := \mathbf{a}_{\pi(t)}(\mathbf{a}_{\pi(t)}^\top \mathbf{x} - b_{\pi(t)}). \qquad (5)$$

($\pi(t)$ is a uniformly random integer from 1 to K). We abuse the notation and let $\mathbf{a}_t = \mathbf{a}_{\pi(t)}$. Note that $\mathbf{g}_t^{(full)}$ is an unbiased estimator $\mathbb{E}[\mathbf{g}_t^{(full)}] = \nabla f(\mathbf{x}_t)$. Although it has received success in many applications, if the precision of sample $\mathbf{a}_t$ can be further decreased, we can save potentially one order of magnitude bandwidth of reading $\mathbf{a}_t$ (e.g., in sensor networks) and the associated computation (e.g., each register can hold more numbers). This motivates us to use low-precision sample points to train the model. The following will introduce the proposed low-precision SGD framework by meeting all three factors for SGD.

2.1. Bandwidth-Efficient Stochastic Quantization

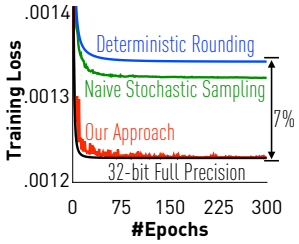
We propose to use stochastic quantization to generate a low-precision version of an arbitrary vector $\mathbf{v}$ in the following way. Given a vector $\mathbf{v}$, let $M(\mathbf{v})$ be a scaling factor such that $-1 \leq \mathbf{v}/M(\mathbf{v}) \leq 1$. Without loss of generality, let $M(\mathbf{v}) = ||\mathbf{v}||_2$. We partition the interval $[-1, 1]$ using $s+1$ separators: $-1 = l_0 \leq l_1 ... \leq l_s = 1$; for each number v in $\mathbf{v}/M(\mathbf{v})$, we quantize it to one of two nearest separators: $l_i \leq v \leq l_{i+1}$. We denote the *stochastic quantization* function by $Q(\mathbf{v}, s)$ and choose the probability of quantizing to different separators such that $\mathbb{E}[Q(\mathbf{v}, s)] = \mathbf{v}$. We use $Q(\mathbf{v})$ when s is not relevant.

2.2. Double Sampling for Unbiased Stochastic Gradient

The naive way to use low-precision samples $\hat{\mathbf{a}}_t := Q(\mathbf{a}_t)$ is

$$\hat{\mathbf{g}}_t := \hat{\mathbf{a}}_t\hat{\mathbf{a}}_t^\top \mathbf{x} - \hat{\mathbf{a}}_t b_t.$$

However, *the naive approach does not work* (that is, it does not guarantee convergence), because it is biased:



$$\mathbb{E}[\hat{\mathbf{g}}_t] := \mathbf{a}_t\mathbf{a}_t^\top \mathbf{x} - \mathbf{a}_t b_t + D_{\mathbf{a}}\mathbf{x},$$

where $D_{\mathbf{a}}$ is diagonal and its ith diagonal element is

$$\mathbb{E}[Q(\mathbf{a}_i)^2] - \mathbf{a}_i^2.$$

Since $D_{\mathbf{a}}$ is non-zero, we obtain a *biased* estimator of the gradient, so the iteration is unlikely to converge. The figure on the right illustrates the bias caused by a non-zero $D_{\mathbf{a}}$. In fact, it is easy to see that in instances where the minimizer $\mathbf{x}$ is large and gradients become small, we will simply diverge.

We now present a simple method to fix the biased gradient estimator. We generate two independent random quantizations and revise the gradient:

$$\mathbf{g}_t := Q_1(\mathbf{a}_t)(Q_2(\mathbf{a}_t)^\top \mathbf{x} - b_t)\ . \qquad (6)$$

This gives us an unbiased estimator of the gradient.

Overhead of Storing Samples. The reader may have noticed that one implication of double sampling is the overhead of sending two samples instead of one. We note that

this will not introduce $2\times$ overhead in terms of data communication. Instead, we start from the observation that the two samples can differ by at most one bit. For example, to quantize the number 0.7 to either 0 or 1. Our strategy is to first store the smallest number of the interval (here 0), and then for each sample, send out 1 bit to represent whether this sample is at the lower marker (0) or the upper marker (1). Under this procedure, once we store the base quantization level, we will need one extra bit for each additional sample. More generally, since samples are used symmetrically, we only need to send a number representing the number of times the lower quantization level has been chosen among all the sampling trials. Thus, sending k samples only requires $\log_2 k$ more bits.

2.3. Variance Reduction

From Theorem 1, the mean variance $\frac{1}{T}\sum_t \mathbb{E}\|\mathbf{g}_t - \nabla f(\mathbf{x})\|^2$ will dominate the convergence efficiency. It is not hard to see that the variance of the double sampling based stochastic gradient in (6) can be decomposed into

$$\begin{aligned}\mathbb{E}\|\mathbf{g}_t - \nabla f(\mathbf{x}_t)\|^2 \leq \mathbb{E}\|\mathbf{g}_t^{(full)} - \nabla f(\mathbf{x}_t)\|^2 \\ + \mathbb{E}\|\mathbf{g}_t - \mathbf{g}_t^{(full)}\|^2.\end{aligned} \tag{7}$$

The first term is from the full stochastic gradient, which can be reduced by using strategies such as mini-batch, weight sampling, and so on. Thus, reducing the first term is an orthogonal issue for this paper. Rather, we are interested in the second term, which is the additional cost of using low-precision samples. All strategies for reducing the variance of the first term can seamlessly combine with the approach of this paper. The additional cost can be bounded by the following lemma.

Lemma 1. *The stochastic gradient variance using double sampling in* (6) $\mathbb{E}\|\mathbf{g}_t - \mathbf{g}_t^{(full)}\|^2$ *can be bounded by*

$$\Theta\left(\mathcal{TV}(\mathbf{a}_t)(\mathcal{TV}(\mathbf{a}_t)\|\mathbf{x}\odot\mathbf{x}\| + \|\mathbf{a}_t^\top\mathbf{x}\|^2 + \|\mathbf{x}\odot\mathbf{x}\|\|\mathbf{a}_t\|^2)\right),$$

where $\mathcal{TV}(\mathbf{a}_t) := \mathbb{E}\|Q(\mathbf{a}_t) - \mathbf{a}_t\|^2$ *and* $\odot$ *denotes the element product.*

Thus, minimizing $\mathcal{TV}(\mathbf{a}_t)$ is key to reducing variance.

Uniform quantization. It makes intuitive sense that, the more levels of quantization, the lower the variance. The following makes this quantitative dependence precise.

Lemma 2. *[(Alistarh et al., 2016)] Assume that quantization levels are uniformly distributed. For any vector* $\mathbf{v}\in\mathbb{R}^n$*, we have that* $\mathbb{E}[Q(\mathbf{v},s)] = \mathbf{v}$*. Further, the variance of uniform quantization with* s *levels is bounded by*

$$\mathcal{TV}_s(\mathbf{v}) := \mathbb{E}[\|Q(\mathbf{v},s)-\mathbf{v}\|_2^2] \leq \min(n/s^2, \sqrt{n}/s))\|\mathbf{v}\|_2^2. .$$

Together with other results, it suggests the stochastic gradient variance of using double sampling is bounded by

$$\mathbb{E}\|\mathbf{g}_t - \nabla f(\mathbf{x}_t)\|^2 \leq \sigma_{(full)}^2 + \Theta\left(n/s^2\right),$$

where $\sigma_{(full)}^2 \geq \mathbb{E}\|\mathbf{g}_t^{(full)} - \nabla f(\mathbf{x})\|^2$ is the upper bound of using the full stochastic gradient, assuming that $\mathbf{x}$ and all $\mathbf{a}_k$'s are bounded. Because the number of quantization levels s is exponential to the number of bits we use to quantize, to ensure that these two terms are comparable (using a low-precision sample does not degrade the convergence rate), the number of bits only needs to be greater than $\Theta(\log n/\sigma_{(full)})$. Even for linear models with millions of features, 32 bits is likely to be "overkill."

3. Optimal Quantization Strategy for Reducing Variance

In the previous section, we have assumed uniformly distributed quantization points. We now investigate the choice of quantization points and present an optimal strategy to minimize the quantization variance term $\mathcal{TV}(\mathbf{a}_t)$.

Problem Setting. Assume a set of real numbers $\Omega = \{x_1,\ldots,x_N\}$ with cardinality N. WLOG, assume that all numbers are in $[0,1]$ and that $x_1 \leq \ldots \leq x_N$.

The goal is to partition $\mathcal{I} = \{I_j\}_{j=1}^s$ of $[0,1]$ into s disjoint intervals, so that if we randomly quantize every $x\in I_j$ to an endpoint of I_j, the variance is minimal over all possible partitions of $[0,1]$ into s intervals. Formally:

$$\begin{aligned}\min_{\mathcal{I}:|\mathcal{I}|=s} \quad & \mathcal{MV}(\mathcal{I}) := \frac{1}{N}\sum_{j=1}^{s}\sum_{x_i\in I_j}\mathrm{err}(x_i,I_j) \\ \text{s.t.} \quad & \bigcup_{j=1}^{s} I_j = [0,1], \quad I_j\cap l_k = \emptyset \text{ for } k\neq j,\end{aligned} \tag{8}$$

where $\mathrm{err}(x,I) = (b-x)(x-a)$ is the variance for point $x\in I$ if we quantize x to an endpoint of $I=[a,b]$. That is, $\mathrm{err}(x,I)$ is the variance of the (unique) distribution D supported on a,b so that $\mathbb{E}_{X\sim D}[X] = x$.

Given an interval $I\subseteq[0,1]$, we let Ω_I be the set of $x_j\in\Omega$ contained in I. We also define $\mathrm{err}(\Omega,I) = \sum_{x_j\in I}\mathrm{err}(x_j,I)$. Given a partition $\mathcal{I}$ of $[0,1]$, we let $\mathrm{err}(\Omega,\mathcal{I}) = \sum_{I\in\mathcal{I}}\mathrm{err}(\Omega,I)$. We let the optimum solution be $\mathcal{I}^* = \mathrm{argmin}_{|\mathcal{I}|=k}\,\mathrm{err}(\Omega,\mathcal{I})$, breaking ties randomly.

3.1. Dynamic Programming

We first present a dynamic programming algorithm that solves the above problem in an exact way. In the next subsection, we present a more practical approximation algorithm that only needs to scan all data points *once*.

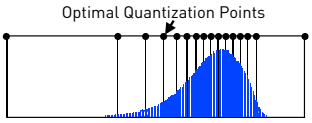


Figure 3. Optimal quantization points calculated with dynamic programming given a data distribution.

This optimization problem is non-convex and non-smooth. We start from the observation that there exists an optimal solution that places endpoints *at input points*.

Lemma 3. *There is a $\mathcal{I}^*$ so that all endpoints of any $I \in \mathcal{I}^*$ are in $\Omega \cup \{0, 1\}$.*

Therefore, to solve the problem in an exact way, we just need to select a subset of data points in Ω as quantization points. Define $T(k, m)$ be the optimal total variance for points in $[0, d_m]$ with k quantization levels choosing $d_m = x_m$ for all $m = 1, 2, \cdots, N$. Our goal is to calculate $T(s, N)$. This problem can be solved by dynamic programing using the following recursion

$$T(k, m) = \min_{j \in \{k-1, k, \cdots, m-1\}} T(k-1, j) + V(j, m),$$

where $V(j, m)$ denotes the total variance of points falling in the interval $[d_j, d_m]$. The complexity of calculating the matrix $V(\cdot, \cdot)$ is $O(N^2 + N)$ and the complexity of calculating the matrix $T(\cdot, \cdot)$ is $O(kN^2)$. The memory cost is $O(kN + N^2)$.

3.2. Heuristics

The exact algorithm has a complexity that is quadratic in the number of data points, which may be impractical. To make our algorithm practical, we develop an approximation algorithm that only needs to scan all data points once and has linear complexity to N.

Discretization. We can discretize the range $[0, 1]$ into M intervals, i.e., $[0, d_1), [d_1, d_2), \cdots, [d_{M-1}, 1]$ with $0 < d_1 < d_2 < \cdots < d_{M-1} < 1$. We then restrict our algorithms to only choose k quantization points within these M points, instead of all N points in the exact algorithm. The following result bounds the quality of this approximation.

Theorem 2. *Let the maximal number of data points in each "small interval" (defined by $\{d_m\}_{m=1}^{M-1}$) and the maximal length of small intervals be bounded by bN/M and a/M, respectively. Let $\mathcal{I}^* := \{l_j^*\}_{k=1}^{k-1}$ and $\hat{\mathcal{I}}^* := \{\hat{l}_k^*\}_{k=1}^{k-1}$ be the optimal quantization to* (8) *and the solution with discretization. Let cM/k be the upper bound of the number of small intervals crossed by any "large interval" (defined by $\mathcal{I}^*$). Then we have the discretization error bounded by*

$$\mathcal{MV}(\hat{\mathcal{I}}^*) - \mathcal{MV}(\mathcal{I}^*) \leq \frac{a^2 bk}{4M^3} + \frac{a^2 bc^2}{Mk}.$$

Theorem 2 suggests that the mean variance using the discrete variance-optimal quantization will converge to the optimal with the rate $O(1/Mk)$.

Dynamic Programming with Candidate Points. Notice that we can apply the same dynamic programming approach given M candidate points. In this case, the total computational complexity becomes $O((k+1)M^2 + N)$, with memory cost $O(kM + M^2)$. Also, to find the optimal quantization, we only need to scan all N numbers once. Figure 3 illustrates an example output for our algorithm.

2-Approximation in Almost-Linear Time. In the full version of this paper (Zhang et al., 2016), we present an algorithm which, given Ω and k, provides a split using at most $4k$ intervals, which guarantees a 2-approximation of the optimal variance for k intervals, using $O(N \log N)$ time. This is a new variant of the algorithm by (Acharya et al., 2015) for the histogram recovery problem. We can use the $4k$ intervals given by this algorithm as candidates for the DP solution, to get a general 2-approximation using k intervals in time $O(N \log N + k^3)$.

3.3. Applications to Deep Learning

In this section, we show that it is possible to apply optimal quantization to training deep neural networks.

State-of-the-art. We focus on training deep neural networks with a quantized model. Let $\mathcal{W}$ be the model and $l(\mathcal{W})$ be the loss function. State-of-the-art quantized networks, such as XNOR-Net and QNN, replace $\mathcal{W}$ with the quantized version $Q(\mathcal{W})$, and optimize for

$$\min_{\mathcal{W}} l(Q(\mathcal{W})).$$

With a properly defined $\frac{\partial Q}{\partial \mathcal{W}}$, we can apply the standard backprop algorithm. Choosing the quantization function Q is an important design decision. For 1-bit quantization, XNOR-Net searches the optimal quantization point. However, for multiple bits, XNOR-Net, as well as other approaches such as QNN, resort to uniform quantization.

Optimal Model Quantization for Deep Learning. We can apply our optimal quantization strategy and use it as the quantization function Q in XNOR-Net. Empirically, this results in quality improvement over the default *multi-bits* quantizer in XNOR-Net. In spirit, our approach is similar to the 1-bit quantizer of XNOR-Net, which is equivalent to our approach when the data distribution is symmetric—we extend this to multiple bits in a principled way. Another related work is the uniform quantization strategy in *log domain* (Miyashita et al., 2016), which is similar to our approach when the data distribution is "log uniform." However, our approach does not rely on any specific assumption

Regression			
Dataset	Training Set	Testing Set	# Features
Synthetic 10	10,000	10,000	10
Synthetic 100	10,000	10,000	100
Synthetic 1000	10,000	10,000	1,000
YearPrediction	463,715	51,630	90
cadata	10,000	10,640	8
cpusmall	6,000	2,192	12
Classification			
Dataset	Training Set	Testing Set	# Features
cod-rna	59,535	271,617	8
gisette	6,000	1,000	5,000
Deep Learning			
Dataset	Training Set	Testing Set	# Features
CIFAR-10	50,000	10,000	$32 \times 32 \times 3$
Tomographic Reconstruction			
Dataset	# Projections	Volumn Size	Proj. Size
	128	128^3	128^3

Table 1. Dataset statistics.

of the data distribution. Han et al. (2016) use k-means to compress the model for *inference* —k-means optimizes for a similar, but different, objective function than ours. In this paper, we develop a dynamic programming algorithm to do optimal stochastic quantization efficiently.

4. Experiments

We now provide an empirical validation of our ZipML framework.

Experimental Setup. Table 1 shows the datasets we use. Unless otherwise noted, we always use diminishing step-sizes α/k, where k is the current number of epoch. We tune α for the full precision implementation, and use the same initial step size for our low-precision implementation. (Theory and experiments imply that the low-precision implementation often favors smaller step size. Thus we do not tune step sizes for the low-precision implementation, as this can only improve the accuracy of our approach.)

Summary of Experiments. Due to space limitations, we only report on **Synthetic 100** for regression, and on **gisette** for classification. The full version of this paper (Zhang et al., 2016) contains (1) several other datasets, and discusses (2) different factors such as impact of the number of features, and (3) refetching heuristics. The FPGA implementation and design decisions can be found in (Kara et al., 2017).

4.1. Convergence on Linear Models

We validate that (1) with double sampling, SGD with low precision converges—in comparable empirical convergence rates—to the same solution as SGD with full precision; and (2) implemented on FPGA, our low-precision prototype achieves significant speedup because of the decrease in bandwidth consumption.

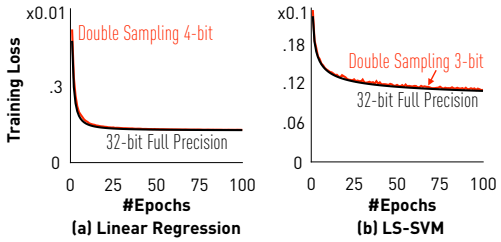


Figure 4. Linear models with end-to-end low precision.

Convergence. Figure 4 illustrates the result of training linear models: (a) linear regression and (b) least squares SVMs, with end-to-end low-precision and full precision. For low precision, we pick the smallest number of bits that results in a smooth convergence curve. We compare the final training loss in both settings and the convergence rate.

We see that, for both linear regression and least squares SVM, using 5- or 6-bit is always enough to converge to the same solution with comparable convergence rate. This validates our prediction that double-sampling provides an unbiased estimator of the gradient. Considering the size of input samples that we need to read, we could potentially save 6–8$\times$ memory bandwidth compared to using 32-bit.

Speedup. We implemented our low-precision framework on a state-of-the-art FPGA platform. The detailed implementation is described in (Kara et al., 2017). This implementation assumes the input data is already quantized and stored in memory (data can be quantized during the first epoch).

Figure 5 illustrates the result of (1) our FPGA implementation with quantized data, (2) FPGA implementation with 32-bit data, and (3) Hogwild! running with 10 CPU cores. Observe that all approaches converge to the same solution. FPGA with quantized data converges 6-7$\times$ faster than FPGA with full precision or Hogwild!. The FPGA implementation with full precision is memory-bandwidth bound, and by using our framework on quantized data, we save up to 8$\times$ memory-bandwidth, which explains the speedup.

Impact of Mini-Batching. We now validate the"sensitivity" of the algorithm to the precision under batching. Equation 7 suggests that, as we increase batch size, the variance term corresponding to input quantization may start to dominate the variance of the stochastic gradient. However, in practice and for reasonable parameter settings, we found this does not occur: convergence trends for small batch size, e.g. 1, are the

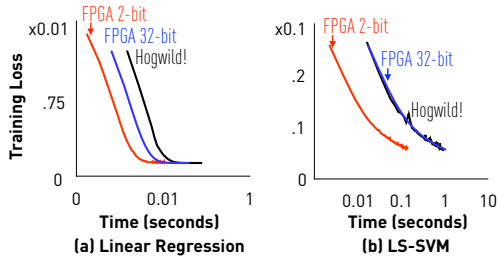


Figure 5. FPGA implementation of linear models.

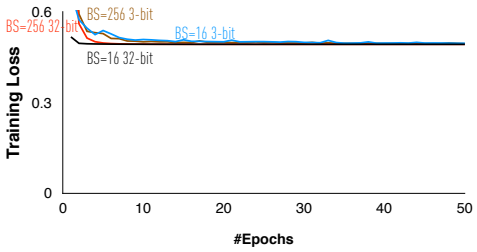


Figure 6. Impact of Using Mini-Batch. BS=Batch Size.

same as for larger sizes, e.g. 256. Figure 6 shows that, if we use larger mini-batch size (256), we need more epochs than using smaller mini-batch size (16) to converge, but for the quantized version, actually the one with larger mini-batch size converges faster.

4.2. Data-Optimal Quantization Strategy

We validate that, with our data-optimal quantization strategy, we can significantly decrease the number of bits that double-sampling requires to achieve the same convergence. Figure 7(a) illustrates the result of using 3-bit and 5-bit for uniform quantization and optimal quantization on the **YearPrediction** dataset. Here, we only consider quantization on data, but not on gradient or model, because to compute the data-optimal quantization, we need to have access to all data and assume the data doesn't change too much, which is not the case for gradient or model. The quantization points are calculated for each feature for both uniform quantization and optimal quantization. We see that, while uniform quantization needs 5-bit to converge smoothly, optimal quantization only needs 3-bit. We save almost $1.7\times$ number of bits by just allocating quantization points carefully.

Comparision with uniform quantization. We validate that, with our data-optimal quantization strategy, we can significantly increase the convergence speed.

Figure 8 illustrates the result of training linear regression models: with uniform quantization points and optimal quantization points. Here, notice that we only quantize data, but not gradient or model. We see that, if we use same number of bits, optimal quantization always converges faster than uniform quantization and the loss curve is more stable, because the variance induced by quantization is smaller. As a result, with our data-optimal quantization strategy, we can either (1) get up to $4\times$ faster convergence speed with the same number of bits; or (2) save up to $1.7\times$ bits while getting the same convergence speed.

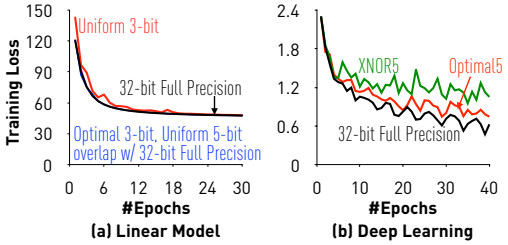


Figure 7. Optimal quantization strategy.

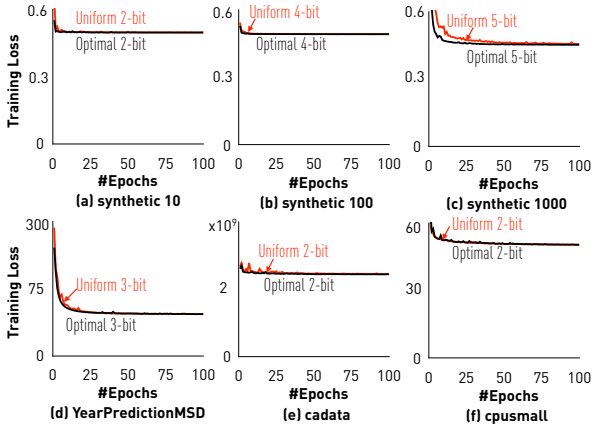


Figure 8. Linear regression with *quantized data* on multiple datasets.

We also see from Figure 8 (a) to (c) that if the dataset has more features, usually we need more bits for quantization, because the variance induced by quantization is higher when the dimensionality is higher.

4.3. Extensions to Deep Learning

We validate that our data-optimal quantization strategy can be used in training deep neural networks. We take Caffe's CIFAR-10 tutorial (Caf) and compare three different quantization strategies: (1) Full Precision, (2) XNOR5, a XNOR-Net implementation that, following the multi-bits strategy in the original paper, quantizes data into five uniform levels, and (3) Optimal5, our quantization strategy with five optimal quantization levels. As shown in Figure 7(b), Optimal5 converges to a significantly lower train-

ing loss compared with XNOR5. Also, Optimal5 achieves >5 points higher testing accuracy over XNOR5. This illustrates the improvement obtainable by training a neural network with a carefully chosen quantization strategy.

5. Related Work

There has been significant work on "low-precision SGD" (De Sa et al., 2015; Alistarh et al., 2016). These results provide theoretical guarantees only for quantized gradients. The model and input samples, on the other hand, are much more difficult to analyze because of the non-linearity. We focus on *end-to-end* quantization, for all components.

Low-Precision Deep Learning. Low-precision training of deep neural networks has been studied intensively and many heuristics work well for a subset of networks. OneBit SGD (Seide et al., 2014) provides a gradient compression heuristic developed in the context of deep neural networks for speech recognition. There are successful applications of end-to-end quantization to training neural networks that result in little to no quality loss (Hubara et al., 2016; Rastegari et al., 2016; Zhou et al., 2016; Miyashita et al., 2016; Li et al., 2016; Gupta et al., 2015). They quantize weights, activations, and gradients to low precision (e.g., 1-bit) and revise the back-propagation algorithm to be aware of the quantization function. The empirical success of these works inspired this paper, in which we try to provide a *theoretical* understanding of end-to-end low-precision training for machine learning models. Another line of research concerns inference and model compression of a pre-trained model (Vanhoucke et al., 2011; Gong et al., 2014; Han et al., 2016; Lin et al., 2016; Kim & Smaragdis, 2016; Kim et al., 2015; Wu et al., 2016). In this paper, we focus on training and leave the study of inference for future work.

Low-Precision Linear Models. Quantization is a fundamental topic studied by the DSP community, and there has been research on linear regression models in the presence of quantization error or other types of noise. For example, Gopi et al. (2013) studied compressive sensing with binary quantized measurement, and a two-stage algorithm was proposed to recover the sparse high-precision solution up to a scale factor. Also, the classic errors-in-variable model (Hall, 2008) could also be relevant if quantization is treated as a source of "error." In this paper, we scope ourselves to the context of stochastic gradient descent, and our insights go beyond simple linear models. For SVM the straw man approach can also be seen as a very simple case of kernel approximation (Cortes et al., 2010).

Other Related Work. Precision of data representation is a key design decision for configurable hardwares such as FPGA. There have been attempts to lower the precision when training on such hardware (Kim et al., 2011). These results are mostly empirical; we aim at providing a theoretical understanding, which enables new algorithms.

6. Discussion and Future Work

Our motivating question was whether end-to-end low-precision data representation can enable efficient computation with convergence guarantees. We show that ZipML, a relatively simple stochastic quantization framework can achieve this for linear models. With this setting, as little as two bits per model dimension are sufficient for good accuracy, and can enable a fast FPGA implementation.

Non-Linear Models. We mainly discussed linear models (e.g. linear regression) in this paper. The natural question is that can we extend our ZipML framework to non-linear models (e.g. logistic regression or SVM) which are arguably more commonly used. In the full version on this paper, we examine this problem, and find that our framework can be generalized to non-linear settings, and that in practice 8-bit is sufficient to achieve good accuracy on a variety of tasks, such as SVM and logistic regression. However, we notice that a strawman approach, which applies naive stochastic rounding over the input data to just 8-bit precision, converges to similar results, without the added complexity. It is interesting to consider the rationale behind these results. Our framework is based on the idea of *unbiased approximation* of the original SGD process. For linear models, this is easy to achieve. For non-linear models, this is harder, and we focus on guaranteeing arbitrarily low bias. However, for a variety of interesting functions such as hinge loss, guaranteeing low bias requires complex approximations. In turn, these increase the variance. The complexity of the approximation and the resulting variance increase force us to increase the *density* of the quantization, in order to achieve good approximation guarantees.

Hardware Selection. We choose to realize our implementation using FPGA because of its flexibility in dealing with low-precision arithmetic, while CPU or GPU can only do at least 8-bit computation efficiently. However, we do observe speed up in other environments – for example, the double sampling techniques are currently being applied to sensor networks with embedded GPUs and CPUs to achieve similar speedup. We are currently conducting an architecture exploration study which aims at understanding the system trade-off between FPGA, CPU, and GPU. This requires us to push the implementation on all three architectures to the extreme. We expect this study will soon provide a full systematic answer to the question that on which hardware can we get the most from the ZipML framework.

Acknowledgements

CZ gratefully acknowledges the support from the Swiss National Science Foundation NRP 75 407540_167266, NVIDIA Corporation for its GPU donation, and Microsoft Azure for Research award program.

References

Caffe CIFAR-10 tutorial. `http://caffe.berkeleyvision.org/gathered/examples/cifar10.html`.

Acharya, Jayadev, Diakonikolas, Ilias, Hegde, Chinmay, Li, Jerry Zheng, and Schmidt, Ludwig. Fast and near-optimal algorithms for approximating distributions by histograms. In *PODS*, 2015.

Alistarh, Dan, Li, Jerry, Tomioka, Ryota, and Vojnovic, Milan. QSGD: Randomized Quantization for Communication-Optimal Stochastic Gradient Descent. *arXiv:1610.02132*, 2016.

Bubeck, Sébastien. Convex optimization: Algorithms and complexity. *Foundations and Trends® in Machine Learning*, 2015.

Cortes, Corinna, Mohri, Mehryar, and Talwalkar, Ameet. On the impact of kernel approximation on learning accuracy. In *AISTATS*, 2010.

De Sa, Christopher M, Zhang, Ce, Olukotun, Kunle, and Ré, Christopher. Taming the wild: A unified analysis of hogwild-style algorithms. In *NIPS*, 2015.

Gong, Yunchao, Liu, Liu, Yang, Ming, and Bourdev, Lubomir. Compressing deep convolutional networks using vector quantization. *arXiv:1412.6115*, 2014.

Gopi, Sivakant, Netrapalli, Praneeth, Jain, Prateek, and Nori, Aditya. One-bit compressed sensing: Provable support and vector recovery. In *ICML*, 2013.

Gupta, Suyog, Agrawal, Ankur, Gopalakrishnan, Kailash, and Narayanan, Pritish. Deep learning with limited numerical precision. In *ICML*, 2015.

Hall, Daniel B. Measurement error in nonlinear models: A modern perspective (2nd ed.). *Journal of the American Statistical Association*, 2008.

Han, Song, Mao, Huizi, and Dally, William J. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. In *ICLR*, 2016.

Hubara, Itay, Courbariaux, Matthieu, Soudry, Daniel, El-Yaniv, Ran, and Bengio, Yoshua. Quantized neural networks: Training neural networks with low precision weights and activations. *arXiv:1609.07061*, 2016.

Kara, Kaan, Alistarh, Dan, Zhang, Ce, Mutlu, Onur, and Alonso, Gustavo. Fpga accelerated dense linear machine learning: A precision-convergence trade-off. In *FCCM*, 2017.

Kim, Jung Kuk, Zhang, Zhengya, and Fessler, Jeffrey A. Hardware acceleration of iterative image reconstruction for x-ray computed tomography. In *ICASSP*, 2011.

Kim, Minje and Smaragdis, Paris. Bitwise neural networks. *arXiv:1601.06071*, 2016.

Kim, Yong-Deok, Park, Eunhyeok, Yoo, Sungjoo, Choi, Taelim, Yang, Lu, and Shin, Dongjun. Compression of deep convolutional neural networks for fast and low power mobile applications. *arXiv:1511.06530*, 2015.

Li, Fengfu, Zhang, Bo, and Liu, Bin. Ternary weight networks. *arXiv:1605.04711*, 2016.

Lin, Darryl, Talathi, Sachin, and Annapureddy, Sreekanth. Fixed point quantization of deep convolutional networks. In *ICML*, 2016.

Miyashita, Daisuke, Lee, Edward H, and Murmann, Boris. Convolutional neural networks using logarithmic data representation. *arXiv:1603.01025*, 2016.

Rastegari, Mohammad, Ordonez, Vicente, Redmon, Joseph, and Farhadi, Ali. Xnor-net: Imagenet classification using binary convolutional neural networks. In *ECCV*, 2016.

Seide, Frank, Fu, Hao, Droppo, Jasha, Li, Gang, and Yu, Dong. 1-bit stochastic gradient descent and application to data-parallel distributed training of speech dnns. In *Interspeech*, 2014.

Vanhoucke, Vincent, Senior, Andrew, and Mao, Mark Z. Improving the speed of neural networks on cpus. In *NIPS Workshop on Deep Learning and Unsupervised Feature Learning*, 2011.

Wu, Jiaxiang, Leng, Cong, Wang, Yuhang, Hu, Qinghao, and Cheng, Jian. Quantized convolutional neural networks for mobile devices. In *CVPR*, 2016.

Zhang, Hantian, Li, Jerry, Kara, Kaan, Alistarh, Dan, Liu, Ji, and Zhang, Ce. The zipml framework for training models with end-to-end low precision: The cans, the cannots, and a little bit of deep learning. *arXiv:1611.05402*, 2016.

Zhou, Shuchang, Wu, Yuxin, Ni, Zekun, Zhou, Xinyu, Wen, He, and Zou, Yuheng. Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients. *arXiv:1606.06160*, 2016.