Master Thesis

The ZipML Framework for Training Models with End-to-End Low Precision

Author(s):
Zhang, Hantian

Publication Date:
2017

Permanent Link:

https://doi.org/10.3929/ethz-a-010890124 →

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Master's Thesis Nr. 160

Systems Group, Department of Computer Science, ETH Zurich

The ZipML Framework for Training Models with End-to-End Low Precision

by

Hantian Zhang

Supervised by

Prof. Ce Zhang

September 2016–March 2017

inf | Informatik
Computer Science

Acknowledgments

I would like to express my special appreciation to my advisor Professor Ce Zhang first and foremost. He shows me how interesting and exciting doing researches can be and teaches me everything about how to do research properly with great patience, regardless of how many mistakes I made.

I am also grateful for my amazing collaborators, Dan Alistarh, Kaan Kara, Jerry Li, Ji Liu and Kevin Schawinski, without whose effort this research cannot be done. I learn a lot from Kevin's enthusiasm about doing research and his astronomy primer. Kaan is the first PhD student I met in the systems group and helps me both in doing research and in daily life. I enjoy discussions with Dan, Ji, and Jerry, who provide many theoretical insights that are really beautiful.

I dedicate this thesis work to my parents, who have been giving me their endless and selfless support and encouragement during my master's studies and my whole life.

Previously published works [49, 39] provide most results mentioned in this thesis. Some descriptions are *directly* from these papers. These papers are joint efforts with several authors, including: D. Alistarh, L. Fowler, K. Kara, J. Li, J. Liu, G. Santhanam, K. Schawinski, C. Zhang.

Abstract

We first present an application where we helped our astrophysicist collaborators to recover features from artificially degraded images with worse seeing and higher noise than the original with a performance which far exceeds simple deconvolution by training a generative adversarial network (GAN) on galaxy images. However, training time is limiting our potential to train on larger data sets. It takes 2 hours to train a GAN using 4105 galaxy images for 20 iterations on an NVIDIA TITAN X GPU. We ask the question: *Can we speed up our machine learning training process by reducing the precision of data representation?*

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?*

For linear models, the answer is yes. We develop a simple framework based on one simple but novel strategy called double sampling. Our 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× 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× in data movement. When training deep networks with quantized models, we achieve higher accuracy than the state-of-the-art XNOR-Net.

Last, we extend our framework through approximation to non-linear models, such as SVM. We show that, although using low-precision data induces bias, we can appropriately bound and control the bias. We find in practice *8-bit* precision is often sufficient to converge to the correct solution. Interestingly, however, in practice we notice that our framework does not always outperform the naive rounding approach. We discuss this negative result in detail.

Contents

Chapter 1

Introduction

Any telescope observation of astrophysical objects is limited by the noise in the image, driven both by the detector used and the sky background. We introduce a method that can reliably recover features in images of galaxies by training a generative adversarial network (GAN) on galaxy images. However, training time is limiting our potential to train on larger data sets. It takes 2 hours to train a GAN using 4105 galaxy images for 20 iterations on an NVIDIA TITAN X GPU. We ask the question: *Can we speed up our machine learning training process by reducing the precision of data representation?*

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 [19, 36, 50, 33]. In this context, the motivating question behind our work is: *When training general machine learning models, can we lower the precision of data representation, communication, and computation, while maintaining provable guarantees?*

In this thesis, we develop a general framework to answer this question, and present both positive and negative results obtained in the context of this framework. Figure 1.1 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× 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× for tomographic reconstruction, while obtaining a negligi-

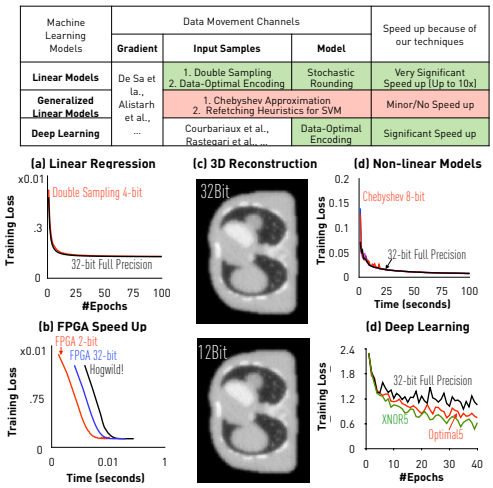


Figure 1.1: Overview of theoretical results and highlights of empirical results. See Introduction for details.

ble quality decrease. Elements of our framework generalize to (d) non-linear models and (e) model compression for training deep learning models. In the following, we describe our technical contributions in more detail.

1.1 Summary of Technical Contributions

1.1.1 GAN recovery of galaxy features

We present a method using a GAN trained on galaxy images that can recover features from artificially degraded images with worse seeing and higher noise than the original with a performance which far exceeds simple deconvolution. The ability to better recover detailed features such as galaxy morphology from low-signal-to-noise and low angular resolution imaging data significantly increases our ability to study existing data sets of astrophysical objects as well as future observations with observatories such as the Large Synoptic Sky Telescope (LSST) and the *Hubble* and *James Webb* space telescopes.

1.1.2 The ZipML Framework for Training Models with End-to-End Low Precision

We consider the following problem in training generalized linear models:

$$\min_{\mathbf{x}} : \quad \frac{1}{2K} \sum_{k=1}^{K} l(\mathbf{a}_k^\top \mathbf{x}, b_k)^2 + R(\mathbf{x}), \tag{1.1}$$

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:

$$\mathbf{x}_{t+1} \leftarrow \text{prox}_{\gamma R(\cdot)}\left(\mathbf{x}_t - \gamma Q_g(\mathbf{g}_k(Q_m(\mathbf{x}_t), Q_s(\mathbf{a}_t)))\right), \tag{1.2}$$

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. The **(+)** sign denotes a "positive result," where we achieve significant practical speedup; it is **(−)** otherwise.

(+) 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× 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 model quantization 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 [18]. To the best of our knowledge, this is the first time such optimal quantization strategies have been applied to training.

(–) Non-Linear Models. We extend our results to non-linear models, such as SVM. We can stretch our multiple-sampling strategy to provide unbiased estimators for any polynomials, at the cost of increased variance. Building further, we employ Chebyshev polynomials to approximate the gradient of *arbitrary smooth loss functions* within arbitrarily low bias, and to provide bounds on the error of an SGD solution obtained from low-precision samples.

Further, we examine whether this approach can be applied to non-smooth loss functions, such as SVM. We find that the machinery described above does not apply, for fundamental reasons. We use ideas from streaming and dimensionality reduction to develop a variant that is provably correct for non-smooth loss functions. We can show that, under reasonable assumptions, the added communication cost of supporting non-smooth functions is negligible.

In practice, using this technique we are able to go as low as 8-bit precision for SVM and logistic regression. However, we notice that the straw man approach, which applies naive stochastic rounding over the input data to just 8-bit precision, converges to similar results, without the added complexity. This negative result is explained by the fact that, to approximate non-linearities such as the step function or the sigmoid well, our framework needs both high degree Chebyshev polynomials and relatively large samples.

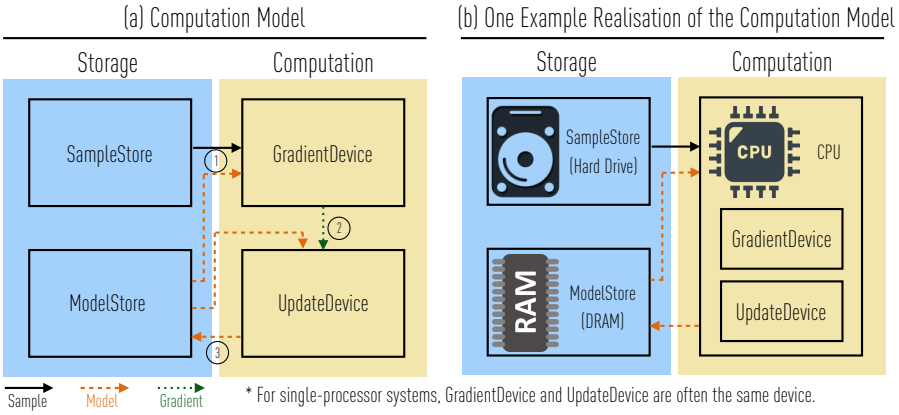


Figure 1.2: (a) A Schematic Representation of the Computation Model and (b) An Example Realisation of the Computation Model. Three types of data, namely (1) sample, (2) model, and (3) gradient, moves in the system in three steps as illustrated in (a). Given different parameters of the computation model, such as computational power and memory bandwidth, the system bottleneck may vary. For example, in realisation (b) having a hard drive, DRAM, and a modern CPU, it is likely that the bottleneck when training a dense generalized linear model is the memory bandwidth between SampleStore and GradientDevice.

Chapter 2

An Application: GAN recovery of galaxy features

2.1 Introduction

Any telescope observation of astrophysical objects is limited by the noise in the image, driven both by the detector used and the sky background. Similarly, the observation is limited in angular resolution by the resolving power of the telescope ($R \sim \lambda / D$) and, if taken from the ground, by the distortions caused by the moving atmosphere (the "seeing"). The total blurring introduced by the combination of the telescope and the atmosphere is described by the point spread function (PSF). An image taken by a telescope can therefore be thought of as a convolution of the true light distribution with this point spread function plus the addition of various sources of noise. The Shannon-Nyquist sampling theorem [35, 41] limits the ability of deconvolution techniques in removing the effect of the PSF, particularly in the presence of noise [32, 10, 43].

Deconvolution has long been known as an "ill-posed" inverse problem because there is often no unique solution if one follows the signal processing approach of backwards modelling [32, 31, 25, 26, 8]. Another standard practice in tackling inverse problems like these is integrating priors using domain knowledge in forward modelling. For example, if the algorithm knows what a galaxy should look like or it knows the output needs to have certain properties such as being "sharp", it will make more informative decisions when choosing among all possible solutions. In this chapter we demonstrate a method using machine learning to automatically introduce such priors. This method can reliably recover features in images of galaxies. We find that machine learning techniques can go beyond this limitation of deconvolutions — by training on higher quality data, a machine learning system can learn to recover information from poor quality data by effectively building priors.

2.2 Method

Our general method is agnostic as to the specific machine learning algorithm used. In this chapter, we choose to use conditional Generative Adversarial Networks (GAN), a state-of-the-art deep learning algorithm for image-to-image translation. In this work, we adopted a standard GAN architecture; therefore we only briefly introduce GAN and interested readers can consult [37] and [14] for details.

In the training phase, the GAN takes as input a set of image pairs—in our case, one image which is degraded (by this we mean: convolved with a worse PSF, or blurred, and with added noise) and the same image without such degradation. The GAN then tries to "learn" to recover the degraded image by minimizing the difference between the recovered image and the non-degraded image. The function that measures the difference between the two images, which is often called the loss function, is often something simple such as the Euclid distance but can be a more sophisticated function. In the case of a GAN, this function is another neural network (hence the name adversarial) whose goal is to distinguish the recovered image from a non-degraded image. These two neural networks are trained at the same time. This allows the system to learn sophisticated loss functions automatically without hand-engineering.[1] In the testing phase, the GAN takes a different set of degraded images and recovers them.

One remaining challenge is how to generate pairs of images with and without degradation for the training phase. In our framework, we take advantage of the centuries of study of the noise introduced by the telescope and the atmosphere to *weakly supervise* a GAN network by *simulating* the blurring process automatically. This allows us to easily harvest a large training set automatically without any human intervention. Furthermore, it allows us to automatically scale our system, and arguably achieve better quality, when future large-scale sky survey data from e.g. LSST [29] or *Euclid* [24] are available. We outline the method in Figure 2.1.

We select a sample of 4,550 galaxies from the Sloan Digital Sky Survey Data Release 12 [48, 2] in the redshift range $0.01 < z < 0.02$ and conduct $10\times$ cross validation for all of our experiments (each fold contains 4,105 images for training and 455 for testing). We obtain the g, r and i-band images for these objects and process them using an asinh stretch ($y = \mathsf{asinh}(10\mathsf{x})/3$) to produce 3-band RGB images. The transform to asinh stretch rather than keeping the linear data follows the best practice of making the range of input values for a neural network comparable across images. This step has been shown to

[1] It is known that if one uses Euclid distance for image recovery, this often produces blurred images because Euclid distance is uniform over the whole image [37], thus a more sophisticated loss function could improve the system.

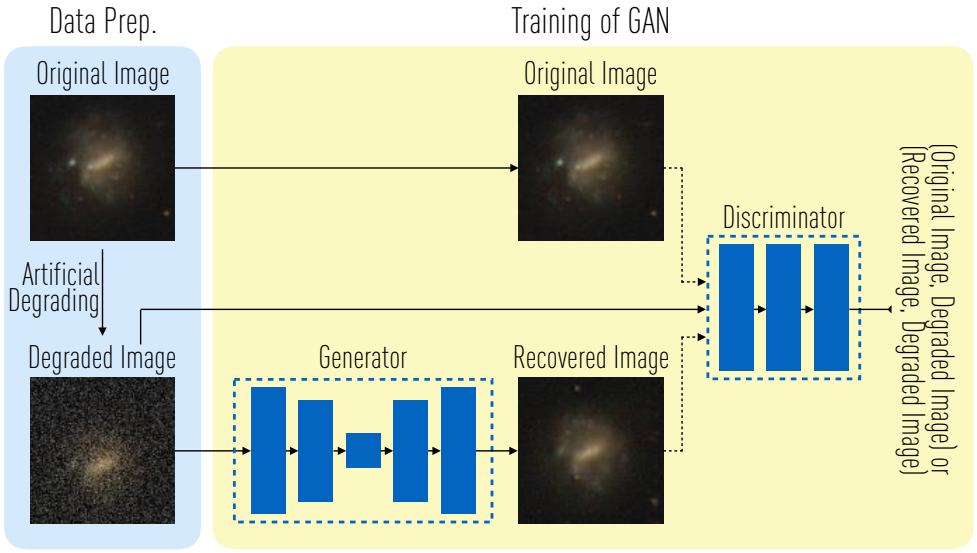


Figure 2.1: Schematic illustration of the training process of our method. The input is a set of original images. From these we automatically generate degraded images, and train a Generative Adversarial Network. In the testing phase, only the generator will be used to recover images.

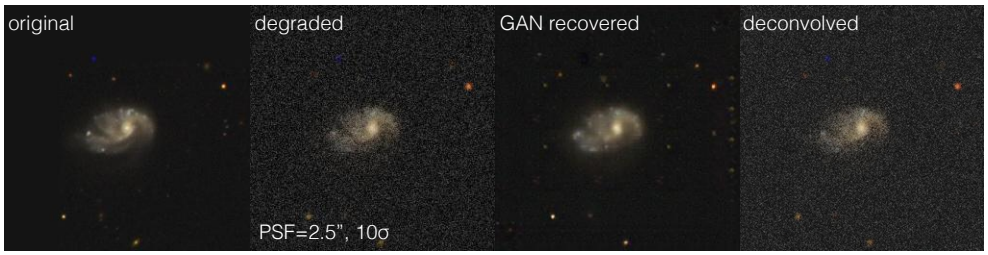


Figure 2.2: We show the results obtained for one example galaxy. From left to right: the original SDSS image, the degraded image with a worse PSF and higher noise level (indicating the PSF and noise level used), the image as recovered by the GAN, and for comparison, the result of a deconvolution. This figure visually illustrates the GAN's ability to recover features which conventional deconvolutions cannot.

help make the training procedure faster and easier in other applications [42]. We note that this process involved clipping extreme pixel values and in principle makes it impossible to fully recover the original flux calibration; exploring how to avoid this while not degrading the neural net training performance is an interesting project.

In order to test the performance of the GAN, we generate a grid of training sets from the galaxy sample. In each training set, we convolve the images with a Gaussian PSF with a full-width at half-maximum (FWHM) of FWHM=$[1.4, 1.8, 2.0, 2.5]$ arcsec. The median seeing of SDSS images is $\sim$ 1.4 arcsec so we explore images of effectively the same resolution all the way to a significantly worse seeing of 2.5 arcsec. After convolving the images with a Gaussian filter representing worse seeing, we adjust the noise level, first restoring it to that of the original image, and then increasing it so that $\sigma_{\text{new}} = [1.0, 1.2, 2.0, 5.0, 10.0]\sigma_{\text{original}}$ to mimic shallower images. We train the GAN using open source code released by [37] with TITAN X PASCAL GPUs. Training finishes in 2 hours per setting per fold (200 hours in total).

2.3 Results

We evaluate our method both quantitatively and qualitatively. Quantitatively, we measure the Peak Signal Noise Ratio (PSNR, [47]) of the blurred image and the recovered image. The PSNR is a popular quantitative measure for image recovery [47]. It is defined as the ratio between the maximum possible power of a signal (original image) and the power of the noise (difference between recovered and original image). When the blurred image contains much noise, a case that is known to be a challenge for deconvolution-based approaches, our method can achieve a PSNR of 37.2dB. Blind deconvolution [5] [2] achieves a PSNR of 19.9dB and Lucy-Richardson deconvolution [38, 30] [3] achieves a PSNR of 18.7dB. We compare our and classic deconvolution approaches in Table 2.1. We also perform an experiment on the impact of the training set size, reported in Table 2.1d: the PSNR achieved increases as we increase the training set size from 5 images to 2,000.

For the qualitative analysis, we show example results in Figure 2.2 and 2.3 where we show the original image, the degraded image with additional noise and a larger PSF, the recovered image and the deconvolved image. We show sample spiral galaxies, early-type galaxies and galaxy mergers, each selected with various levels of degradation. These sample images show what our method is able to recover, and contrast it to the performance of deconvolution.

[2] We compare with blind deconvolution in Matlab. https://www.mathworks.com/help/images/ref/deconvblind.html

[3] We compare with Lucy-Richardson deconvolution in Matlab. https://www.mathworks.com/help/images/ref/deconvlucy.html

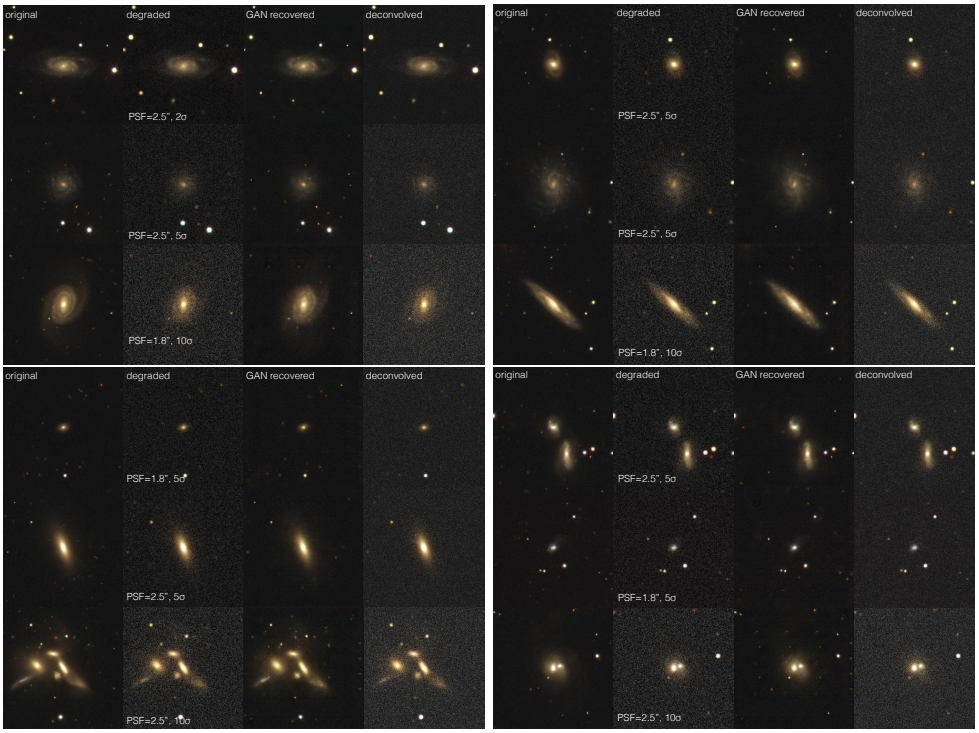


Figure 2.3: We show some further representative results for different galaxy types and with various levels of degradation. In each row, we show with the same layout as Figure 2.2. Since the GAN has been trained on images of galaxies with similar properties, it is able to recover details which the deconvolution cannot, such as star-forming regions, dust lanes and the shape of spiral arms. The top two panels are examples of spiral galaxies. The bottom-left panel shows early-type galaxies (including a dense cluster). The bottom-right panel shows galaxy mergers; note in particular that the GAN reconstruction makes it easier to identify these systems as merging, as opposed to being undisturbed or superpositions . For more detailed results, see Appendix A.

White noise level relative to original image	Gaussian PSF FWHM parameter in arcsec			
	1.4arcsec	1.8arcsec	2.2arcsec	2.5arcsec
$1.0\sigma_{\text{original}}$	39.6	39.8	39.2	38.9
$1.2\sigma_{\text{original}}$	40.0	39.5	39.5	38.9
$2.0\sigma_{\text{original}}$	38.9	39.4	39.2	38.8
$5.0\sigma_{\text{original}}$	37.8	38.1	38.1	37.8
$10.0\sigma_{\text{original}}$	35.4	37.3	37.3	36.9

(a) PSNR (dB) of GAN

White noise level relative to original image	Gaussian PSF FWHM parameter in arcsec			
	1.4arcsec	1.8arcsec	2.2arcsec	2.5arcsec
$1.0\sigma_{\text{original}}$	36.2	35.7	35.0	34.5
$1.2\sigma_{\text{original}}$	35.1	34.7	34.0	33.6
$2.0\sigma_{\text{original}}$	31.6	31.4	31.1	30.9
$5.0\sigma_{\text{original}}$	24.9	24.8	24.8	25.4
$10.0\sigma_{\text{original}}$	20.0	20.0	20.0	20.6

(b) PSNR (dB) of Blind Deconvolution

White noise level relative to original image	Gaussian PSF FWHM parameter in arcsec			
	1.4arcsec	1.8arcsec	2.2arcsec	2.5arcsec
$1.0\sigma_{\text{original}}$	34.7	34.8	34.9	35.0
$1.2\sigma_{\text{original}}$	33.3	33.5	33.6	33.7
$2.0\sigma_{\text{original}}$	29.5	29.7	29.9	30.0
$5.0\sigma_{\text{original}}$	23.0	23.1	23.3	23.5
$10.0\sigma_{\text{original}}$	18.8	18.8	18.9	19.0

(c) PSNR (dB) of Lucy-Richardson Deconvolution

# Images	5	10	100	2000
PSNR	34.6	36.0	37.7	37.9

(d) Impact of the Size of Training Set

Table 2.1: (a)-(c) PSNR of images recovered by GAN, Blind Deconvolution, and Lucy-Richardson Deconvolution. (d) Impact of the size of training set on the GAN.

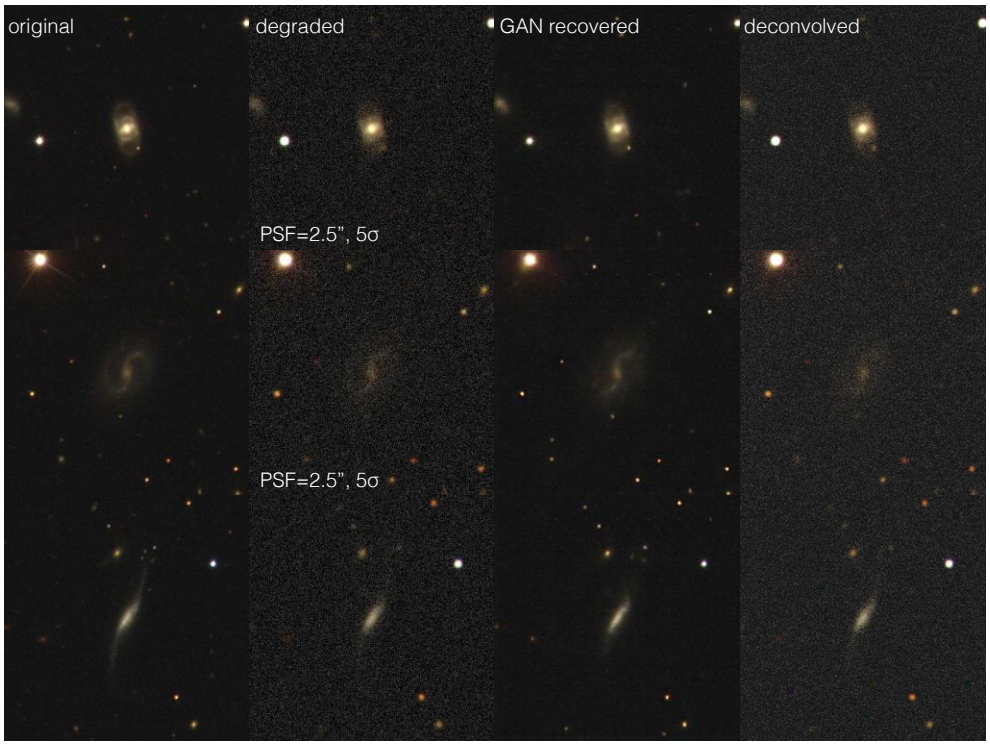


Figure 2.4: We show three examples where the GAN reconstruction fails at some level. Two of the three can be accounted for by the fact that the objects in question are rare, and so the training set did not sufficiently prepare the GAN to deal with them. In the top row is a rare kinematic structure [7]. The middle row is a barred spiral whose outer arms were so far below the noise that they could not be reconstructed. The bottom row is a tidally warped edge-on disk, a class of galaxies sufficiently rare that the training did not prepare the GAN to recognise it.

In Figure 2.4, we show some examples where the reconstruction method had problems. We stress that these are rare, and we present the results for all parts of the degradation grid evaluated on test galaxies in Appendix A, which contains the full output as online-only. In general, the GAN fails on rare objects which were absent or low in number in the training set, stressing that the performance of our method is strongly tied to the training set; it cannot reconstruct features it has not learned to recognize.

2.4 Discussion

Our method and those like it naturally suffer from some limitations. The main limitation is that the training set ultimately limits the ability to recover features. We trained on galaxies in the nearby universe, and applied the re-

sulting training to similar galaxies in the same redshift range taken with the same camera under similar conditions. If one were to apply it to galaxies at higher redshift, one could simulate effects such as k-correction and cosmological dimming, but the intrinsic morphology of galaxies at say $z \sim 2$ or $z \sim 6$ are fundamentally different from those at $z \sim 0$. One solution to this issue would be to train on images from simulations of galaxy formation at the appropriate epoch. Another limitation of our method is that sophisticated details such as weak lensing shear are impossible to recover via this route, as subtle distortions are truly irrecoverable. Similarly, if there are rare objects absent from the training set, the method may fail, as illustrated in Figure 2.4.

We have shown that it is possible to recover features from low quality imaging data of galaxies which are not possible to recover with a deconvolution. Despite the limitations, our method has enormous scope for application in astrophysics research as it increases the potential for research and discovery. Provided that a suitable training set is available, our method and further developments based on it can be applied to a wide range of existing and future data from SDSS, *Euclid* [24] and the LSST [29]. For example, using training sets from simulations, one could push to high redshift observation with the *Hubble* and *James Webb* space telescopes to analyze galaxies in the early universe.

We make all the code available and provide instructions to access a virtual machine which can reproduce the entire analysis at `http://space.ml/proj/GalaxyGAN.html`.

Apart from the challenge of recovering features from noisy image, we also face the challange of not begin able to use all available data because of the limitation of training time. It takes 2 hours to train a GAN using only 4105 galaxy images for 20 iterations on an NVIDIA TITAN X GPU. This motivates the research in the following chapters of this thesis. We try to understand if we can train models with low-precision data representation, while getting *the same result* and how much *speed up* can we get by using low-precision.

Chapter 3

Linear Models

In this chapter, 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

$$F(\mathbf{x}) = \underbrace{\frac{1}{K} \sum_{k=1}^{K} \|\mathbf{a}_k^\top \mathbf{x} - b_k\|_2^2}_{=:f(\mathbf{x})} + R(\mathbf{x}) \,, \tag{3.1}$$

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

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

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

Theorem 3.1 *[e.g., [6], 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* (3.1)*:*

$$F\left(\frac{1}{T}\sum_{t=0}^{T} \mathbf{x}_t\right) - \min_{\mathbf{x}} F(\mathbf{x}) \leq \Theta\left(\frac{1}{T} + \frac{\sigma}{\sqrt{T}}\right) \tag{3.2}$$

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 factors to ensure for SGD:

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:

$$\mathbf{g}_t = \mathbf{g}_t^{(full)} := \mathbf{a}_{\pi(t)}(\mathbf{a}_{\pi(t)}^\top \mathbf{x} - b_{\pi(t)}). \tag{3.3}$$

($\pi(t)$ 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.

3.1 Stochastic Quantization for Saving Bandwidth

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.

3.2 Double Sampling for Unbiased Stochastic Gradient

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

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

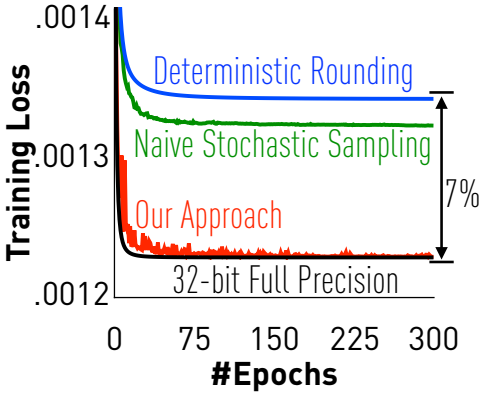
However, *the naive approach does not work* (that is, it does not guarantee convergence), because it is biased:

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

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

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

t



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:

$$\mathbf{g}_t := Q_1(\mathbf{a}_t)(Q_2(\mathbf{a}_t)^\top \mathbf{x} - b_t)\,. \tag{3.4}$$

This gives us an unbiased estimator of the gradient.

3.2.1 Overhead of Storing Two 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, just one additional bit is needed for the second sample because of correlation, as the two samples differ by at most one bit. More generally, since samples are used symmetrically, sending k samples only requires $\log_2 k$ more bits.

3.3 Variance Reduction

From Theorem 3.1, 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 (3.4) can be decomposed into

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

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. Reducing the first term is issue orthogonal to this thesis. 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 3.2 *The stochastic gradient variance using double sampling in* (3.4) $\mathbb{E}\|\mathbf{g}_t - \mathbf{g}_t^{(full)}\|^2$ *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.*

Proof Let $\mathbf{a}$ denote $\mathbf{a}_t$ for short in the followed proof.

$$\begin{aligned}
& \mathbb{E}\left\|Q_1(\mathbf{a})(Q_2(\mathbf{a})^\top\mathbf{x} + b_t) - \mathbf{a}(\mathbf{a}^\top\mathbf{x} + b_t)\right\|^2 \\
\leq\ & 2\mathbb{E}\left\|(Q_1(\mathbf{a}) - \mathbf{a})Q_2(\mathbf{a})^\top\mathbf{x}\right\|^2 + 2\mathbb{E}\left\|\mathbf{a}(Q_2(\mathbf{a}) - \mathbf{a})^\top\mathbf{x})\right\|^2 \\
\leq\ & 2\mathbb{E}_1\|Q_1(\mathbf{a}) - \mathbf{a}\|^2\mathbb{E}_2(Q_2(\mathbf{a})^\top\mathbf{x})^2 + 2\|\mathbf{a}\|^2\mathbb{E}((Q_2(\mathbf{a}) - \mathbf{a})^\top\mathbf{x})^2 \\
\leq\ & 2\mathcal{TV}(\mathbf{a})(2\|\mathbf{a}\|^2\mathbb{E}((Q_2(\mathbf{a}) - \mathbf{a})^\top\mathbf{x})^2 + 2(\mathbf{a}^\top\mathbf{x})^2) + 2\|\mathbf{a}\|^2\mathbb{E}((Q_2(\mathbf{a}) - \mathbf{a})^\top\mathbf{x})^2 \\
\leq\ & \Theta\left(\mathcal{TV}(\mathbf{a})(\mathcal{TV}(\mathbf{a})\|\mathbf{x}\odot\mathbf{x}\| + \|\mathbf{a}^\top\mathbf{x}\|^2 + \|\mathbf{x}\odot\mathbf{x}\|\,\|\mathbf{a}\|^2)\right),
\end{aligned}$$

which completing the proof. □

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

3.3.1 Uniform quantization

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

Lemma 3.3 *[[3]] 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 $\sigma^2_{(full)} \geq \mathbb{E}\|\mathbf{g}_t^{(full)} - \nabla f(\mathbf{x})\|^2$ is the upper bound of using the full stochastic gradient, assuming that $\mathbf{x}$ and all $\mathbf{a}_k$'s are bounded. Because the quantization level s is exponential to the number of bits, 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".

The uniform quantization provides us an intuitive dependence between variance and the level of quantization. However, in practice, we can do better by designing the optimal quantization separation given the quantization level s. This motivates our study in the next chapter.

Chapter 4

Optimal Quantization Strategy for Reducing Variance

In the previous chapter, 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 k intervals. Formally:

$$\begin{aligned} \min_{\mathcal{I}:|\mathcal{I}|=s} \quad & \mathcal{MV}(\mathcal{I}) := \frac{1}{N}\sum_{j=1}^{k}\sum_{x_i \in I_j} \mathrm{err}(x_i, I_j) \\ \text{s.t.} \quad & \bigcup_{j=1}^{s} I_j = [0,1], \quad I_j \cap l_k = \emptyset \text{ for } k \neq j, \end{aligned} \tag{4.1}$$

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.

4.1 Dynamic Programming

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

One challenge is due to non-convexity and non-smoothness. We start from the observation that there exists an optimal solution that places endpoints at input points.

Lemma 4.1 *There is a $\mathcal{I}^*$ so that all endpoints of any $I \in \mathcal{I}^*$ are in $\Omega \cup \{0,1\}$.*

Proof Fix any endpoint b of intervals in $\mathcal{I}^*$. WLOG assumes that $b \neq 0,1$. Then we must have $I = [a,b]$ and $I' = [b,c]$ for some $I, I' \in \mathcal{I}^*$. Observe that the choice of b only affects the error for points in $I \cup I'$. We have that $\mathsf{err}(\Omega, I) + \mathsf{err}(\Omega, I')$ is given by

$$\begin{aligned}&\sum_{x\in I}(b-x)(x-a) + \sum_{x\in I'}(c-x)(x-b)\\ &= Ab + C\,,\end{aligned}$$

where A, C are constants that do not depend on b. Hence, this is a linear objective in b. Since b can freely range between the rightmost point in I and the leftmost point in I', there is an optimizer for this solution at one of those two points. Hence we may choose $b \in \Omega$. □

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 into $[d_j, d_m]$. The complexity of calculating the matrix $V(\cdot,\cdot)$ is $O(N^2+N)$ and the complexity of calculating matrix $T(\cdot,\cdot)$ is $O(kN^2)$. The memory cost is $O(kN+N^2)$.

4.2 Heuristics

The exact algorithm has a complexity that is quadratic to the number of data points. 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.

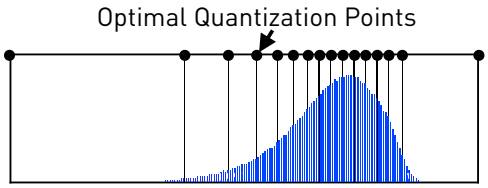


Figure 4.1: Optimal quantization points calculated with dynamic programming given a data distribution.

4.2.1 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 bounded the quality of this approximation.

Theorem 4.2 *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* (4.1) *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*

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

Proof Let p_0^* be 0 and $p_K^* = 1$.We quantitize $\{p_k^*\}_{k=1}^{K-1}$ one element by one element, while monitor the changing of the total variance $N \times \mathcal{MV}(\cdot)$. We first quantize p_1^* to the closest value (denoted it by $Q(p_1^*)$) in $\{d_m\}_{m=1}^{M-1} \cup \{p_0^*, p_K^*\}$ under the monotonicity constraint, that is, $p_0^* \le Q(p_1^*) \le p_2^*$. Here, one important observation is $|p_1^* - Q(p_1^*)| \le a/M$. Consider the total variance of this new solution $Q(p_1^*), p_2^*, \cdots, p_{K-1}^*$. The variance of points falling into the range $[p_2^*, 1]$ does not change at all. Without the loss of generality, assume $p_1^* \ge Q(p_1^*)$.

Next we consider points falling into the following three sets $C_1 = [p_0^*, Q(p_1^*)]$, $C_2 = [Q(p_1^*), p_1^*]$, and $C_3 = [p_1^*, p_2^*]$. The variance of points of falling into C_1 gets reduced from the form of variance in (4). Next we only need to check the variance change for points in C_2 and C_3. Consider C_2 first. The variance for point x in C_2 is

$$(x - Q(p_1^*))(p_1^* - x) \le \frac{a^2}{4M^2}.$$

Thus, the change of variance for points in C_2 would be bounded by $\frac{a^2}{4M^2}$. Then consider C_3. The change of variance for point x in C_3 is

$$\begin{aligned}&(x - Q(p_1^*))(p_2^* - x) - (x - p_1^*)(p_2^* - x)\\ =&(p_1^* - Q(p_1^*))(p_2^* - x)\\ \leq&\frac{a}{M}(p_2^* - x)\end{aligned}$$

Therefore, the change of total variance from $\{p_1^*, p_2^*, \cdots, p_{K-1}^*\}$ to $\{Q(p_1^*), p_2^*, \cdots, p_{K-1}^*\}$ is bounded by

$$\begin{aligned}&\sum_{x\in C_2}\frac{a^2}{4M^2} + \sum_{x\in C_3}\frac{a}{M}(p_2^* - x)\\ \leq&\frac{Nb}{M}\frac{a^2}{4M^2} + \frac{a}{M}\frac{Nb}{M}\sum_{t=1}^{cM/K} t\frac{a}{M}\\ \leq&\frac{a^2bN}{4M^3} + \frac{a^2bc^2N}{MK^2}\end{aligned} \tag{4.2}$$

Similarly, we quantitize p_2^2 in $\{Q(p_1^*), p_2^*, \cdots, p_{K-1}^*\}$ to get a new solution $\{Q(p_1^*), Q(p_2^*), \cdots, p_{K-1}^*\}$ while maintain the monotonicity. We can establish the same upper bound to (4.2). Following this idea, we can obtain a quantization solution $\{Q(p_1^*), Q(p_2^*), \cdots, Q(p_{K-1}^*)\}$. Therefore, we obtain that

$$\begin{aligned}&\mathcal{MV}(Q(p_1^*), Q(p_2^*), \cdots, Q(p_{K-1}^*)) - \mathcal{MV}(p_1^*, \cdots, p_{K-1}^*)\\ &\quad\leq \frac{a^2bK}{4M^3} + \frac{a^2bc^2}{MK}.\end{aligned}$$

Using the fact that $\mathcal{MV}(p_1^*, \cdots, p_{K-1}^*)$ is smaller than $\mathcal{MV}(Q(p_1^*), Q(p_2^*), \cdots, Q(p_{K-1}^*))$ proves the claim. □

Theorem 4.2 suggests that the mean variance using the discrete variance-optimal quantization will converge to the optimal with the rate $O(1/Mk)$.

4.2.2 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 4.1 illustrates an example output for our algorithm.

4.3 Extension to Deep Learning

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

4.3.1 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.

4.3.2 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* [33], which is similar to our approach when the data distribution is "log uniform." However, our approach does not rely on any specific assumption of the data distribution. [18] use k-means to compress the model for *inference* —k-means optimizes for a similar, but different, objective function than ours. In this thesis, we develop a dynamic programming algorithm to do optimal stochastic quantization efficiently.

Chapter 5

Non-Linear Models

In this chapter, we extend our framework to approximate arbitrary classification losses within arbitrarily small bias.

5.1 Quantizing Polynomials

Given a degree d polynomial $P(x) = \sum_{i=0}^{d} m_i z^i$, our goal is to evaluate at $\mathbf{a}^\top \mathbf{x}$, while quantizing $\mathbf{a}$, so as to preserve the value of $P(\mathbf{a}^\top \mathbf{x})$ in expectation.

We will use d independent quantizations of $\mathbf{a}$, $Q_1(\mathbf{a}), Q_2(\mathbf{a}), \ldots, Q_d(\mathbf{a})$. Given these quantizations, our reconstruction of the polynomial at $(\mathbf{a}^\top \mathbf{x})$ will be

$$Q(P) := \sum_{i=0}^{d} m_i \prod_{j \le i} Q_j(\mathbf{a})^\top \mathbf{x}.$$

The fact that this is an unbiased estimator of $P(\mathbf{a}^\top \mathbf{x})$ follows from the independence of the quantizations. Using Lemma 3.3 yields:

Lemma 5.1 $\mathbb{E}[Q(P)^2] \le \left(\sum_{i=0}^{d} m_i r(s)^i (\mathbf{a}^\top \mathbf{x})^i\right)^2$.

5.2 Quantizing Smooth Classification Losses

We now examine a standard classification setting, where samples $[(\mathbf{a}_i, b_i)]_i$ are drawn from a distribution $\mathcal{D}$. Given a smooth loss function $\ell : \mathbb{R} \to \mathbb{R}$, we wish to find $\mathbf{x}$ which minimizes $\mathbb{E}_{\mathcal{D}}[\ell(b \cdot \mathbf{a}^\top \mathbf{x})]$. The gradient of ℓ is given by

$$\nabla_{\mathbf{x}}(b \cdot \mathbf{a}^\top \mathbf{x}) = b\ell'(b \cdot \mathbf{a}^\top \mathbf{x})\mathbf{a}.$$

Assume normalized samples, i.e. $\|\mathbf{a}_i\|_2 \le 1, \forall i$, and that $\mathbf{x}$ is constrained such that $\|\mathbf{x}\|_2 \le R$, for some real value $R > 0$. We wish to approximate the gradient within some target accuracy ε.

To achieve this, fix a minimal-degree polynomial P such that $|P(z) - \ell'(z)| \leq \varepsilon, \forall z \leq R$. Assume this polynomial is known to both transmitter (sample source) and receiver (computing device). The protocol is as follows.

- For a given sample $(\mathbf{a}_i, b_i)$ to be quantized, the source will transmit b_i, as well as $d+1$ independent quantizations $Q_1, Q_2, \ldots, Q_{d+1}$ of $\mathbf{a}_i$.
- The receiver computes $b \cdot Q(P)Q_{d+1}(\mathbf{a}_i)$ and uses it as the gradient.

It is easy to see that the bias in each step is bounded by ε. We can extend Lemma 5.1 to obtain a general guarantee on convergence.

Lemma 5.2 *For any $\varepsilon > 0$ and any convex classification loss function $\ell : \mathbb{R} \to \mathbb{R}$, there exists a polynomial degree $D(\varepsilon, \ell)$ such that the polynomial approximation framework converges to within ε of OPT.*

5.2.1 Chebyshev Approximations

For *logistic loss*, with sigmoid gradient, we notice that polynomial approximations have been well studied. In particular, we use the Chebyshev polynomial approximation of [45].

5.3 Quantizing Non-Smooth Classification Losses

Our techniques further extend to convex loss functions with non-smooth gradients. For simplicity, in the following we focus on SVM, whose gradient (the step function), is discontinuous. This gradient is hard to approximate generally by polynomials; yet, the problem is approachable on intervals of the type $[-R, R] \setminus [-\delta, \delta]$, for some small parameter $\delta > 0$ [12, 4]; the latter reference provides the optimal approximation via Chebyshev polynomials, which we use in our experiments.

The key challenge is that these results do not provide any non-trivial guarantees for our setting, since gradients within the interval $[-\delta, \delta]$ can differ from the true gradient by $\Omega(1)$ in expectation. In particular, due to quantization, the gradient might be *flipped*: its relative value with respect to 0 changes, which corresponds to having the *wrong* label for the current sample.[1] We show two approaches for controlling the error resulting from these errors.

The first is to just ignore such errors: under generative assumptions on the data, we can prove that quantization does not induce significant error. In particular, the error vanishes by taking more data points. The second approach is more general: we use ideas from dimensionality reduction, specifically, low randomness Johnson-Lindenstrauss projections, to detect (with high probability)

[1] Training SVM with noisy labels has been previously considered, e.g. [34], but in a setting where labels are corrupted uniformly at random. It is not hard to see that label corruptions are not uniform random in this case.

if our gradient could be flipped. If so, we refetch the full precision data points. This approach is always correct; however, it requires more communication.

5.3.1 Practical Considerations

The above strategy introduces a precision-variance trade-off, since increasing the precision of approximation (higher polynomial degree) also increases the variance of the gradient. Fortunately, we can reduce the variance and increase the approximation quality by increasing the density of the quantization. In practice, a total of 8 bits per sample is sufficient to ensure convergence for both hinge and logistic loss.

5.3.2 The Refetching Heuristic

The second theoretical approach inspires the following heuristic. Consider hinge loss, i.e. $\sum_{k=1}^{K} \max(0, 1 - b_k \mathbf{a}_k^\top \mathbf{x})$. We first transmit a single low-precision version of $\mathbf{a}_k$, and calculate upper and lower bounds on $b_k \mathbf{a}_k^\top \mathbf{x}$ at the receiver. If the sign of $1 - b_k \mathbf{a}_k^\top \mathbf{x}$ cannot change because of quantization, then we apply the approximate gradient. If the sign could change, then we *refetch* the data at full precision. In practice, this works for 8-bit while only refetching $< 10\%$ of the data.

Chapter 6

Experiments

We provide empirical validation of our framework.

Experimental Setup Table 6.1 shows the datasets we use. Unless otherwise noted, we always use diminishing stepsizes 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.)

6.1 Linear Models

For 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.

6.1.1 Convergence

Figure 6.1 and 6.2 illustrates the result of training linear models: linear regression and least squares SVMs, respectively, 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, on all our datasets, 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

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 6.1: Dataset statistics

of input samples that we need to read, we could potentially save 6–8× memory bandwidth compared to using 32-bit.

We also see from Figure 6.1 (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.

6.1.2 Speedup

We implemented our low-precision framework on a state-of-the-art FPGA platform. For detailed FPGA implementation, please refer to [20]. This implementation assumes the input data is already quantized and stored in memory (data can be quantized during the first epoch).

Figure 6.3 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× 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× memory-bandwidth, which explains the speedup.

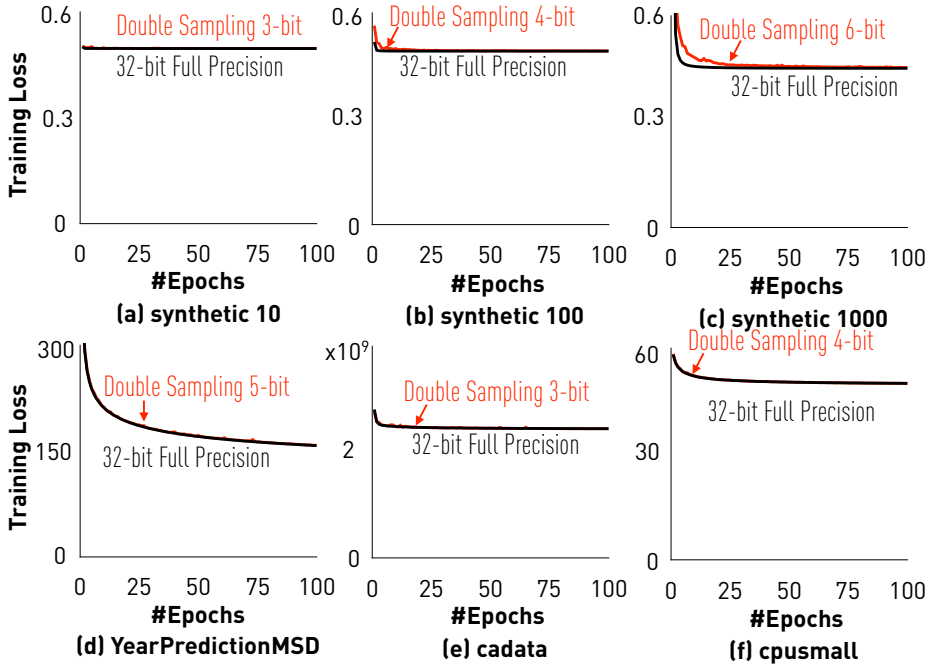


Figure 6.1: Linear regression with *end-to-end quantization* on multiple datasets

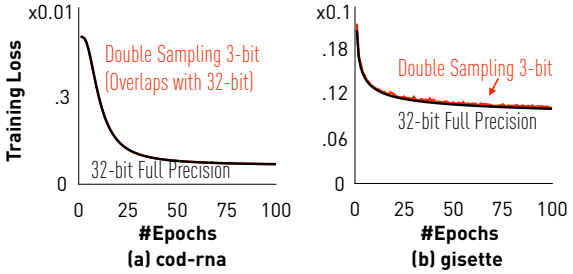


Figure 6.2: Least squares SVM with *end-to-end quantization* on multiple datasets

6.2 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 6.4(a) illustrates the result of using 3-bit and 5-bit for uniform quantization and optimal quantization on the **YearPrediction** dataset. We see that, while uniform quantization needs 5-bit to converge

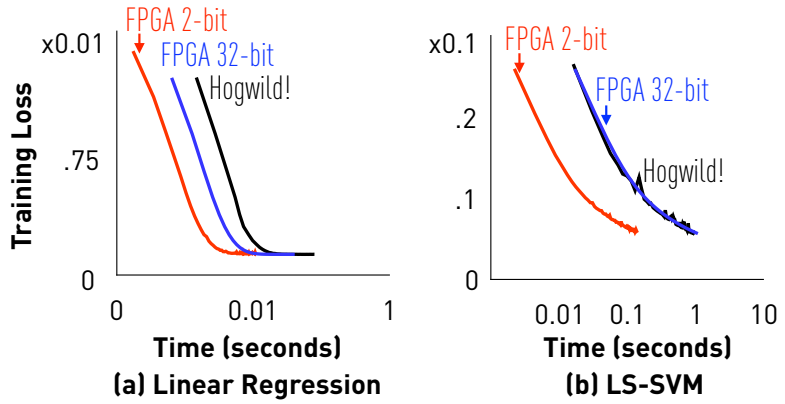


Figure 6.3: FPGA implementation of linear models.

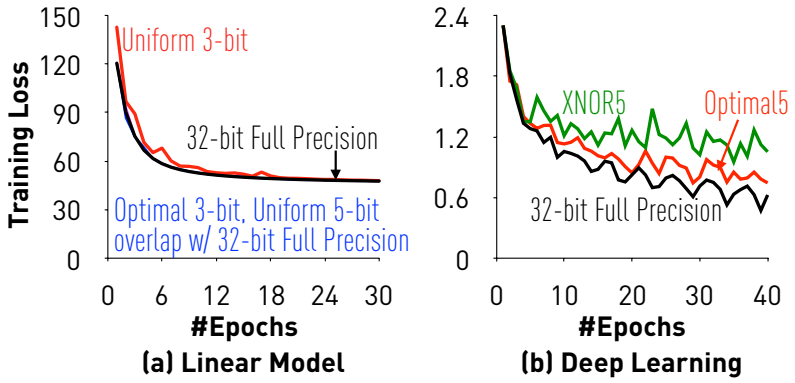


Figure 6.4: Optimal quantization strategy.

smoothly, optimal quantization only needs 3-bit. We save almost 1.7× number of bits by just allocating quantization points carefully.

We validate that, with our data-optimal quantization strategy, we can significantly increase the convergence speed when using the same number of bits.

Figure 6.5 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 higher convergence speed with the same number of bits; or (2) use less bits while getting the same convergence speed.

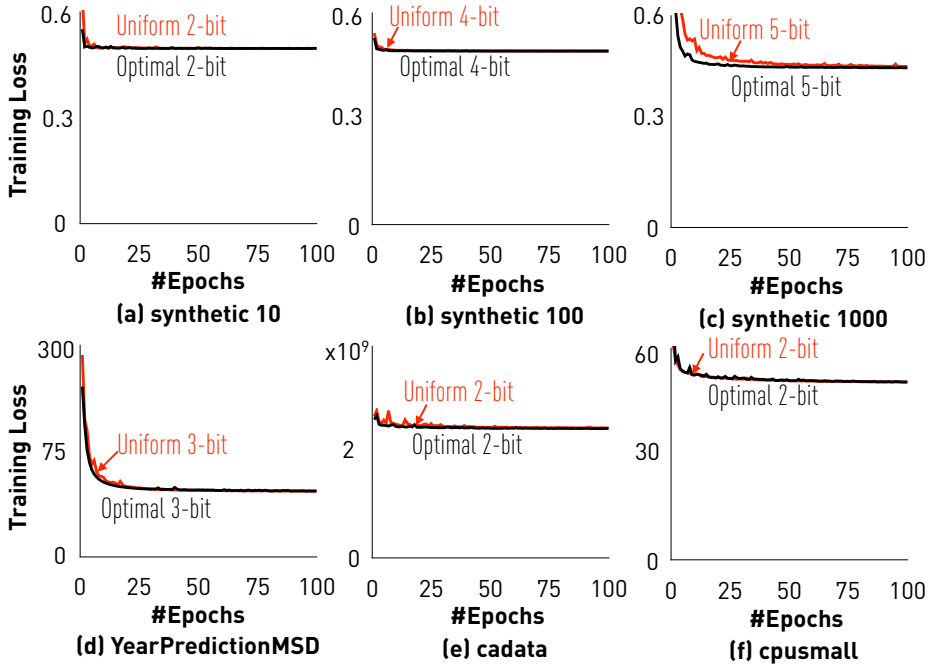


Figure 6.5: optimal quantization vs uniform quantization

6.3 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 [1] 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 6.4(b), Optimal5 converges to a significantly lower training 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.

6.4 Non-Linear Models

We validate that (1) our Chebyshev approximation approach is able to converge to almost the same solution with 8-bit precision for both SVM and logistic regression; (2) our refetching strategy is able to converge to the same solution with 8-bit precision while only refetching $< 10\%$ of the data.; and (3) we are nevertheless able to construct a straw man with 8-bit determin-

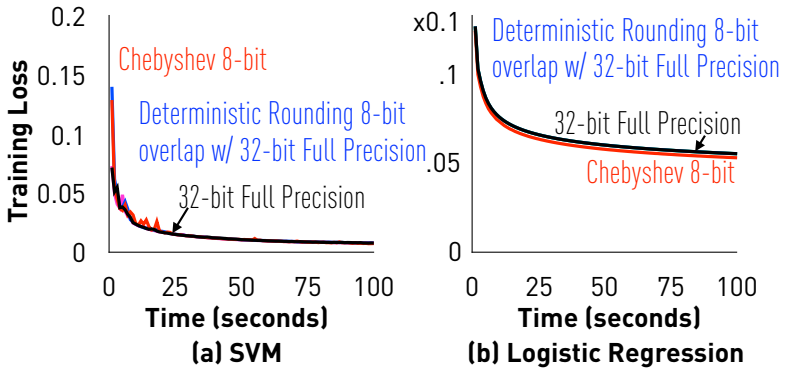


Figure 6.6: Non-linear models with Chebyshev approximation.

istic rounding or naive stochastic rounding to achieve the same quality and convergence rate.

6.4.1 Chebyshev Approximations

Figure 6.6 illustrates the result of training SVM and logistic regression with Chebyshev approximation. Here, we use Chebyshev polynomials up to degree 15 (which requires 16 samples that can be encoded with 4 extra bits). For each sample, the precision is 4-bit, and therefore, in total we use 8-bit for each single number in input samples. We see that, with our quantization framework, SGD converges to similar training loss with a comparable empirical convergence rate for both SVM and logistic regression. We also experience no loss in test accuracy.

6.4.2 Refetching

Figure 6.7 illustrates the result of training SVM with refetching heuristic. We see that, with our refetching heuristic, SGD converges to similar training loss with a comparable empirical convergence rate for SVM. If we increase the number of bits we use, we need to refetch less data and if we use 8-bit quantization, we only need to fetch about 6% of data. We also experience no loss in test accuracy.

6.4.3 Negative Results

We are able to construct the following, much simpler strategy that also uses 8-bit to achieve the same quality and convergence rate as our Chebyshev. In practice, as both strategies incur bias on the result, we do *not* see strong reasons to use our Chebyshev approximation, thus we view this as a negative result. As shown in Figure 6.6, if we simply round the input samples to the nearest 8-bit fix point representation (or do rounding stochastically), we

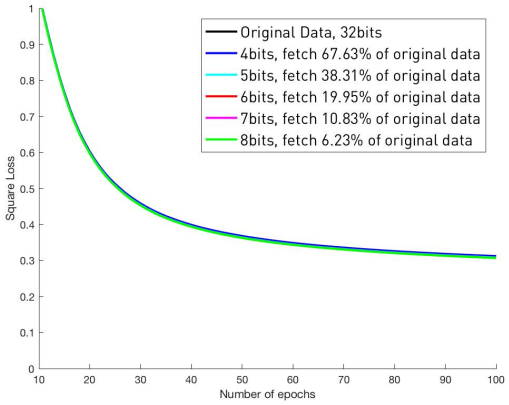


Figure 6.7: SVM with low precision data and refetching on cod-rna dataset

achieve the same, and sometimes better, convergence than our Chebyshev approximation.

Chapter 7

Related Work

There has been significant work on "low-precision SGD" [11, 3]. 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.

7.1 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 [40] 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 [19, 36, 50, 33, 27, 16]. They quantize weights, activations, and gradients to low precision (e.g., 1-bit) and revise the backpropagation algorithm to be aware of the quantization function. The empirical success of this work 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 [44, 13, 18, 28, 22, 23, 46]. In this paper, we focus on training and leave the study of inference for future work.

7.2 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, [15] 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 [17] 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 [9].

7.3 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 [21]. These results are mostly empirical; we aim at providing a theoretical understanding, which enables new algorithms.

Chapter 8

Discussion

Our motivating question was whether end-to-end low-precision data representation can enable efficient computation with convergence guarantees. We show that 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.

For non-linear models, the picture is more nuanced. In particular, we find that our framework can be generalized to this setting, and that in practice *8-bit is sufficient* to achieve good accuracy on a variety of tasks, such as SVM and logistic regression. However, in this generalized setting, naive rounding has similar performance on many practical tasks.

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.

Appendix A

Full GAN test output

We present the full evaluation results for all parameter sets in the space of FWHM=$[1.4, 1.8, 2.0, 2.5\,\text{arcsec}]$ and $\sigma_{\text{new}} = [1.0, 1.2, 2.0, 5.0, 10.0]\sigma_{\text{original}}$ in Figure A.1. We show an example in Figure 2.3 and provide the full set, including the 10x cross validation, is available at `http://space.ml/supp/GalaxyGAN.html`.

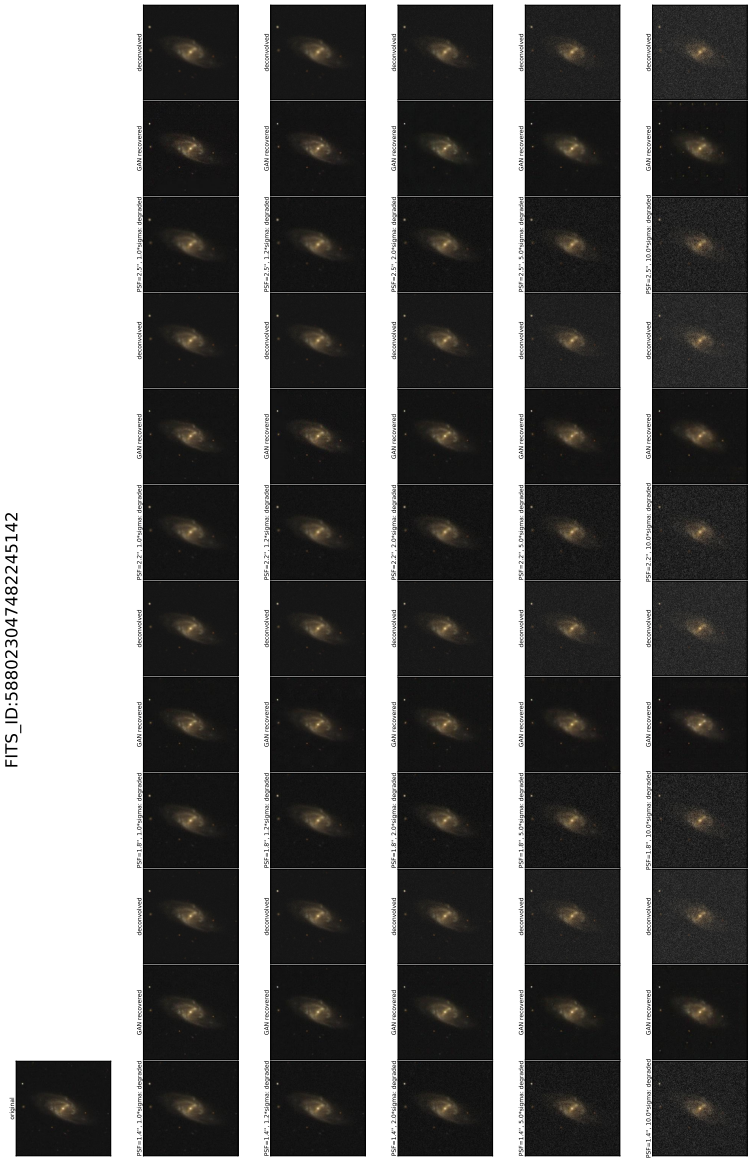


Figure A.1: Full outputs for the entire test grid of PSF FWHM and noise levels, for one object. We only show the first object, the remaining outputs are included as online-only. The full set, including the 10x cross validation, is available at `http://space.ml/supp/GalaxyGAN.html`

Bibliography

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

[2] S. Alam, F. D. Albareti, C. Allende Prieto, F. Anders, S. F. Anderson, T. Anderton, B. H. Andrews, E. Armengaud, É. Aubourg, S. Bailey, and et al. The Eleventh and Twelfth Data Releases of the Sloan Digital Sky Survey: Final Data from SDSS-III. *The Astrophysical Journal Supplement Series*, 219:12, July 2015.

[3] Dan Alistarh, Jerry Li, Ryota Tomioka, and Milan Vojnovic. QSGD: Randomized Quantization for Communication-Optimal Stochastic Gradient Descent. *ArXiv e-prints*, October 2016.

[4] Zeyuan Allen-Zhu and Yuanzhi Li. Faster principal component regression via optimal polynomial approximation to sgn (x). *arXiv:1608.04773*, 2016.

[5] Anthony J Bell and Terrence J Sejnowski. An information-maximization approach to blind separation and blind deconvolution. *Neural computation*, 7(6):1129–1159, 1995.

[6] Sébastien Bubeck. Convex optimization: Algorithms and complexity. *Foundations and Trends® in Machine Learning*, 8(3-4):231–357, 2015.

[7] R. Buta and F. Combes. Galactic Rings. *Fundamentals of Cosmic Physics*, 17:95–281, 1996.

[8] N. Cantale, F. Courbin, M. Tewes, P. Jablonka, and G. Meylan. Firedec: a two-channel finite-resolution image deconvolution algorithm. *Astronomy and Astrophysics*, 589:A81, May 2016.

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

[10] F. Courbin. *Deconvolution et combinaison optimale d'images astronomiques: application au cas des mirages gravitationnels*. PhD thesis, Institut d'astrophysique, Universite de Liege, Belgium; Observatoire de Paris Meudon - DAEC, FRance, 1999.

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

[12] Roy Frostig, Cameron Musco, Christopher Musco, and Aaron Sidford. Principal component projection without principal component analysis. *arXiv:1602.06872*, 2016.

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

[14] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *NIPS*, pages 2672–2680, 2014.

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

[16] Suyog Gupta, Ankur Agrawal, Kailash Gopalakrishnan, and Pritish Narayanan. Deep learning with limited numerical precision. In *ICML*, pages 1737–1746, 2015.

[17] Daniel B. Hall. Measurement error in nonlinear models: A modern perspective (2nd ed.). *Journal of the American Statistical Association*, 103:427–427, 2008.

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

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

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

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

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

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

[24] R. Laureijs, J. Amiaux, S. Arduini, J. . Augueres, J. Brinchmann, R. Cole, M. Cropper, C. Dabin, L. Duvet, A. Ealet, and et al. Euclid Definition Study Report. *ArXiv e-prints*, October 2011.

[25] G. Letawe, P. Magain, F. Courbin, P. Jablonka, K. Jahnke, G. Meylan, and L. Wisotzki. On-axis spectroscopy of the host galaxies of 20 optically luminous quasars at z ˜ 0.3. *Monthly Notices of the Royal Astronomical Society*, 378:83–108, June 2007.

[26] Y. Letawe, P. Magain, G. Letawe, F. Courbin, and D. Hutsemékers. Study of the QSO HE0354-5500 with combined HST imaging and VLT spectroscopy . An example of a deconvolution-based method for probing the QSOs host galaxies characteristics. *Memoire della Societa Astronomica Italiana*, 79:1251, 2008.

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

[28] Darryl Lin, Sachin Talathi, and Sreekanth Annapureddy. Fixed point quantization of deep convolutional networks. In *ICML*, pages 2849–2858, 2016.

[29] LSST Science Collaboration, P. A. Abell, J. Allison, S. F. Anderson, J. R. Andrew, J. R. P. Angel, L. Armus, D. Arnett, S. J. Asztalos, T. S. Axelrod, and et al. LSST Science Book, Version 2.0. *ArXiv e-prints*, December 2009.

[30] Leon B Lucy. An iterative technique for the rectification of observed distributions. *The astronomical journal*, 79:745, 1974.

[31] P. Magain, F. Courbin, M. Gillon, S. Sohy, G. Letawe, V. Chantry, and Y. Letawe. A deconvolution-based algorithm for crowded field photometry with unknown point spread function. *Astronomy and Astrophysics*, 461:373–379, January 2007.

[32] P. Magain, F. Courbin, and S. Sohy. Deconvolution with Correct Sampling. *The Astrophysical Journal Letters*, 494:472–477, February 1998.

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

[34] Nagarajan Natarajan, Inderjit S Dhillon, Pradeep K Ravikumar, and Ambuj Tewari. Learning with noisy labels. In *NIPS*. 2013.

[35] Harry Nyquist. Certain topics in telegraph transmission theory. 1928.

[36] Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi. Xnor-net: Imagenet classification using binary convolutional neural networks. In *ECCV*, pages 525–542. Springer, 2016.

[37] S. Reed, Z. Akata, X. Yan, L. Logeswaran, B. Schiele, and H. Lee. Generative Adversarial Text to Image Synthesis. *ArXiv e-prints*, May 2016.

[38] William Hadley Richardson. Bayesian-based iterative method of image restoration. *JOSA*, 62(1):55–59, 1972.

[39] Kevin Schawinski, Ce Zhang, Hantian Zhang, Lucas Fowler, and Gokula Krishnan Santhanam. Generative adversarial networks recover features in astrophysical images of galaxies beyond the deconvolution limit. *Monthly Notices of the Royal Astronomical Society: Letters*, 467(1):L110–L114, 2017.

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

[41] C. E. Shannon. Communication in the presence of noise. *Proceedings of the IRE*, 37(1):10–21, Jan 1949.

[42] J. Sola and J. Sevilla. Importance of input data normalization for the application of neural networks to complex industrial problems. *IEEE Transactions on Nuclear Science*, 44(3):1464–1468, Jun 1997.

[43] J. L. Starck, E. Pantin, and F. Murtagh. Deconvolution in Astronomy: A Review. *Publications of the Astronomical Society of the Pacific*, 114:1051–1069, October 2002.

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

[45] Miroslav Vlcek. Chebyshev polynomial approximation for activation sigmoid function. *Neural Network World*, 22(4):387, 2012.

[46] Jiaxiang Wu, Cong Leng, Yuhang Wang, Qinghao Hu, and Jian Cheng. Quantized convolutional neural networks for mobile devices. In *CVPR*, pages 4820–4828, 2016.

[47] Li Xu, Jimmy S. Ren, Ce Liu, and Jiaya Jia. Deep convolutional neural network for image deconvolution. *NIPS*, pages 1790–1798, 2014.

[48] D. G. York, J. Adelman, J. E. Anderson, Jr., S. F. Anderson, J. Annis, N. A. Bahcall, J. A. Bakken, R. Barkhouser, S. Bastian, E. Berman, W. N. Boroski, S. Bracker, C. Briegel, J. W. Briggs, J. Brinkmann, R. Brunner, S. Burles, L. Carey, M. A. Carr, F. J. Castander, B. Chen, P. L. Colestock, A. J. Connolly, J. H. Crocker, I. Csabai, P. C. Czarapata, J. E. Davis, M. Doi, T. Dombeck, D. Eisenstein, N. Ellman, B. R. Elms, M. L. Evans, X. Fan, G. R. Federwitz, L. Fiscelli, S. Friedman, J. A. Frieman, M. Fukugita, B. Gillespie, J. E. Gunn, V. K. Gurbani, E. de Haas, M. Haldeman, F. H. Harris, J. Hayes, T. M. Heckman, G. S. Hennessy, R. B. Hindsley, S. Holm, D. J. Holmgren, C.-h. Huang, C. Hull, D. Husby, S.-I. Ichikawa, T. Ichikawa, Ž. Ivezić, S. Kent, R. S. J. Kim, E. Kinney, M. Klaene, A. N. Kleinman, S. Kleinman, G. R. Knapp, J. Korienek, R. G. Kron, P. Z. Kunszt, D. Q. Lamb, B. Lee, R. F. Leger, S. Limmongkol, C. Lindenmeyer, D. C. Long, C. Loomis, J. Loveday, R. Lucinio, R. H. Lupton, B. MacKinnon, E. J. Mannery, P. M. Mantsch, B. Margon, P. McGehee, T. A. McKay, A. Meiksin, A. Merelli, D. G. Monet, J. A. Munn, V. K. Narayanan, T. Nash, E. Neilsen, R. Neswold, H. J. Newberg, R. C. Nichol, T. Nicinski, M. Nonino, N. Okada, S. Okamura, J. P. Ostriker, R. Owen, A. G. Pauls, J. Peoples, R. L. Peterson, D. Petravick, J. R. Pier, A. Pope, R. Pordes, A. Prosapio, R. Rechenmacher, T. R. Quinn, G. T. Richards, M. W. Richmond, C. H. Rivetta, C. M. Rockosi, K. Ruthmansdorfer, D. Sandford, D. J. Schlegel, D. P. Schneider, M. Sekiguchi, G. Sergey, K. Shimasaku, W. A. Siegmund, S. Smee, J. A. Smith, S. Snedden, R. Stone, C. Stoughton, M. A. Strauss, C. Stubbs, M. SubbaRao, A. S. Szalay, I. Szapudi, G. P. Szokoly, A. R. Thakar, C. Tremonti, D. L. Tucker, A. Uomoto, D. Vanden Berk, M. S. Vogeley, P. Waddell, S.-i. Wang, M. Watanabe, D. H. Weinberg, B. Yanny, N. Yasuda, and SDSS Collaboration. The Sloan Digital Sky Survey: Technical Summary. *The Astronomical Journal*, 120:1579–1587, September 2000.

[49] Hantian Zhang, Jerry Li, Kaan Kara, Dan Alistarh, Ji Liu, and Ce Zhang. The ZipML Framework for Training Models with End-to-End Low Precision: The Cans, the Cannots, and a Little Bit of Deep Learning. *ArXiv e-prints*, November 2016.

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

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Declaration of originality

The signed declaration of originality is a component of every semester paper, Bachelor's thesis, Master's thesis and any other degree paper undertaken during the course of studies, including the respective electronic versions.

Lecturers may also require a declaration of originality for other written papers compiled for their courses.

I hereby confirm that I am the sole author of the written work here enclosed and that I have compiled it in my own words. Parts excepted are corrections of form and content by the supervisor.

Title of work (in block letters):

The ZipML framework for Training Models with End-to-End Low Precision

Authored by (in block letters):

For papers written by groups the names of all authors are required.

Name(s):	**First name(s):**
ZHANG	HANTIAN

With my signature I confirm that

- I have committed none of the forms of plagiarism described in the 'Citation etiquette' information sheet.
- I have documented all methods, data and processes truthfully.
- I have not manipulated any data.
- I have mentioned all persons who were significant facilitators of the work.

I am aware that the work may be screened electronically for plagiarism.

Place, date	**Signature(s)**
Zurich, 19.03.2017	

For papers written by groups the names of all authors are required. Their signatures collectively guarantee the entire content of the written paper.