FINITE PRECISION DEEP LEARNING WITH THEORETICAL GUARANTEES

BY

CHARBEL SAKR

DISSERTATION

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Electrical and Computer Engineering
in the Graduate College of the
University of Illinois Urbana-Champaign, 2021

Urbana, Illinois

Doctoral Committee:

Professor Naresh Shanbhag, Chair
Professor Sarita Adve
Professor Minh Do
Dr. Kailash Gopalakrishnan, IBM

ABSTRACT

Recent successes of deep learning have been achieved at the expense of a very high computational and parameter complexity. Today, deployment of both inference and training of deep neural networks (DNNs) is predominantly in the cloud. A recent alternative trend is to deploy DNNs onto untethered, resource-constrained platforms at the Edge. To realize on-device intelligence, the gap between algorithmic requirements and available resources needs to be closed. One popular way of doing so is via implementation in finite precision.

While ad-hoc trial and error techniques in finite precision deep learning abound, theoretical guarantees on network accuracy are elusive. The work presented in this dissertation builds a theoretical framework for the implementation of deep learning in finite precision. For inference, we theoretically analyze the worst-case accuracy drop in the presence of weight and activation quantization. Furthermore, we derive an optimal clipping criterion (OCC) to minimize the precision of dot-product outputs. For implementations using in-memory computing, OCC lowers ADC precision requirements. We analyze fixed-point training and present a methodology for implementing quantized back-propagation with close-to-minimal per-tensor precision. Finally, we study accumulator precision for reduced precision floating-point training using variance analysis techniques.

We first introduce our work on fixed-point inference with accuracy guarantees. Theoretical bounds on the mismatch between limited and full precision networks are derived. Proper precision assignment can be readily obtained employing these bounds, and weight-activation as well as per-layer precision trade-offs are derived. Applied to a variety of networks and datasets, the presented analysis is found to be tight to within 2 bit. Furthermore, it is shown that a minimum precision network can have up to $\sim 3.5\times$ lower hardware complexity than a binarized network at iso-accuracy. In general, a minimum precision network can reduce complexity by up to $\sim 10\times$ compared to a full

precision baseline while maintaining accuracy. Per-layer precision analysis indicates that precision requirements of common networks vary from 2 bit to 10 bit to guarantee an accuracy close to the floating-point baseline.

Then, we study DNN implementation using in-memory computing (IMC), where we propose OCC to minimize the column ADC precision. The signal-to-quantization-noise ratio (SQNR) of OCC is shown to be within 0.8 dB of the well-known optimal Lloyd-Max quantizer. OCC improves the SQNR of the commonly employed full range quantizer by 14 dB which translates to a 3 bit ADC precision reduction. The input-serial weight-parallel (ISWP) IMC architecture is studied. Using bit-slicing techniques, significant energy savings can be achieved with minimal accuracy lost. Indeed, we prove that a dot-product can be realized with a single memory access while suffering no more than 2 dB SQNR drop. Combining the proposed OCC and ISWP noise analysis with our proposed DNN precision analysis, we demonstrate $\sim 6\times$ reduction of energy consumption in DNN implementation at iso-accuracy.

Furthermore, we study the quantization of the back-propagation training algorithm. We propose a systematic methodology to obtain close-to-minimal per-layer precision requirements for guaranteed statistical similarity between fixed-point and floating-point training. The challenges of quantization noise, inter-layer and intra-layer precision trade-offs, dynamic range, and stability are jointly addressed. Applied to several benchmarks, fixed-point training is demonstrated to achieve high fidelity to the baseline with an accuracy drop no greater than 0.56%. The derived precision assignment is shown to be within 1 bit per tensor of the minimum. The methodology is found to reduce representational, computational, and communication costs of training by up to $6\times$, $8\times$, and $4\times$, respectively, compared to the baseline and related works.

Finally, we address the problem of reduced precision floating-point training. In particular, we study accumulation precision requirements. We present the variance retention ratio (VRR), an analytical metric measuring the suitability of accumulation mantissa precision. The analysis expands on concepts employed in variance engineering for weight initialization. An analytical expression for the VRR is derived and used to determine accumulation bit-width for precise tailoring of computation hardware. The VRR also quantifies the benefits of effective summation reduction techniques such as chunked accumulation and sparsification. Experimentally, the validity and tightness of our analysis are verified across multiple deep learning benchmarks.

To my parents, Toufic and Léna, for all their sacrifices.

ACKNOWLEDGMENTS

Virtues are formed in man by his doing the actions.[1]

Doing a PhD certainly feels like a Herculean task; and today, I am certainly not the same man I was when it all started. As I finalize my dissertation, I can say without a doubt that my achievements were facilitated by various sources of help I was fortunate to enjoy over the past six years.

First of all, the mentorship I have received from my advisor, Professor Naresh Shanbhag, has been essential in getting me here. Prof. Shanbhag's vision and guidance encouraged me to explore uncharted waters in finite precision deep learning. I am very proud of the numerous contributions we have made in this area which would have been impossible without his help. In addition, working with Prof. Shanbhag has seriously enhanced my skills as a researcher, and for that I am very grateful.

I wish to thank my dissertation committee members: Professor Sarita Ave, Professor Minh Do, and Doctor Kailash Gopalakrishnan. Their feedback has been very insightful and useful in improving the quality of this dissertation.

As a member of Prof. Shanbhag's research group, I had the chance to work with many fine individuals: Ameya Patil, Sujan Gonugondla, Hassan Dbouk, Saion Roy, Han-Mo Ou, Hanfei Geng, Michael Tuttle, Kuk-Hwan Kim, Sai Zhang, Mingu Kang, Sungmin Lim, Yingyan Lin, and Yongjune Kim. Our collaborations and discussions helped broaden my horizons, which I appreciate.

I wish to thank Dr. Gopalakrishnan for the opportunity to work with him and his team at IBM in 2017 and 2018. Getting firsthand exposure to industry research was a tremendous experience and I wish to particularly thank our ICLR 2019 co-authors: Naigang Wang, Chia-Yu Chen, Jungwook Choi, and Ankur Agrawal.

[1] A quote from *Nicomachean Ethics* by Aristotle.

On a more personal level, I wish to thank a few of my friends who I believe had an impact in my completing this PhD, particularly during a pandemic. Thank you to Peter, Marcel, and Ali for being there.

In hindsight, I realize how selfish it is to do a PhD. My family back in Lebanon has been living in very trying circumstances. Perhaps I should have been there for them, but I was not. In fact, I was able to work on my PhD here in Illinois because of my parents' sacrifices. I am so grateful to them and so proud to be their son. Toufic and Léna, thank you from the bottom of my heart; this dissertation is dedicated to you.

Finally, I thank the University of Illinois and the Department of Electrical and Computer Engineering (ECE) for the opportunity to pursue my PhD. Thank you to the ECE department making me the recipient of two fellowships. Thank you to the Coordinated Science Laboratory (CSL) for physically hosting me and my work.

The work in this dissertation was supported in part by the following agencies: Systems On Nanoscale Information fabriCs (SONIC) Center, Center for Brain Inspired Computing (C-BRIC) and IBM-ILLINOIS Center for Cognitive Computing Systems Research (C3SR).

TABLE OF CONTENTS

LIST OF TABLES ix

LIST OF FIGURES x

CHAPTER 1 INTRODUCTION 1
1.1 Related Work 5
1.2 Dissertation Contributions and Organization 10

CHAPTER 2 FIXED-POINT INFERENCE WITH THEORETICAL GUARANTEES 14
2.1 Motivation 14
2.2 Fixed-Point Neural Networks 17
2.3 Bounds on Mismatch Probability 20
2.4 Bound Validation Results 26
2.5 Per-layer Precision Analysis 31
2.6 Numerical Results - Small Models and Datasets 34
2.7 Numerical Results - Large Models and Datasets 38
2.8 Summary 44
2.9 Addendum: Proofs of Bounds on Mismatch Probability 45

CHAPTER 3 MINIMIZING ADC PRECISION FOR INFERENCE ON IN-MEMORY ARCHITECTURE 52
3.1 Motivation 52
3.2 Problem Setup 54
3.3 The Optimal Clipping Criterion 57
3.4 Bit Slicing Analysis 64
3.5 DNN Implementation Methodology 67
3.6 Numerical Results 73
3.7 Summary 77
3.8 Addendum: Proofs of Theoretical Results 77

CHAPTER 4 FIXED-POINT TRAINING WITH CLOSE-TO-MINIMAL PRECISION 82
4.1 Motivation 82
4.2 Problem Setup, Notation, and Metrics 84

4.3 Precision Assignment Methodology and Analysis 86
4.4 Numerical Results 89
4.5 Discussion 95
4.6 Addendum: Proofs and Additional Results 96

CHAPTER 5 FLOATING-POINT TRAINING WITH ACCUMULATION BIT-WIDTH SCALING 116
5.1 Motivation 116
5.2 Background on Floating-Point Arithmetic 119
5.3 Accumulation Variance 120
5.4 Mantissa Precision Requirements Analysis 122
5.5 Numerical Results 128
5.6 Summary 132
5.7 Addendum: Proofs of Theoretical Results 132

CHAPTER 6 CONCLUSION AND FUTURE WORK 139
6.1 Summary of Contributions 139
6.2 Future Research Prospects 140

REFERENCES 143

LIST OF TABLES

2.1 Results for the MNIST dataset 26
2.2 Results for the CIFAR-10 dataset 29

3.1 Values of analog noise parameters in a 65 nm process 56
3.2 Comparison of MSE between OCC and LM 60
3.3 Topological details of the VGG-9 DNN 72
3.4 Achieved accuracy under various implementations 73
3.5 Per-layer ADC precision and bitcell capacitance values for various implementations considered 74

4.1 Complexity and accuracy comparisons for fixed-point training 93
4.2 Feedforward path precisions in the CIFAR-10 ConvNet 108
4.3 Backward path precisions in the CIFAR-10 ConvNet 109
4.4 Feedforward path precisions in the SVHN ConvNet 110
4.5 Backward path precisions in the SVHN ConvNet 111
4.6 Feedforward path precisions in the CIFAR-10 ResNet 112
4.7 Backward path precisions in the CIFAR-10 ResNet 113
4.8 Feedforward path precisions in the CIFAR-100 ResNet 114
4.9 Backward path precisions in the CIFAR-100 ResNet 115

5.1 Predicted mantissa precision required for accumulations of our considered networks 129

LIST OF FIGURES

1.1 Cloud, Edge, and TinyML applications 2
1.2 Challenges in resource-constrained machine learning 3
1.3 Approaches to reduced complexity machine learning 5
1.4 Recent trends in in-training and post-training quantization . . 7
1.5 Accuracy vs. Complexity of binarized networks and post-training quantization approaches 8
1.6 Difficulty in training with noise 10

2.1 Validity of bounds for the MNIST dataset 25
2.2 Validity of bounds for the CIFAR-10 dataset 29
2.3 Per-layer precision assignments for a feedforward neural network architecture 32
2.4 Plots showing quantization noise gains and per-layer precision assignments for MNIST and CIFAR-10 35
2.5 Test error vs. minimum precision, computational and representational costs for MNIST and CIFAR-10 36
2.6 Precision analysis for ResNet-18 on CIFAR-10 38
2.7 Precision analysis for AlexNet on ImageNet 40
2.8 Precision analysis for VGG-16 on ImageNet 41
2.9 Accuracy-aware complexity minimization for AlexNet and VGG-16 on ImageNet 43

3.1 The input-serial weight-parallel (ISWP) architecture 54
3.2 Illustration of various quantization strategies for a unit Gaussian signal 59
3.3 Comparison of FR, OCC, and LM for output and ADC quantization 62
3.4 Bit slicing gain β 65
3.5 Impact of bit slicing on the accuracy of IMC dot-products . . 66
3.6 Per-layer precision normalized energy per operation E_{OP} for the FS/OCC/NE and BS/FR/NE implementations 76

4.1 Fixed-point training problem setup 84
4.2 Predicted precision configurations for fixed-point training . . . 91
4.3 Convergence curves for fixed-point training 92

4.4 Additional experiments on minimality and sensitivity of fixed-point training 106

5.1 The importance of accumulation precision in floating-point training 117
5.2 The three GEMM calls in a back-propagation iteration 120
5.3 Snapshot of measured weight gradient variance as a function of layer index during floating-point training 122
5.4 Illustration of the difference between full swamping and partial swamping 122
5.5 Normalized variance lost as a function of accumulation length 127
5.6 Convergence curves for floating-point training 130

CHAPTER 1

INTRODUCTION

About a decade ago, the deep neural network (DNN) AlexNet [1] won the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) [2]. It was the only deep learning model presented at the competition in 2012, and it beat its runner up by more than 10 percentage points in the Top-5 error metric. Since then, intensive research interest in deep learning arose and major advances were realized.

In subsequent editions of ILSVRC, all winners of the competitions used deep learning models. Recently, the models have surpassed human level accuracy. First achieving this feat was ResNet [3] in 2017. The scope of deep learning successes even expanded beyond visual data processing. Indeed, DNNs are today the most powerful predictive models in different cognitive tasks such as speech [4, 5] and natural language [6, 7] processing. Deep learning has also helped machines beat human champions in complex games such as Go [8].

Much of the success of DNNs has been achieved at the expense of a very high computational and parameter complexity. For instance, AlexNet requires over 800 million multiply-accumulates (MACs) per image and has 60 million parameters. Google's Deepface [9] requires 500 million MACs/image and involves more than 120 million parameters. In addition, training such models using the stochastic gradient descent (SGD) algorithm is much more compute intensive with 100s of exa-ops and gigabytes of data storage required [10]. Such enormous complexities have forced deployment of both inference and training of DNNs in the cloud (see Figure 1.1) rather than onto untethered, resource-constrained platform at the Edge and tiny devices [11].

While deep learning has thrived thus far in the cloud, some applications require deployment of DNN onto Edge and tinyML [11] devices [12], as depicted in Figure 1.1. Mobile visual data processing is required for real-time

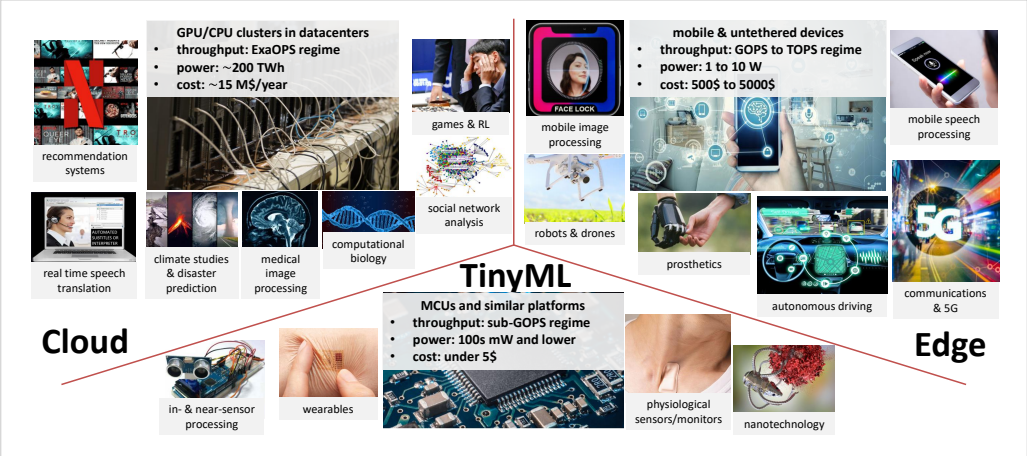


Figure 1.1: Cloud, Edge, and TinyML applications.

face recognition [13]. Similarly, mobile speech processing is required for voice activity detection and recognition [14]. Private and local text analysis and translation necessitates natural language processing at the Edge [15]. Stringent latency constraints require local processing in autonomous driving and systems [16]. Finally, patient-specific physiological data should not be transmitted by biomedical wearables [17].

Without on-device intelligence, the above applications are traditionally implemented using a sensor-cloud-actuator setup. A device captures raw data, transmits it to the cloud, waits for it to be processed, and finally receives an instruction for actuation. Such a scheme has three drawbacks. First, the application level latency is harmed by the wait/idle time in which data is transferred to and from the cloud. Second, the device energy consumption itself is dominated by the cost of the transceiver. Third, the device-cloud data exchange can lead to privacy concerns [18]. In contrast, with on-device intelligence, these bottlenecks are bypassed and decisions are immediately generated in-situ. Depending on the application, the total energy and latency can be reduced by orders of magnitude [19], while preserving privacy. However, several challenges remain in the quest to realize on-device intelligence.

When DNNs are to be deployed onto embedded devices, a large mismatch arises between algorithmic requirements and available resources. Figure 1.2 describes an example of such mismatch in the context of real-time inference for image classification. A typical low complexity state-of-the-art model requires $\sim$ 20 MB storage, $\sim$ 165 GOPS throughput (for real-time inference),

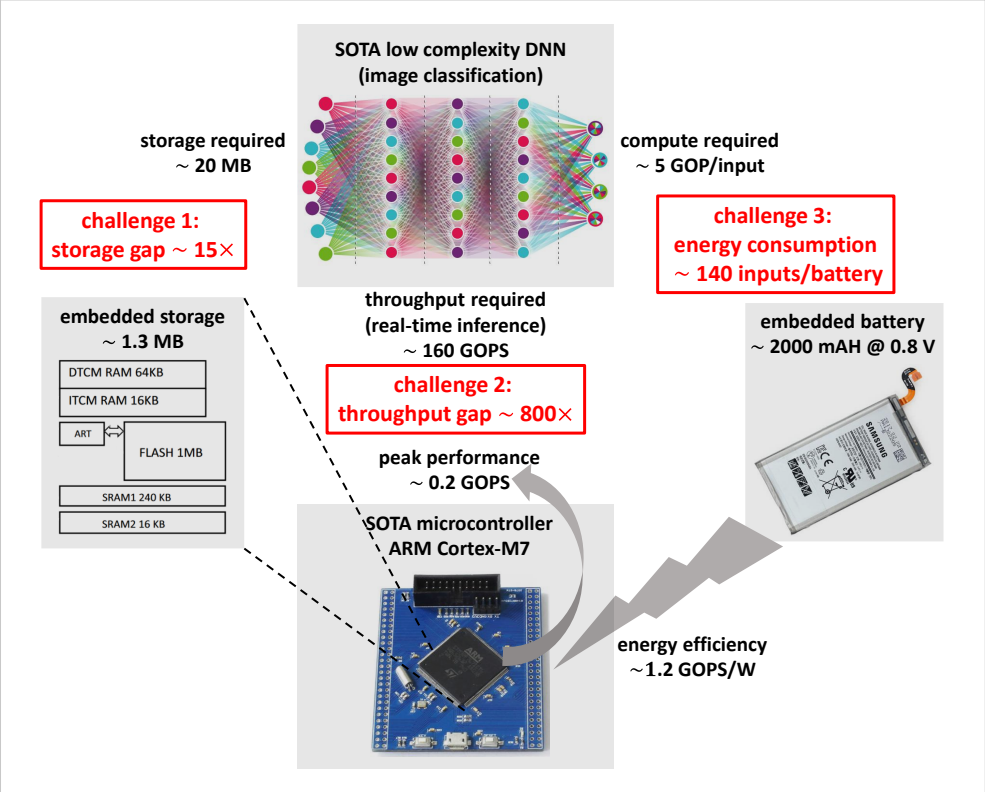


Figure 1.2: Challenges in resource-constrained machine learning.

and $\sim$ 5 GOPS/input compute. We target the implementation of such algorithm onto an ARM Cortex-M7 microcontroller, the state-of-the-art in tinyML applications [20]. This device has an embedded storage of $\sim$ 1.3 MB, a peak performance of $\sim$ 0.2 GOPS, and an energy efficiency of $\sim$ 1.2 GOPS/W. We further assume it to be powered by an embedded battery such the Samsung S8 Battery, rated at $\sim$ 2000 mAH when operated at 0.8 V. Thus, comparing algorithmic requirements to available resources, we find a $\sim 15\times$ storage gap, a $\sim 800\times$ throughput gap, and a capability to process only $\sim$ 140 inputs before the battery runs out.

The above difficulties in Edge deployment of DNNs can be alleviated via the use of accelerator architectures. Such architectures, in contrast to CPUs and GPUs, provide significant benefits in terms of energy and latency. Indeed, accelerators are precision-optimized as per DNN quantization noise tolerance. Furthermore, accelerator architecture is tailored to match the dataflow of algorithms. Such tailoring leads to orders-of-magnitude in energy and latency savings compared to traditional architectures found in CPUs, GPUs, and even MCUs.

Recently, several accelerators have been proposed in the context of both inference and training. An early example is Eyeriss [21], a 16 bit fixed-point inference accelerator. Later, a tensor processing unit was introduced by Google [22] and supported both inference (8 bit fixed-point) and training (16 bit floating-point). IBM released a training accelerator using 16 bit floating-point [23] which was recently upgraded to 4 bit radix-4 floating-point [24]. Intel worked on a neural processing system using a new number format (16 bit flexpoint) [25]. Nvidia demonstrated prototyping of an 8 bit fixed-point inference accelerator using novel synthesis techniques [26]. Beyond digital realizations, analog/mixed-signal circuits have also been employed using in-memory computing. These have typically implemented binary (1 bit) DNNs [27, 28, 29, 30, 31, 14, 32, 33, 34, 35, 36, 37, 38].

The list of accelerators mentioned above is by no means exhaustive; nonetheless, it highlights an important trend. Traditional single (32 bit) and double (64 bit) floating-point precisions found in CPUs and GPUs are never utilized. Justifiably, reduced precision is required in accelerators as it leads to linear and quadratic reduction of storage and compute complexities, respectively.

Rather than optimizing precision in a theoretically sound manner, the above works have relied on trial-and-error techniques. Indeed, extensive simulations on several benchmarks are performed, and if results indicate a suitable empirical accuracy, the corresponding precision is selected for implementation. Such empiricism requires a substantial design effort and does not guarantee optimality, i.e., that minimum DNN precision requirements have been found. This leads to the formulation of two fundamental questions:

1. What are a DNN's minimum precision requirements for a given accuracy level?

2. Can those requirements be determined analytically?

The work presented in this dissertation will answer both questions. The methods proposed benefit acceleration at Edge [12], tinyML [11], and also cloud [23] platforms.

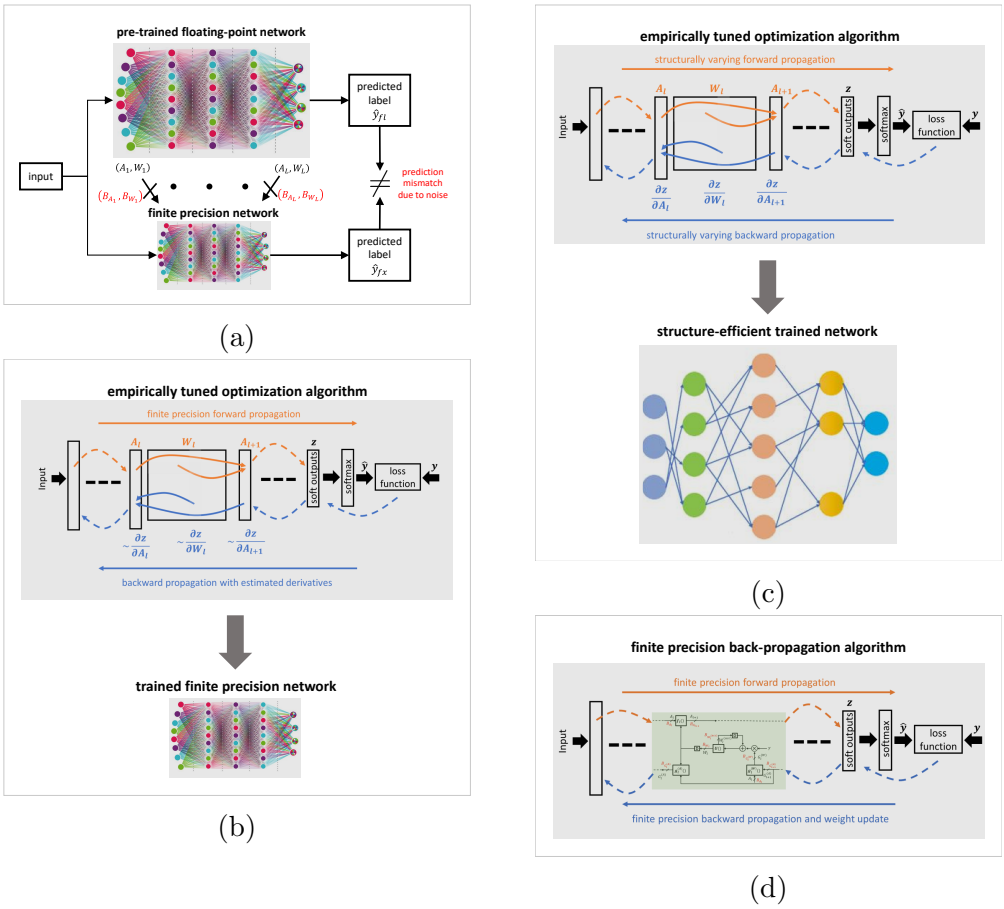


Figure 1.3: Approaches to reduced complexity machine learning: (a) post-training quantization, (b) in-training quantization, (c) structural methods, and (d) quantized back-propagation.

1.1 Related Work

In Figure 1.3, we cluster approaches to reduced complexity machine learning into four categories: (a) post-training quantization, (b) in-training quantization, (c) structural methods, and (d) quantized back-propagation. We henceforth describe each of these areas and review notable works related to them.

Post-training quantization is depicted in Figure 1.3 (a). The setup simply consists of taking a pre-trained floating-point network and implementing it in finite precision. The challenge in doing so is to determine suitable precision requirements for weights and activations across the multiple layers. A notable advantage of this approach is that it does not necessitate any training. Thus, any state-of-the art floating-point network realization

[1, 39, 3, 40, 41, 42, 43, 44] can be employed. Furthermore, accuracy guarantees can be obtained when using this approach as explained in Chapter 2 of this dissertation. Finally, as training is not required, privacy concerns are avoided [18].

Prior work in post-training quantization includes a signal-to-quantization (SQNR) study for custom fixed-point representations [45]. Despite being analytical, the proposed solution does not optimize the actual network-level accuracy and makes the unrealistic assumption of perfectly linear activations. More recently, a similar approach, termed Data Free Quantization (DFQ), was proposed [46] where an improvement to the analysis was done by leveraging converged BatchNorm [47] statistics. Other works in this area have proposed using hybrid high/low precision regimes to improve accuracy [48, 49]. While all of these works have the merit of leveraging high accuracy state-of-the-art DNNs, none provide guarantees on the accuracy of the finite precision derived network. Recent trends in post-training quantization are depicted in Figure 1.4.

In-training quantization is depicted in Figure 1.3 (b). Finite precision inference is embedded into the forward path of the back-propagation training loop. Since SGD is known to be noise tolerant, it is hoped that the training process compensates for forward quantization. If successful, accurate finite precision networks can be obtained. The SGD algorithm requires loss gradients of activations and weights. Forward quantization introduces a non-differentiable operation, and approximations are made in the backward path. Derivatives are computed using the straight-through estimator (STE) [50] which hides the discontinuous discretization operation from the optimization algorithm.

In-training quantization has two main drawbacks: 1) it combines optimization techniques that are not well understood theoretically (non-convex gradient descent and STE), and 2) it forces the user to be involved in the training process, which is a typically highly complex task. As such, it is currently unknown if this approach has any theoretical guarantees. Furthermore, training implies access to a dataset which can be problematic in certain applications. Nonetheless, in-training quantization has generated significant interest by virtue of numerous empirical successes.

Early work on in-training quantization showed that the STE can be used to obtain accurate binary weighted [51, 52] and fully binarized [51] networks,

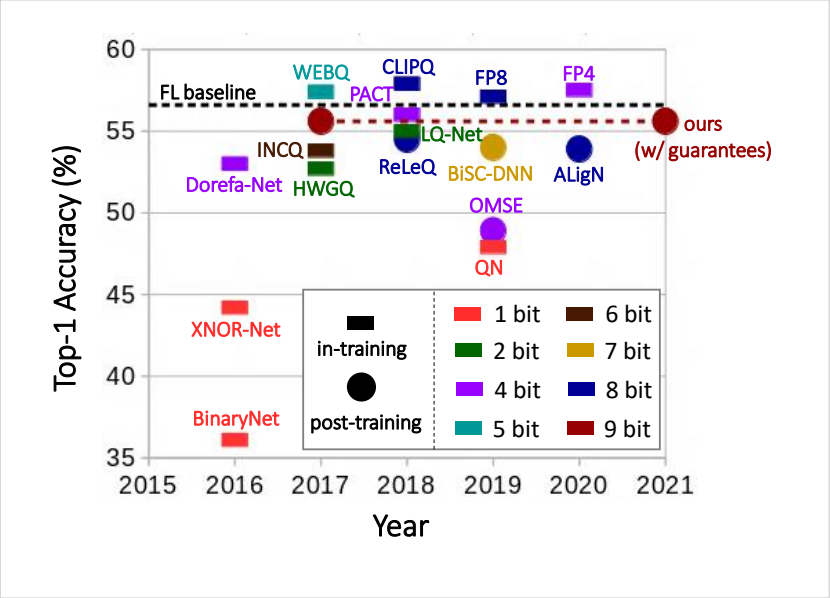
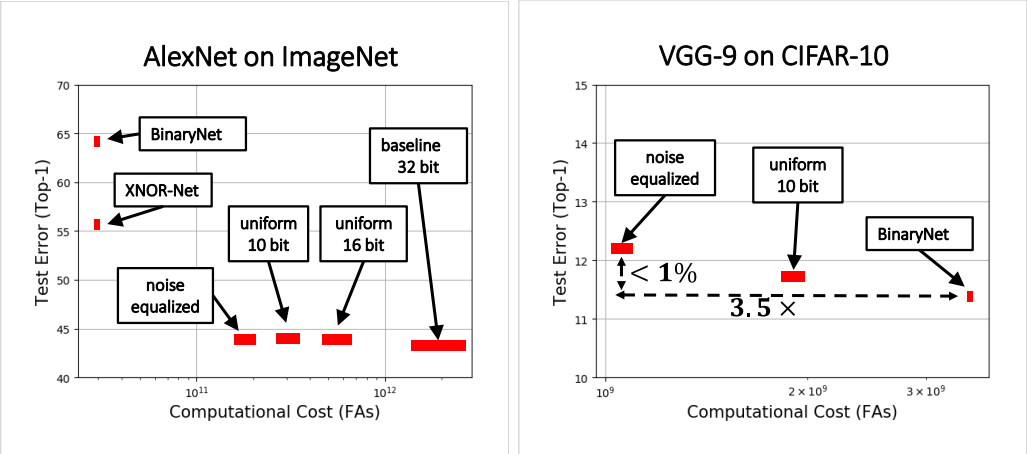


Figure 1.4: Recent trends in in-training and post-training quantization. Reported Top-1 accuracy is for AlexNet. Referred works are: BinaryNet [51], XNOR-Net [52], Dorefa-Net [53], HWGQ [54], INCQ [55], WEBQ [56], LQ-Net [57], PACT [58], CLIPQ [59], QN [60], FP8 [61], FP4 [24], ReLeQ [62], BISC-DNN [49], OMSE [63], and ALigN [64].

provided the employed networks and datasets are simple. While this surprised the community, it is appreciated today that state-of-the-art binary DNNs are still elusive when considering complex networks and datasets. DorefaNet [53] showed that by increasing forward precision to 2 bit or 4 bit, in-training quantization accuracy can be improved. LQNet [57] has achieved state-of-the-art accuracy with 2 bit forward representations. However, the quantization is non-uniform and the number format used is unconventional, which can cause hurdles in hardware implementations. PACT [58] has achieved state-of-the-art accuracy with 4 bit uniform quantization in the forward path. The feat was achieved by embedding clipping on the activations as a learnable parameter in the SGD loop. Recent trends in in-training quantization are depicted in Figure 1.4.

In-training quantization is often preferred to post-training quantization due to empirical evidence that it allows more aggressive precision reduction. However, post-training quantization, when properly employed, has the advantage of having accuracy guarantees. What is then the price to pay for such guarantees? More specifically, how much more reduction does in-



(a) (b)

Figure 1.5: Accuracy vs. Complexity of binarized network and post-training quantization approaches for: (a) AlexNet on ImageNet, and (b) VGG-9 on CIFAR-10. The computational cost counts the number of full additions (FAs) involved in processing one input.

training quantization offer compared to post-training quantization? Some works on post-training quantization such as BISC-DNN [49] and ALigN [64] have employed a precision of 7 bit and 8 bit, respectively. Nonetheless, these works have failed to maintain state-of-the-art accuracy. Our work has shown that 9 bit are typically required for guaranteeing accuracy close to that of the baseline (see Figure 1.4). In contrast, PACT [58], the present state-of-the-art in-training quantization technique, achieves similar accuracy using 4 bit. Thus, the price to pay for accuracy guarantees is at most 5 bit. More details are provided in Chapter 2 of this dissertation.

Figure 1.4 also shows that binarized networks, though popular, have yet to achieve state-of-the-art accuracy on large models and datasets such as AlexNet deployed on ImageNet. Common binarized networks such as BinaryNet [51] and XNOR-Net [52] have an accuracy drop with respect to the baseline of over 10 percentage points as shown in Figure 1.5 (a). Binarized networks do achieve state-of-the-art accuracy on simple models and datasets, such as a VGG-9 network on the CIFAR-10 [65] dataset (see Figure 1.5). Interestingly, we demonstrate $\sim 3.5\times$ complexity reduction over BinaryNet at iso-accuracy for that network. This is due to the fact that whenever binarized networks have achieved high accuracy, the network topology has usually been made wider, therefore causing the overall complexity to be higher.

Structural methods, such as model compression, constitute a set of inference efficiency techniques orthogonal but related to quantization. As depicted in Figure 1.3 (c), efficient models are searched during the optimization process via time-varying forward structures. The most popular structural method is pruning, or parameter sparsification [66]. Pruning can reduce the parameter complexity of large models by up to $\sim 100\times$. One limitation of pruning is that the resulting structure is very irregular and hence not hardware-friendly. Parameter clustering [67] is another approach which applies vector quantization algorithms on weight tensors. Recently, neural architecture search (NAS) [68] has emerged as a promising technique for finding structurally efficient networks.

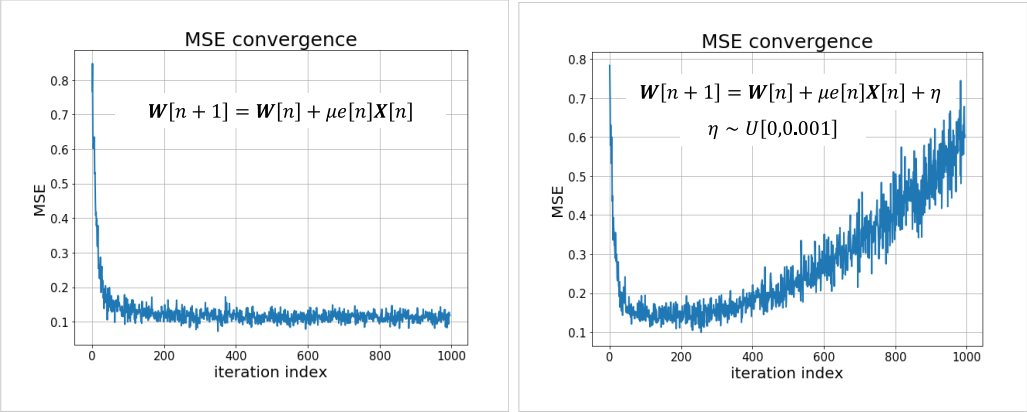
In general, the drawbacks of structural methods are similar to those of in-training quantization. Accuracy guarantees are lacking, and significant training complexity is required. To achieve state-of-the-art accuracy, the required structure-modifying training can take weeks and even months [69].

Combining the approaches discussed thus far is possible. A popular method to improve finite precision accuracy is to combine post- and in-training quantization. Often referred to as retraining [70], this hybrid approach performs in-training quantization with a state-of-the-art pre-trained network as an initial point. Quantization and structural techniques can also be applied together as was done in trained ternary quantization (TTQ) [71].

So far we have discussed inference efficiency techniques. To complete the story of reduced complexity machine learning, we also discuss efficient training. This problem is significantly harder than that of efficient inference.

Quantized back-propagation is used to reduce the complexity of the training algorithm and is depicted in Figure 1.3 (d). The goal is to implement the entire back-propagation algorithm in finite precision. It was first explored by quantizing all tensors to 16 bit fixed-point, save for the weight accumulators which remained in 32 bit floating-point [72]. Successful convergence was achieved thanks to the use of stochastic quantization.

Major advances since then have been achieved by exploring novel number formats. For instance, flexpoint [25] was proposed to track the dynamic range of 16 bit fixed-point tensors via the use of a 5 bit shared exponent. Augmenting flexpoint with stochastic quantization effectively results in WAGE [73]. Alternatively, novel low precision floating-point formats have been successful in training at 8 bit [74] and 4 bit [24]. The novelties have been in



(a) (b)

Figure 1.6: Difficulty in training with noise. Implementation of the least mean square (LMS) regression algorithm using: (a) correct update equations, and (b) biased noise in the updates.

modifying the exponent/mantissa representations and using radix-4 for the latter's exponent.

Many of the previous works have used stochastic quantization as a regularizer. All prior works have used full precision accumulators. Neither feature is hardware friendly, but both are necessary to avoid undesired biases in the weight updates. Indeed, the presence of a bias prohibits convergence in any feedback loop based algorithm, such SGD. We illustrate the phenomenon in Figure 1.6 where we implement a least mean square (LMS) regressor. When a non-zero mean noise term is inserted into the feedback computation, LMS fails to converge. In our work in Chapter 4, we show how to circumvent these issues in the context of training DNNs.

1.2 Dissertation Contributions and Organization

Much progress has been made in the area of finite precision machine learning. As discussed above, ad-hoc trial and error techniques abound, but none provide guarantees. Consequently, principled techniques guiding the design of deep learning systems on resource-constrained systems remain elusive. The work presented in this dissertation builds a theoretical framework for the implementation of deep learning in finite precision.

Specifically, the contributions of this dissertations are: (1) development

of theoretical bounds on inference accuracy of fixed-point deep neural networks, (2) derivation of optimal clipping strategies in dot-product outputs to minimize analog-to-digital converter precision in in-memory computing architectures, (3) methodology for implementing fixed-point back-propagation with close-to-minimal per-tensor precision, and (4) analysis of accumulation bit-width required in reduced precision floating-point training.

Contributions in this dissertation lead towards a framework for obtaining theoretical guarantees on DNN precision for both inference and training. For inference, dot-product input (weight/activation) and output (column ADC in IMC) precision is determined so as to guarantee accuracy. For training, precision of all back-propagation tensors (in fixed-point) and accumulated partial sums (in floating-point) is determined to guarantee convergence.

The remainder of this dissertation is organized as follows:

Chapter 2 introduces our work on fixed-point inference with accuracy guarantees. This work applies to the post-training quantization setup and enables design of minimum precision fixed-point networks. Unlike existing methods which determine precision in ad-hoc manners, via extensive simulations, our results are based on theoretical analyses and lead to accuracy guarantees. In particular, theoretical bounds on the mismatch between limited and full precision networks are derived. Consequently, proper precision assignment can be readily obtained employing these bounds, and weight-activation as well as per-layer precision trade-offs are derived. The proposed principled precision reduction is applied to a variety of networks and datasets. Results indicate that the presented analysis is tight within 2 bit and that theoretical bounds successfully predict trends of accuracy vs. precision. In particular, supported by empirical validation, our bounds reveal a $\sim 4\times$ increase in accuracy drop per bit reduction in precision. Furthermore, for small datasets, it is shown that a minimum precision network can have up to $\sim 3.5\times$ lower hardware complexity than a binarized network at iso-accuracy. For large models and datasets, a minimum precision network can reduce complexity by up to $\sim 10\times$ compared to a full precision baseline while maintaining accuracy. An ad-hoc quantized 8 bit network is shown to have higher complexity and lower accuracy than a minimum precision one, highlighting the importance of our methods. Per-layer precision analysis indicates that precision requirements of common networks vary from 2 bit to 10 bit for guaranteed accuracy.

Chapter 3 proposes the optimal clipping criterion (OCC) to minimize the column ADC precision of in-memory architectures. These ADCs consume a significant amount of energy and face stringent area constraints. Current practices in ADC precision design are overly conservative. Using our proposed OCC, ADC precision can be significantly reduced while maintaining accuracy, thereby reducing energy consumption and facilitating ADC design. It is first shown that the signal-to-quantization-noise ratio (SQNR) of OCC is within 0.8 dB of the well-known optimal Lloyd-Max (LM) quantizer. OCC improves the SQNR of the commonly employed full range (FR) quantizer by 14 dB which translates to a 3 bit ADC precision reduction. Furthermore, the input-serial weight-parallel (ISWP) IMC architecture is studied. Using bit-slicing techniques, significant energy savings can be achieved with minimal accuracy lost. Indeed, we prove that a dot-product can be realized with a single memory access while suffering no more than 2 dB SQNR drop. An analytical methodology for proper IMC design subject to DNN accuracy constraint is derived. Combining the proposed OCC and ISWP noise analysis, we demonstrate $\sim 6\times$ reduction of energy consumption at iso-accuracy.

Chapter 4 includes a systematic methodology to obtain close-to-minimal per-layer precision requirements for guaranteed statistical similarity between fixed-point and floating-point training. The difficulties in realizing quantized back-propagation arise from an improper understanding of feedback tensor quantization effects. Existing methods only focus on quantization of a subset of required tensors such as gradients. Further, current practices are typically based on heuristics where modifications to the number format are used, which cause hurdles in hardware implementations. In contrast, we study fixed-point training and address all of its challenges: quantization noise, inter-layer and intra-layer precision trade-offs, dynamic range, and stability. Our methodology is applied to several benchmarks, and fixed-point training is demonstrated to achieve high-fidelity to the baseline with an accuracy drop less than 0.56%. The derived precision assignment is shown to be within 1 bit per tensor of the minimum. The methodology is found to reduce representational, computational, and communication costs of training by up to $6\times$, $8\times$, and $4\times$, respectively, compared to the baseline and related works.

Chapter 5 presents an analysis of mantissa requirements in accumulators for guaranteed convergence of reduced precision floating-point training.

Most work on quantized back-propagation has focused on the issue of tensor representation precision. Such advances led to complexity reduction in multipliers employed in a hardware implementation. However, the problem of partial sum accumulation has been largely overlooked. Instead, full precision accumulators are used causing the complexity of a multiply-accumulate unit to be dominated by that of its adder. Our work addresses the problem of partial sum accumulation quantization via the definition of the variance retention ratio (VRR). This analytical metric measures the suitability of accumulation mantissa precision. The analysis expands on concepts employed in variance engineering for proper weight initialization. An analytical expression for the VRR is derived and used to determine accumulation bit-width for precise tailoring of computation hardware. The VRR analysis is also applied to techniques such as chunked accumulation and sparsification. Such techniques are known to alleviate the burden on accumulator precision requirements, and their benefits are quantified via the VRR. Experimentally, the validity and tightness of our analysis are verified across multiple deep learning benchmarks.

Chapter 6 concludes this dissertation by providing a comprehensive summary and directions for future research.

CHAPTER 2

FIXED-POINT INFERENCE WITH THEORETICAL GUARANTEES

In this chapter, we introduce our work on fixed-point inference with accuracy guarantees. We derive theoretical bounds on the misclassification rate in the presence of limited precision. Employing these bounds, we readily determine minimum precision assignment and derive weight-activation as well as per-layer precision trade-offs. The analysis is then applied to a variety of networks and datasets and is shown to be tight. Finally, it is found that our minimum precision networks offer the best complexity vs. accuracy trade-offs, outperforming even stat-of-the-art binarized networks.

2.1 Motivation

Neural networks have achieved state-of-the-art accuracy on many machine learning tasks. AlexNet [1] had a deep impact a few years ago in the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) and triggered intensive research efforts on deep neural networks. Recently, ResNet [3] has outperformed humans in recognition tasks.

These networks have very high computational complexity. For instance, AlexNet has 60 million parameters and 650,000 neurons [1]. Its convolutional layers alone require 666 million multiply-accumulates (MACs) per 227×227 image ($13k$ MACs/pixel) and 2.3 million weights [21]. Deepface's network involves more than 120 million parameters [9]. ResNet is a 152-layer deep residual network. This high complexity of deep neural networks prevents its deployment on energy and resource-constrained platforms such as mobile devices and autonomous platforms.

2.1.1 Related Work

One of the most effective approaches for reducing resource utilization is to implement fixed-point neural networks. As mentioned in [45], there are two approaches for designing fixed-point neural networks: (1) directly train a fixed-point neural network, and (2) quantize a pre-trained floating-point neural network to obtain a fixed-point network.

As an example of fixed-point training, [72] showed that 16 bit fixed-point representation incurs little accuracy degradation by using stochastic rounding. A more aggressive approach is to design binary networks such as [75] which used bitwise operations to replace the arithmetic operations and [52] which explored optimal binarization schemes. BinaryConnect [76] trained networks using binary weights while BinaryNet [51] trained networks with binary weights and activations.

Although these fixed-point training approaches make it possible to design fixed-point neural networks achieving excellent accuracy, training based on fixed-point arithmetic is generally harder than floating-point training since the optimization is done in a discrete space.

Hence, in this chapter, we focus on the second approach that quantizes a pre-trained floating-pointing network to a fixed-point network. This approach leverages the extensive work in training state-of-the-art floating-point neural networks such as dropout [44], maxout [42], network-in-network [43], and residual learning [3] to name a few. In this approach, proper precision needs to be determined after training to reduce complexity while minimizing the accuracy loss. In [77], exhaustive search is performed to determine a suitable precision allocation. Recently, [45] offered an analytical solution for non-uniform bit precision based on the signal-to-quantization-noise ratio (SQNR). However, the use of non-uniform quantization step sizes at each layer is difficult to implement as it requires multiple variable precision arithmetic units.

In addition to fixed-point implementation, many approaches have been proposed to lower the complexity of deep neural networks in terms of the number of arithmetic operations. Han et al. [78] employ a three-step training method to identify important connections, prune the unimportant ones, and retrain on the pruned network. In [79] original convolutional layers are replaced by smaller sequential layers to reduce computations. These ap-

proaches are complementary to our technique of quantizing a pre-trained floating-point neural network into a fixed-point one.

In this chapter, we obtain analytical bounds on the accuracy of fixed-point networks that are obtained by quantizing a conventionally trained floating-point network. Furthermore, by defining meaningful measures of a fixed-point network's hardware complexity, viz. computational and representational costs, we develop a principled approach to precision assignment using these bounds in order to minimize these complexity measures.

2.1.2 Contributions

Our contributions are both theoretical and practical. We summarize our main contributions:

- We derive *theoretical bounds* on the misclassification rate in presence of limited precision and thus determine *analytically how accuracy and precision trade-off with each other*.
- Employing the theoretical bounds and the back-propagation algorithm, we show that *proper precision assignments* can be readily determined while *maintaining accuracy close to floating-point networks*.
- We analytically determine which of weights or activations need more precision, and we show that typically *the precision requirements of weights are greater than those of activations* for fully-connected networks and are similar to each other for networks with shared weights such as convolutional neural networks.
- We introduce *computational and representational costs* as meaningful metrics to evaluate the complexity of neural networks under fixed precision assignment.
- We validate our findings on numerous networks and datasets demonstrating the ease with which fixed-point networks with *complexity smaller than state-of-the-art binary networks* can be derived from pre-trained floating-point networks with minimal loss in accuracy.

It is worth mentioning that our proposed method is general and can be applied to every class of neural networks such as multilayer perceptrons and convolutional neural networks.

2.2 Fixed-Point Neural Networks

2.2.1 Accuracy of Fixed-Point and Floating-Point Networks

For a given floating-point neural network and its fixed-point counterpart we define: 1) the floating-point error probability $p_{e,fl} = \Pr\{\hat{Y}_{fl} \neq Y\}$ where $\hat{Y}_{fl}$ is the output of the floating-point network and Y is the true label; 2) the fixed-point error probability $p_{e,fx} = \Pr\{\hat{Y}_{fx} \neq Y\}$ where $\hat{Y}_{fx}$ is the output of the fixed-point network; 3) the mismatch probability between fixed-point and floating-point $p_m = \Pr\{\hat{Y}_{fx} \neq \hat{Y}_{fl}\}$. Observe that:

$$p_{e,fx} \leq p_{e,fl} + p_m. \tag{2.1}$$

The right-hand side represents the worst case of having no overlap between misclassified samples and samples whose predicted labels are in error due to quantization. We provide a formal proof of (2.1) in Section 2.9. Note that $p_{e,fx}$ is a quantity of interest as it characterizes the accuracy of the fixed-point system. We employ p_m as a proxy to $p_{e,fx}$ because it brings in the effects of quantization into the picture as opposed to $p_{e,fl}$ which solely depends on the algorithm. This observation was made in [80] and allowed for an analytical characterization of linear classifiers as a function of precision.

2.2.2 Fixed-Point Quantization

The study of fixed-point systems and algorithms is well established in the context of signal processing and communication systems [12]. A popular example is the least mean square (LMS) algorithm for which bounds on precision requirements for input, weights, and updates have been derived [81]. In such analyses, it is standard practice [82] to assume all signed quantities lie in $[-1, 1]$ and all unsigned quantities lie in $[0, 2]$. Of course, activations and weights can be designed to satisfy this assumption during training. A B

bit fixed-point number a_{fx} would be related to its floating-point value a as follows:

$$a_{fx} = a + q_a, \tag{2.2}$$

where q_a is the quantization noise which is modeled as an independent uniform random variable distributed over $\left[-\frac{\Delta}{2}, \frac{\Delta}{2}\right]$, where $\Delta = 2^{-(B-1)}$ is the quantization step [82].

2.2.3 Complexity in Fixed-Point

We argue that the complexity of a fixed-point system has two aspects: computational and representational costs. In what follows, we consider activations and weights to be quantized to B_A and B_W bit, respectively.

The *computational cost* is a measure of the computational resources utilized for generating a single decision, and is defined as the number of 1 bit full adders (FAs). A full adder is a canonical building block of arithmetic units. We assume arithmetic operations are executed using the commonly used ripple carry adder [83] and Baugh-Wooley multiplier [84] architectures designed using FAs. Consequently, the number of FAs used to compute a D-dimensional dot product of activations and weights is [85]:

$$DB_AB_W + (D-1)(B_A + B_W + \lceil \log_2(D) \rceil - 1). \tag{2.3}$$

Hence, an important aspect of the computational cost of a dot product is that it is an increasing function of the *product* of activation precision (B_A), weight precision (B_W), and dimension (D).

We define the *representational cost* as the total number of bits needed to represent all network parameters, i.e., both activations and weights. This cost is a measure of the storage complexity and communications costs associated with data movement. The total representational cost of a fixed-point network is:

$$|\mathcal{A}|\, B_A + |\mathcal{W}|\, B_W \tag{2.4}$$

bits, where $\mathcal{A}$ and $\mathcal{W}$ are the index sets of all activations and weights in

the network, respectively. Observe that the representational cost is *linear* in activation and weight precisions as compared to the computational cost.

Equations (2.3) - (2.4) illustrate that, though computational and representational costs are not independent, they are different. Together, they describe the implementation costs associated with a network. We shall employ both when evaluating the complexity of fixed-point networks.

2.2.4 Setup

Here, we establish notation. Let us consider neural networks deployed on a M-class classification task. For a given input, the network would typically have M class scores $\{z_i\}_{i=1}^{M}$ and the decision would be $\hat{y} = \arg\max_{i=1,\ldots,M} z_i$. Each numerical output is a function of weights and activations in the network:

$$z_i = f\left(\{a_h\}_{h\in\mathcal{A}}, \{w_h\}_{h\in\mathcal{W}}\right) \tag{2.5}$$

for $i = 1, \ldots, M$, where a_h denotes the activation indexed by h and w_h denotes the weight indexed by h. When activations and weights are quantized to B_A and B_W bits, respectively, the output z_i is corrupted by quantization noise q_{z_i} so that:

$$z_i + q_{z_i} = f\left(\{a_h + q_{a_h}\}_{h\in\mathcal{A}}, \{w_h + q_{w_h}\}_{h\in\mathcal{W}}\right), \tag{2.6}$$

where q_{a_h} and q_{w_h} are the quantization noise terms of the activation a_h and weight w_h, respectively. Here, $\{q_{a_h}\}_{h\in\mathcal{A}}$ are independent uniformly distributed random variables on $\left[-\frac{\Delta_A}{2}, \frac{\Delta_A}{2}\right]$ and $\{q_{w_h}\}_{h\in\mathcal{W}}$ are independent uniformly distributed random variables on $\left[-\frac{\Delta_W}{2}, \frac{\Delta_W}{2}\right]$, with $\Delta_A = 2^{-(B_A-1)}$ and $\Delta_W = 2^{-(B_W-1)}$.

In quantization noise analysis, it is standard to ignore cross-products of quantization noise terms as their contribution is negligible. Therefore, using Taylor's theorem, we express the total quantization noise at the output of the fixed-point network as:

$$q_{z_i} = \sum_{h\in\mathcal{A}} q_{a_h} \frac{\partial z_i}{\partial a_h} + \sum_{h\in\mathcal{W}} q_{w_h} \frac{\partial z_i}{\partial w_h}. \tag{2.7}$$

Note that the derivatives in (2.7) are obtained as part of the back-propagation

algorithm. Thus, using our results, it is possible to estimate the precision requirements of deep neural networks during training itself. As will be shown later, this requires one additional back-propagation iteration to be executed after the weights have converged.

2.3 Bounds on Mismatch Probability

2.3.1 Second-order Bound

We present our first result. It is an analytical upper bound on the mismatch probability p_m between a fixed-point neural network and its floating-point counterpart.

Theorem 1. *Given B_A and B_W, the mismatch probability p_m between a fixed-point network and its floating-point counterpart is upper bounded as follows:*

$$p_m \leq \frac{\Delta_A^2}{24}\mathbb{E}\left[\sum_{\substack{i=1 \\ i\neq\hat{Y}_{fl}}}^{M} \frac{\sum_{h\in\mathcal{A}}\left|\frac{\partial(Z_i - Z_{\hat{Y}_{fl}})}{\partial A_h}\right|^2}{|Z_i - Z_{\hat{Y}_{fl}}|^2}\right] + \frac{\Delta_W^2}{24}\mathbb{E}\left[\sum_{\substack{i=1 \\ i\neq\hat{Y}_{fl}}}^{M} \frac{\sum_{h\in\mathcal{W}}\left|\frac{\partial(Z_i - Z_{\hat{Y}_{fl}})}{\partial w_h}\right|^2}{|Z_i - Z_{\hat{Y}_{fl}}|^2}\right], \tag{2.8}$$

where expectations are taken over a random input and $\{A_h\}_{h\in\mathcal{A}}$, $\{Z_i\}_{i=1}^{M}$, and $\hat{Y}_{fl}$ are thus random variables.

Proof. The detailed proof can be found in Section 2.9. Here, we provide the main idea and the intuition behind the proof.

The heart of the proof lies in evaluating $\Pr\left(z_i + q_{z_i} > z_j + q_{z_j}\right)$ for any pair of outputs z_i and z_j where $z_j > z_i$. Equivalently, we need to evaluate $\Pr\left(q_{z_i} - q_{z_j} > z_j - z_i\right)$. But from (2.7), we have:

$$q_{z_i} - q_{z_j} = \sum_{h\in\mathcal{A}} q_{a_h}\frac{\partial(z_i - z_j)}{\partial a_h} + \sum_{h\in\mathcal{W}} q_{w_h}\frac{\partial(z_i - z_j)}{\partial w_h}. \tag{2.9}$$

In (2.9), we have a linear combination of quantization noise terms, and $q_{z_i} - q_{z_j}$ is a zero mean random variable having a symmetric distribution. This means

that $\Pr\left(q_{z_i} - q_{z_j} > z_j - z_i\right) = \frac{1}{2}\Pr\left(|q_{z_i} - q_{z_j}| > |z_j - z_i|\right)$, which allows us to use Chebyshev's inequality. Indeed, from (2.9), the variance of $q_{z_i} - q_{z_j}$ is given by:

$$\frac{\Delta_A^2}{12}\sum_{h\in\mathcal{A}}\left|\frac{\partial(z_i - z_j)}{\partial a_h}\right|^2 + \frac{\Delta_W^2}{12}\sum_{h\in\mathcal{W}}\left|\frac{\partial(z_i - z_j)}{\partial w_h}\right|^2,$$

so that

$$\Pr\left(z_i + q_{z_i} > z_j + q_{z_j}\right) \leq \frac{\Delta_A^2\sum_{h\in\mathcal{A}}\left|\frac{\partial(z_i - z_j)}{\partial a_h}\right|^2 + \Delta_W^2\sum_{h\in\mathcal{W}}\left|\frac{\partial(z_i - z_j)}{\partial w_h}\right|^2}{24\,|z_i - z_j|^2}. \tag{2.10}$$

As explained in Section 2.9, it is possible to obtain to (2.8) from (2.10) using standard probabilistic arguments. □

Before proceeding, we point out that the two expectations in (2.8) are taken over a random input but the weights $\{w_h\}_{h\in\mathcal{W}}$ are frozen after training and are hence deterministic.

Several observations are to be made. First notice that the mismatch probability p_m increases with Δ_A^2 and Δ_W^2. This is to be expected as smaller precision leads to more mismatch. Theorem 1 says a little bit more: the mismatch probability decreases exponentially with precision because $\Delta_A = 2^{-(B_A-1)}$ and $\Delta_W = 2^{-(B_W-1)}$.

Note that the quantities in the expectations in (2.8) can be obtained as part of a standard back-propagation procedure. Indeed, once the weights are frozen, it is enough to perform one forward pass on an estimation set (which should have statistically significant cardinality), record the numerical outputs, perform one backward pass and probe all relevant derivatives. Thus, (2.8) can be readily computed.

Another practical aspect of Theorem 1 is that this operation needs to be done only once as these quantities do not depend on precision. Once they are determined, for any given precision assignment, we simply evaluate (2.8) and combine it with (2.1) to obtain an estimate (upper bound) on the accuracy of the fixed-point instance. This way the precision necessary to achieve a specific mismatch probability is obtained from a trained floating-point network. This clearly highlights the gains in practicality of our analytical approach over a

trial-and-error based search.

Finally, (2.8) reveals a very interesting aspect of the trade-off between activation precision B_A and weight precision B_W. We rewrite (2.8) as:

$$p_m \leq \Delta_A^2 E_A + \Delta_W^2 E_W, \tag{2.11}$$

where

$$E_A = \mathbb{E}\left[\sum_{\substack{i=1 \\ i \neq \hat{Y}_{fl}}}^{M} \frac{\sum_{h \in \mathcal{A}} \left| \frac{\partial (Z_i - Z_{\hat{Y}_{fl}})}{\partial A_h} \right|^2}{24 |Z_i - Z_{\hat{Y}_{fl}}|^2}\right]$$

and

$$E_W = \mathbb{E}\left[\sum_{\substack{i=1 \\ i \neq \hat{Y}_{fl}}}^{M} \frac{\sum_{h \in \mathcal{W}} \left| \frac{\partial (Z_i - Z_{\hat{Y}_{fl}})}{\partial w_h} \right|^2}{24 |Z_i - Z_{\hat{Y}_{fl}}|^2}\right].$$

The first term in (2.11) characterizes the impact of quantizing activations on the overall accuracy while the second characterizes that of weight quantization. It might be the case that one of the two terms dominates the sum depending on the values of E_A and E_W. This means that either the activations or the weights are assigned more precision than necessary. An intuitive first step to efficiently get a smaller upper bound is to make the two terms of comparable order. That can be made by setting $\Delta_A^2 E_A = \Delta_W^2 E_W$ which is equivalent to

$$B_A - B_W = round\left(\log_2 \sqrt{\frac{E_A}{E_W}}\right), \tag{2.12}$$

where $round()$ denotes the rounding operation. This is an effective way of taking care of one of the two degrees of freedom introduced by (2.8).

A natural question to ask would be which of E_A and E_W is typically larger. That is to say, to whom, activations or weights, should one assign more precision? In deep neural networks, there are more weights than activations, a trend particularly observed in deep networks with most layers fully connected. This trend, though not as pronounced, is also observed in networks with shared weights, such as convolutional neural networks. However, there exist a few counterexamples such as the networks in [51] and [86]. It is thus reasonable to expect $E_W \geq E_A$, and consequently *the precision requirements*

of weights will, in general, be more than those of activations.

One way to interpret (2.11) is to consider minimizing the upper bound in (2.8) subject to $B_A + B_W = c$ for some constant c. Indeed, it can be shown that (2.12) would be a necessary condition of the corresponding solution. This is an application of the arithmetic-geometric mean inequality. Effectively, (2.11) is of particular interest when considering computational cost which increases as a function of the product of both precisions (see Section 2.2.3).

2.3.2 Tighter Bound

We present a tighter upper bound on p_m based on the Chernoff bound.

Theorem 2. *Given B_A and B_W, the mismatch probability p_m between a fixed-point network and its floating-point counterpart is upper bounded as follows:*

$$p_m \leq \mathbb{E}\left[\sum_{\substack{i=1 \\ i\neq \hat{Y}_{fl}}}^{M} e^{-S^{(i,\hat{Y}_{fl})}} P_1^{(i,\hat{Y}_{fl})} P_2^{(i,\hat{Y}_{fl})}\right], \tag{2.13}$$

where, for $i \neq j$,

$$S^{(i,j)} = \frac{3(Z_i - Z_j)^2}{\sum_{h\in\mathcal{A}} \left(D_{A_h}^{(i,j)}\right)^2 + \sum_{h\in\mathcal{W}} \left(D_{w_h}^{(i,j)}\right)^2},$$

$$D_{A_h}^{(i,j)} = \frac{\Delta_A}{2}\frac{\partial(Z_i - Z_j)}{\partial A_h}, \quad D_{w_h}^{(i,j)} = \frac{\Delta_W}{2}\frac{\partial(Z_i - Z_j)}{\partial w_h},$$

$$P_1^{(i,j)} = \prod_{h\in\mathcal{A}} \frac{\sinh\left(T^{(i,j)} D_{A_h}^{(i,j)}\right)}{T^{(i,j)} D_{A_h}^{(i,j)}},$$

$$P_2^{(i,j)} = \prod_{h\in\mathcal{W}} \frac{\sinh\left(T^{(i,j)} D_{w_h}^{(i,j)}\right)}{T^{(i,j)} D_{w_h}^{(i,j)}},$$

and

$$T^{(i,j)} = \frac{S^{(i,j)}}{Z_j - Z_i}.$$

Proof. Again, we leave the technical details for Section 2.9. Here we also provide the main idea and intuition.

As in Theorem 1, we shall focus on evaluating $\Pr\left(z_i + q_{z_i} > z_j + q_{z_j}\right) = \Pr\left(q_{z_i} - q_{z_j} > z_j - z_i\right)$ for any pair of outputs z_i and z_j where $z_j > z_i$. The key difference here is that we will use the Chernoff bound in order to account for the complete quantization noise statistics. Indeed, letting $v = z_j - z_i$, we have:

$$\Pr\left(q_{z_i} - q_{z_j} > v\right) \leq e^{-tv}\mathbb{E}\left[e^{t(q_{z_i} - q_{z_j})}\right]$$

for any $t > 0$. We show that:

$$\mathbb{E}\left[e^{t(q_{z_i} - q_{z_j})}\right] = \prod_{h\in\mathcal{A}} \frac{\sinh\left(td_{a,h}\right)}{td_{a,h}} \prod_{h\in\mathcal{W}} \frac{\sinh\left(td_{w,h}\right)}{td_{w,h}},$$

where $d_{a,h} = \frac{\Delta_A}{2}\frac{\partial(z_i - z_j)}{\partial a_h}$ and $d_{w,h} = \frac{\Delta_W}{2}\frac{\partial(z_i - z_j)}{\partial w_h}$. This yields:

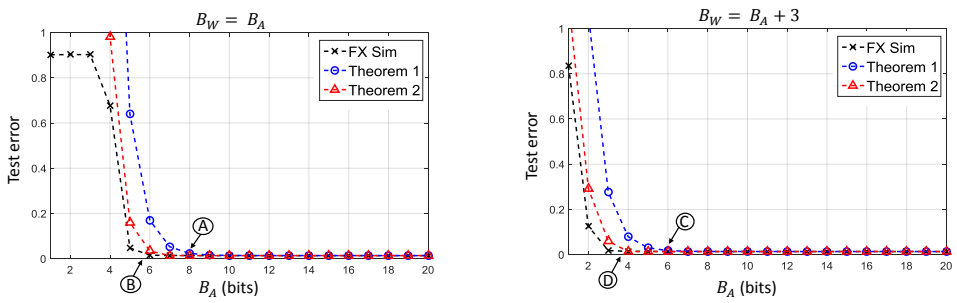
$$\Pr\left(q_{z_i} - q_{z_j} > v\right) \leq e^{-tv} \prod_{h\in\mathcal{A}} \frac{\sinh\left(td_{a,h}\right)}{td_{a,h}} \prod_{h\in\mathcal{W}} \frac{\sinh\left(td_{w,h}\right)}{td_{w,h}}. \tag{2.14}$$

We show that the right-hand-side is minimized over positive values of t when:

$$t = \frac{3v}{\sum_{h\in\mathcal{A}}\left(d_{a,h}\right)^2 + \sum_{h\in\mathcal{W}}\left(d_{w,h}\right)^2}.$$

Substituting this value of t into (2.14) and using standard probabilistic arguments, we obtain (2.13). □

The first observation to be made is that Theorem 2 indicates that, on average, p_m is upper bounded by an exponentially decaying function of the quantity $S^{(i,\hat{Y}_{fl})}$ for all $i \neq \hat{Y}_{fl}$ up to a correction factor $P_1^{(i,\hat{Y}_{fl})}P_2^{(i,\hat{Y}_{fl})}$. This correction factor is a product of terms typically centered around 1 (each term is of the form $\frac{\sinh(x)}{x} \approx 1$ for small x). On the other hand, $S^{(i,\hat{Y}_{fl})}$, by definition, is the ratio of the excess confidence the floating-point network has in the label $\hat{Y}_{fl}$ over the total quantization noise variance reflected at the output, i.e., $S^{(i,\hat{Y}_{fl})}$ is the SQNR. Hence, Theorem 2 states that the tolerance of a neural network to quantization is, on average, exponentially decaying with the SQNR at its output. In terms of precision, Theorem 2 states that p_m is bounded by a double exponentially decaying function of precision (that



(a) (b)

Figure 2.1: Validity of bounds for MNIST when: (a) $B_W = B_A$ and (b) $B_W = B_A + 3$ as dictated by (2.12) ($E_A = 41$ and $E_W = 3803$ so that $\log_2 \sqrt{\frac{E_W}{E_A}} \approx 3.2$). FX Sim denotes fixed point simulations.

is an exponential function of an exponential function). Note how this bound is tighter than that of Theorem 1.

This double exponential relationship between accuracy and precision is not too surprising when one considers the problem of binary hypothesis testing under additive Gaussian noise [87] scenario. In this scenario, it is well known that the probability of error is an exponentially decaying function of the signal-to-noise ratio (SNR) in the high-SNR regime. Theorem 2 points out a similar relationship between accuracy and precision but it does so using rudimentary probability principles without relying on high-SNR approximations.

While Theorem 2 is much tighter than Theorem 1 theoretically, it is not as convenient to use. In order to use Theorem 2, one has to perform a forward-backward pass and select relevant quantities and apply (2.13) for each choice of B_A and B_W. However, a lot of information, e.g. the derivatives, can be reused at each run, and so the runs may be lumped into one forward-backward pass. In a sense, the complexity of computing the bound in Theorem 2 lies between the evaluation of (2.11) and the complicated conventional trial-and-error based search.

We now illustrate the applications of these bounds.

Table 2.1: Results for MNIST: Comparison of accuracy, computational cost, and representational cost with state-of-the-art related works. Chosen precision assignments are obtained from Figure 2.1.

Precision Assignment	Test error (%)	Computational Cost (10^6 FAs)	Representational Cost (10^6 bits)
Floating-point	1.36	N/A	N/A
$(8, 8)$	1.41	82.9	7.5
$(6, 6)$	1.54	53.1	**5.63**
$(6, 9)$	**1.35**	72.7	8.43
$(4, 7)$	1.43	**44.7**	6.54
SQ $(16, 16)$ [72]	1.4	533	28
BN $(1, 1)$ [51]	1.4	117	10

2.4 Bound Validation Results

We conduct numerical simulations to illustrate both the validity and usefulness of the analysis developed in the previous section. We show how it is possible to reduce precision in an aggressive yet principled manner. We present results on two popular datasets: MNIST and CIFAR-10. The metrics we address are threefold:

- Accuracy measured in terms of test error.
- Computational cost measured in $\#FAs$ (see Section 2.2.3, (2.3) was used to compute $\#FAs$ per MAC).
- Representational cost measured in bits (see Section 2.2.3, (2.4) was used).

We compare our results to similar works conducting similar experiments: 1) the work on fixed-point training with stochastic quantization (SQ) [72] and 2) BinaryNet (BN) [51].

2.4.1 DNN on MNIST

First, we conduct simulations on the MNIST dataset for handwritten character recognition [88]. The dataset consists of 60K training samples and 10K test samples. Each sample consists of an image and a label. Images are of

size 28×28 pixels representing a handwritten digit between 0 and 9. Labels take the value of the corresponding digit.

In this first experiment, we chose an architecture of $784 - 512 - 512 - 512 - 10$, i.e., 3 hidden layers, each of 512 units. We first trained the network in floating-point using the back-propagation algorithm. We used a batch size of 200 and a learning rate of 0.1 with a decay rate of 0.978 per epoch. We restore the learning rate every 100 epochs, the decay rate makes the learning rate vary between 0.1 and 0.01. We train the first 300 epochs using 15% dropout, the second 300 epochs using 20% dropout, and the third 300 epochs using 25% dropout (900 epochs overall). It appears from the original dropout work [44] that the typical 50% dropout fraction works best for very wide multi-layer perceptrons (MLPs) (4096 to 8912 hidden units). For this reason, we chose to experiment with smaller dropout fractions.

The only pre-processing done is to scale the inputs between -1 and 1. We used ReLU activations with the subtle addition of a right rectifier for values larger than 2 (as discussed in Section 2.2). The resulting activation is also called a hard sigmoid. We also clipped the weights to lie in $[-1, 1]$ at each iteration. The resulting test error we obtained in floating-point is 1.36%.

Figure 2.1 illustrates the validity of our analysis. Indeed, both bounds (based on Theorems 1 and 2) successfully upper bound the test error obtained through fixed-point simulations. Figure 2.1 (b) demonstrates the utility of (2.12). Indeed, setting $B_W = B_A$ allows us to reduce the precision to about 6 or 7 bit before the accuracy start degrading. In addition, under these conditions we found $E_A = 41$ and $E_W = 3803$ so that $\log_2 \sqrt{\frac{E_W}{E_A}} \approx 3.2$. Thus, setting $B_W = B_A + 3$ as dictated by (2.12) allows for more aggressive precision reduction. Activation precision B_A can now be reduced to about 3 or 4 bit before the accuracy degrades. To compute the bounds, we used an estimation set of 1000 random samples from the dataset.

As observed in Figure 2.1, the bounds closely track the increase in test error. Indeed, it is found that the test error increase roughly quadruples for every bit reduced. This is in accordance with our theoretical results where p_m has an exponential relationship with precision and is expected to quadruple as the number of bits is decremented. This relationship between accuracy and precision can be exploited in end-to-end system design where a fixed-point network's accuracy needs to be tuned in order to achieve an overall inference accuracy.

We compare our results with SQ which used a $784-1000-1000-10$ architecture on 16 bit fixed-point activations and weights. A stochastic rounding scheme was used to compensate for quantization. We also compare our results with BN with a $784-2048-2048-2048-10$ architecture on binary quantities. A stochastic rounding scheme was also used during training.

Table 2.1 shows some comparisons with related works in terms of accuracy, computational cost, and representational cost. For comparison, we selected four notable design options from Figures 2.1 (a,b):

A. Smallest (B_A, B_W) such that $B_W = B_A$ and $p_m \leq 1\%$ as bounded by Theorem 1. In this case $(B_A, B_W) = (8, 8)$.

B. Smallest (B_A, B_W) such that $B_W = B_A$ and $p_m \leq 1\%$ as bounded by Theorem 2. In this case $(B_A, B_W) = (6, 6)$.

C. Smallest (B_A, B_W) such that $B_W = B_A + 3$ as dictated by (2.12) and $p_m \leq 1\%$ as bounded by Theorem 1. In this case $(B_A, B_W) = (6, 9)$.

D. Smallest (B_A, B_W) such that $B_W = B_A + 3$ as dictated by (2.12) and $p_m \leq 1\%$ as bounded by Theorem 2. In this case $(B_A, B_W) = (4, 7)$.

As can be seen in Table 2.1, the accuracy is similar across all design options including the results reported by SQ and BN. Interestingly, for all four design options, our network has a *smaller computational cost than BN*. In addition, SQ's computational cost is about $4.6\times$ that of BN (533M/117M). The greatest reduction in computational cost is obtained for a precision assignment of $(4, 7)$ corresponding to a $2.6\times$ and $11.9\times$ reduction compared to BN (117M/44.7M) and SQ (533M/44.7M), respectively. The corresponding test error rate is of 1.43%. Similar trends are observed for representational costs. Again, our four designs have *smaller representational cost than even BN*. BN itself has $2.8\times$ smaller representational cost than SQ (28M/10M). Note that a precision assignment of $(6, 6)$ yields $1.8\times$ and $5.0\times$ smaller representational costs than BN (10M/5.63M) and SQ (28M/5.63M), respectively. The corresponding test error rate is 1.54%.

The fact that we are able to achieve lesser computational and representational costs than BN while maintaining similar accuracy highlights two important points. First, the width of a network severely impacts its complexity. We made our network four times as narrow as BN's and still managed to use

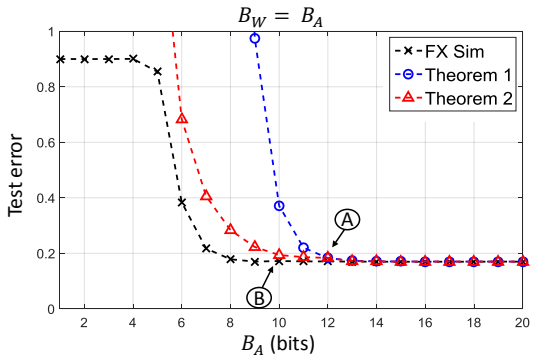


Figure 2.2: Validity of bounds for CIFAR when $B_W = B_A$ which is also dictated by (2.12) ($E_A = 21033$ and $E_W = 31641$ so that $\log_2 \sqrt{\frac{E_W}{E_A}} \approx 0.29$). FX Sim denotes fixed point simulations.

Table 2.2: Results for CIFAR-10: Comparison of accuracy, computational cost, and representational cost with state-of-the-art related works. Chosen precision assignments are obtained from Figure 2.2.

Precision Assignment	Test error (%)	Computational Cost (10^6 FAs)	Representational Cost (10^6 bits)
Floating-point	17.02	N/A	N/A
$(12, 12)$	17.08	3626	5.09
$(10, 10)$	17.23	**2749**	**4.24**
SQ $(16, 16)$ [72]	25.4	4203	4.54
BN $(1, 1)$ [51]	**10.15**	3608	6.48

eight times as many bits per parameter without exceeding BN's complexity. Second, our results illustrate the strength of numbering systems, specifically, the strength of fixed-point representations. Our results indicate that a correct and meaningful multi-bit representation of parameters is better in both complexity and accuracy than a 1 bit unstructured allocation.

2.4.2 CNN on CIFAR 10

We conduct a similar experiment on the CIFAR10 dataset [65]. The dataset consists of 60k color images each representing airplanes, automobiles, birds, cats, deer, dogs, frogs, horses, ships, and trucks. Of these images, 50k constitute the training set, and the 10k remaining are for testing. SQ's architecture

on this dataset is a simple one: three convolutional layers, interleaved by max pooling layers. The output of the final pooling layer is fed to a 10-way softmax output layer. The reported accuracy using 16 bit fixed-point arithmetic is a 25.4% test error. BN's architecture is a much wider and deeper architecture based on VGG [39]. The reported misclassification rate of the binary network is an impressive 10.15% which is of benchmarking quality even for full precision networks.

We adopt a similar architecture as SQ, but leverage recent advances in convolutional neural networks (CNNs) research. It has been shown that adding networks within convolutional layers (in the 'Network in Network' sense) as described in [43] significantly enhances accuracy, while not incurring much complexity overhead. Hence, we replace SQ's architecture by a deep one which we describe as $64C5-64C1-64C1-MP2-64C5-64C1-64C1-MP2-64C5-64FC-64FC-64FC-10$, where $C5$ denotes 5×5 kernels, $C1$ denotes 1×1 kernels (they emulate the networks in networks), $MP2$ denotes 2×2 max pooling, and FC denotes fully connected components. As is customary for this dataset, we apply zero-phase component analysis (ZCA) whitening to the data before training. Because this dataset is a challenging one, we first fine-tune the hyperparameters (learning rate, weight decay rate, and momentum), then train for 300 epochs. The best accuracy we reach in floating point using this 12-layer deep network is 17.02%.

Figure 2.2 shows the results of our fixed-point simulation and analysis. Note that, while both bounds from Theorems 1 and 2 still successfully upper bound the test error, these are not as tight as in our MNIST experiment. Furthermore, in this case, (2.12) dictates keeping $B_W = B_A$ as $E_A = 21033$ and $E_W = 31641$ so that $\log_2 \sqrt{\frac{E_W}{E_A}} \approx 0.29$. The fact that $E_W \geq E_A$ is expected as there are typically more weights than activations in a neural network. However, note that in this case the contrast between E_W and E_A is not as sharp as in our MNIST experiment. This is mainly due to the higher weight to activation ratio in fully connected DNNs than in CNNs. Furthermore, the trends of test error increase as a function of precision are once again verified. The mismatch rate approximately quadruples when bitwidth is decremented as estimated by our theoretical bounds.

We again select two design options:

A. Smallest (B_A, B_W) such that $B_W = B_A$ and $p_m \leq 1\%$ as bounded by

Theorem 1. In this case $(B_A, B_W) = (12, 12)$.

B. Smallest (B_A, B_W) such that $B_W = B_A$ and $p_m \leq 1\%$ as bounded by Theorem 2. In this case $(B_A, B_W) = (10, 10)$.

Table 2.2 indicates that BN is the most accurate with 10.15% test error. Interestingly, it has lesser computational cost but more representational cost than SQ. This is due to the dependence of the computational cost on the *product* of B_A and B_W. The least complex network is ours when setting $(B_A, B_W) = (10, 10)$ and its test error is 17.23% which is already a large improvement on SQ in spite of having smaller computational and representational costs. This network is also less complex than that of BN.

The main takeaway here is that CNNs are quite different from fully connected DNNs when it comes to precision requirements. Furthermore, from Table 2.2 we observe that BN achieves the least test error. It seems that this better accuracy is due to its greater representational power rather than its computational power (BN's representational cost is much higher than the others as opposed to its computational cost).

2.5 Per-layer Precision Analysis

Next, we employ the DNN precision analysis framework above to provide a theoretical basis and an analytical method for *per-layer precision assignment* in DNNs. The proposed method can be employed by DNN designers to minimize overall precision without having to resort to expensive trial-and-error simulation-based approaches that are prevalent today. We show that per-layer minimum precision of input to output layers varies from 7 bit to 2 bit and from 11 bit to 2 bit for networks processing the MNIST [88] and CIFAR-10 [65] datasets, respectively. Therefore, per layer precision assignment leads to much greater savings in complexity compared to a uniform assignment of layer precisions. Indeed, for the MNIST and CIFAR-10 datasets, and for the same level of accuracy, we show up to 4 bit reduction in minimum precision compared to the coarse-grained approach of [89] and up to 8 bit reduction compared to a naive uniform assignment [72, 21]. Moreover, we achieve same accuracy but $\sim 3.5\times$ less complexity than a state-of-the-art binary network, BinaryNet [51]. Compared to a state-of-the-art fixed-point network [72], the

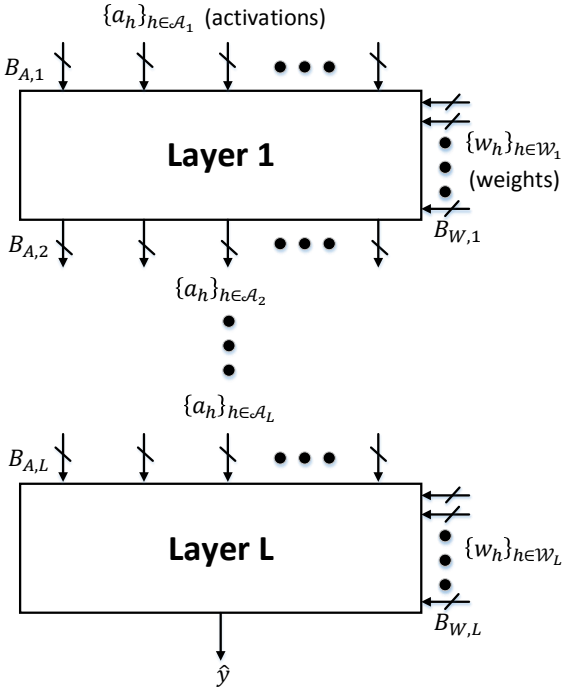


Figure 2.3: Per-layer precision assignments for a feedforward neural network architecture. Each layer performs either a matrix vector multiplication, or a set of 2D convolutions, followed by a non-linear activation function. The precision assignments for layer l are $B_{A,l}$ and $B_{W,l}$ for activations and weights, respectively.

complexity savings are even higher (up to $\sim 14\times$) in spite of better accuracy.

2.5.1 Fine-grained Precision Analysis

For a given neural network with L layers, let $\{\mathcal{A}_l\}_{l=1}^{L}$ $\{\mathcal{W}_l\}_{l=1}^{L}$ be the layer-wise partitions of $\mathcal{A}$ and $\mathcal{W}$, respectively. Consequently, as shown in Figure 2.3, if the per-layer precisions are $\{B_{A,l}\}_{l=1}^{L}$ and $\{B_{W,l}\}_{l=1}^{L}$ for activations and weights, respectively, then (2.8) can be re-written as:

$$p_m \leq \sum_{l=1}^{L} \left(\Delta_{A,l}^2 E_{A,l} + \Delta_{W,l}^2 E_{W,l}\right), \tag{2.15}$$

where $\Delta_{A,l} = 2^{-(B_{A,l}-1)}$ and $\Delta_{W,l} = 2^{-(B_{W,l}-1)}$ are the activation and weight quantization step-sizes at layer l, respectively, and

$$E_{A,l} = \mathbb{E}\left[\sum_{\substack{i=1 \\ i\neq\hat{Y}_{fl}}}^{M} \frac{\sum\limits_{h\in\mathcal{A}_l}\left|\frac{\partial(Z_i - Z_{\hat{Y}_{fl}})}{\partial A_h}\right|^2}{24|Z_i - Z_{\hat{Y}_{fl}}|^2}\right]$$

and

$$E_{W,l} = \mathbb{E}\left[\sum_{\substack{i=1 \\ i\neq\hat{Y}_{fl}}}^{M} \frac{\sum\limits_{h\in\mathcal{W}_l}\left|\frac{\partial(Z_i - Z_{\hat{Y}_{fl}})}{\partial w_h}\right|^2}{24|Z_i - Z_{\hat{Y}_{fl}}|^2}\right]$$

are the activation and weight quantization noise gains at layer l, respectively.

Observe that (2.15) is a sum of $2L$ terms where the quantization noise gains are computed only once after training. The design parameters are the $2L$ precision assignments, $\{B_{A,l}\}_{l=1}^{L}$ and $\{B_{W,l}\}_{l=1}^{L}$. Once again, a sum of independent terms is to be balanced. To do so, the minimum quantization noise gain is first computed:

$$E_{\min} = \min\left(\{E_{A,l}\}_{l=1}^{L}, \{E_{W,l}\}_{l=1}^{L}\right). \tag{2.16}$$

Then, a reference minimum precision $B_{\min}$ is chosen, and for each layer l, similar to (2.12), the precision is set as follows:

$$B_{A,l} = round\left(\log_2\left(\sqrt{\frac{E_{A,l}}{E_{\min}}}\right)\right) + B_{\min} \tag{2.17}$$

and

$$B_{W,l} = round\left(\log_2\left(\sqrt{\frac{E_{W,l}}{E_{\min}}}\right)\right) + B_{\min}. \tag{2.18}$$

Note that at least one of the 2L precision assignments will equal $B_{\min}$.

Once again, (2.15)-(2.18) can be used to efficiently find the *fine-grained* minimum precision requirements of a network. Here, (2.16), (2.17), and (2.18) are used to reduce the search space from a $2L$ dimensional grid to just a one dimensional axis. Afterwards, (2.15) is used to provide an initial estimate of precision requirements. Note that the search space reduction is

massive. For instance, for a 5 layer network, if we are considering precisions up to 16 bit, the search space is reduced from 16^{10} to only 16 design points.

2.5.2 Complexity in Fixed-point

In order to quantify the benefits of the proposed precision reduction method, we shall consider two measures of complexity [89]: *computational* and *representational* costs.

The computational cost is the total number of full adders needed per decision. Note that each layer implements an ensemble of dot products in order to realize either a matrix vector multiplication or a set of 2D convolutions. Hence, the computational cost (measured in 1 bit full adders or FAs) of a network is [89]:

$$\sum_{l=1}^{L} \left[N_l \left(D_l B_{A,l} B_{W,l} + (D_l - 1)(B_{A,l} + B_{W,l} + \lceil \log_2(D_l) \rceil - 1) \right) \right],$$

where N_l and D_l are the number and dimensionality of dot products computed at layer l, respectively.

The representational cost is the total number of bits needed to represent both weights and activations, and is given by:

$$\sum_{l=1}^{L} \left(|\mathcal{A}_l| \, B_{A,l} + |\mathcal{W}_l| \, B_{W,l} \right).$$

2.6 Numerical Results - Small Models and Datasets

Numerical experiments on two datasets are considered: the MNIST dataset for character recognition [88], and the CIFAR-10 dataset for object recognition [65]. For each, a neural network is first pre-trained as follows:

- MNIST: A multi-layer perceptron with architecture $784 - 512 - 512 - 512 - 10$. The network is pre-trained using Vanilla SGD [90] and has a baseline test error of 1.10% in floating-point.
- CIFAR-10: A convolutional neural network with architecture $32C3 -$

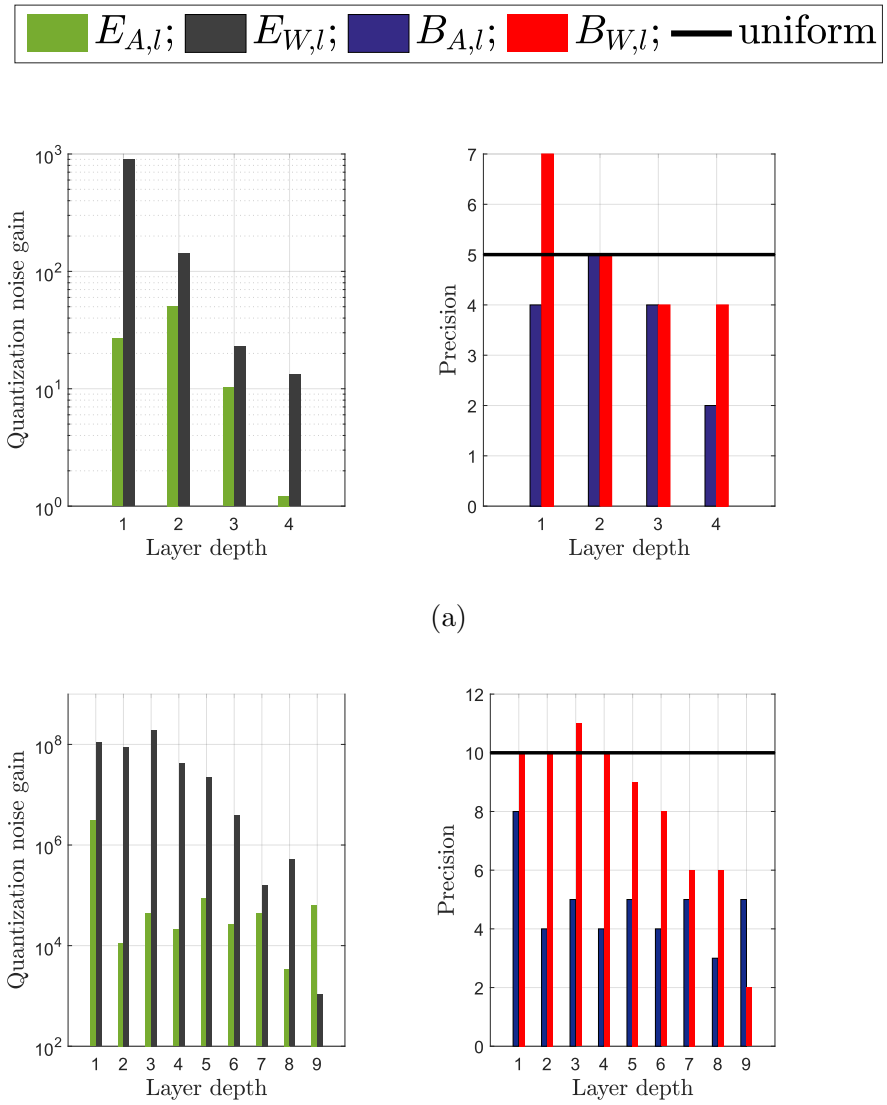


Figure 2.4: Plots showing quantization noise gains and per-layer precision assignments for (a) MNIST and (b) CIFAR-10, to satisfy $p_m \leq 1\%$. The minimum precision to satisfy $p_m \leq 1\%$ with uniform precision assignment is also shown.

$32C3 - MP2 - 64C3 - 64C3 - MP2 - 128C3 - 128C3 - 256FC - 256FC - 10$. The network is pre-trained is trained using Vanilla Adam [91] and has a baseline test error of 11.87% in floating-point.

Note that the architectures described above are inspired from those used by BinaryNet [51] and are actually obtained by reducing the height of each layer

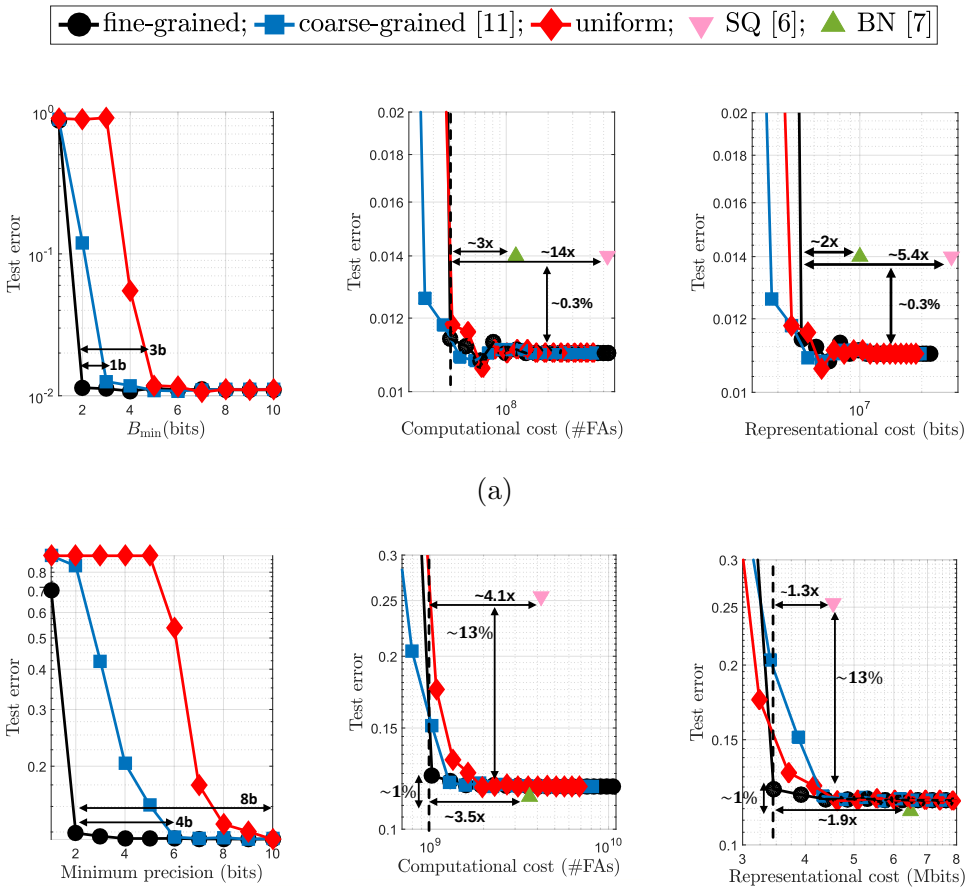


Figure 2.5: Test error vs. minimum precision ($B_{\min}$), computational and representational costs, for: (a) MNIST and (b) CIFAR-10 networks. A comparison with SQ [72] and BN [51] is also included.

by a factor of 4.

Each network is then quantized to fixed-point using three precision assignment methods:

- The proposed *fine-grained* precision assignment.
- A *coarse-grained* precision assignment [89].
- A *uniform* or identical precision assignment for all activations and weights.

The obtained results are compared to those reported by two state-of-the-art works on reduced precision neural networks:

- Stochastic quantization (SQ) [72] - trained fixed-point networks.

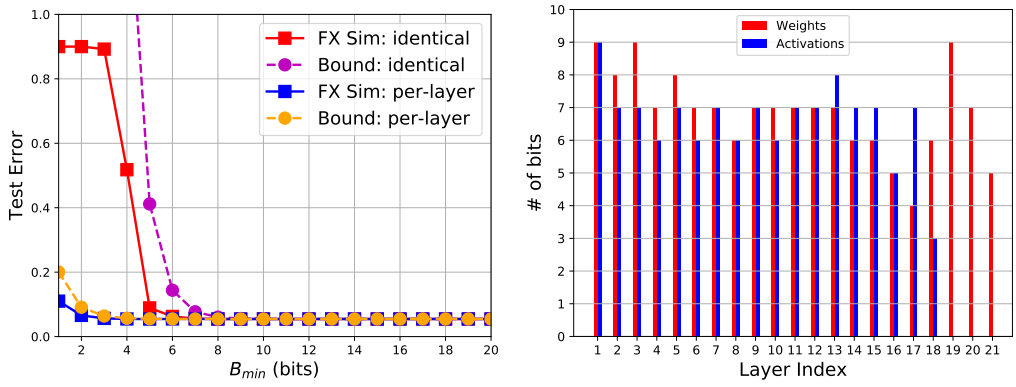
- BinaryNet (BN) [51] - trained binarized networks.

2.6.1 Results

Figure 2.4 shows the benefits of the proposed approach (2.15)-(2.18) compared to a naive uniform precision assignment. Indeed, to satisfy a mismatch probability of $p_m \leq 1\%$, most precisions are less than the required uniform precision. Furthermore, observe that the precision assignment matches the quantization noise gain profile on a logarithmic scale. This is due to the use of (2.16), (2.17), and (2.18). In addition, a general trend of decreasing precision requirements with layer depth is noticed. This is in accordance with recent findings demonstrating that perturbations at the early layers of neural networks are often the most destructive [92]. Here, the proposed method is naturally countering this effect by assigning more precision to the lower levels. Similarly, it is seen that the precision assignments of weights are typically more than those of activations, confirming the findings of [89].

Figure 2.5 shows that the coarse-grained approach [89] achieves a good initial improvement over the naive uniform assignment, and is able to reduce the minimum precision by up to 4 bit before the accuracy starts to degrade. However, the proposed fine-grained method is noticeably superior and able to reduce the minimum precision to just 2 bit without any notable accuracy degradation for both networks. This corresponds to 4 bit less than the minimum precision obtained via the coarse-grained method for the CIFAR-10 network.

As far as complexity is concerned, the results obtained are superior to those of SQ. For instance, on the CIFAR-10 dataset, the test error obtained via the fine-grained precision assignment is $\sim 13\%$ less than that reported by SQ in spite of requiring $\sim 4\times$ lower computational cost. Moreover, the complexity savings of BN (binarized) are surpassed. Indeed, even though the reported levels of accuracy are very close to those obtained via the proposed method, the latter achieves $\sim 3\times$ and $\sim 3.5\times$ less computational cost for the MNIST and CIFAR-10 networks, respectively, over the fully binarized BN. Similar trends also hold for the representational costs. These results reflect the importance of depth vs. width vs. precision considerations when exploring reduced complexity neural networks.



(a) (b)

Figure 2.6: Precision analysis for ResNet-18 on CIFAR-10: (a) Test error vs. $B_{\min}$ for an identical weight/activation precision assignment for all layers and a per-layer precision assignment based on the noise equalization method, and (b) per-layer weight and activation precisions required to achieve $p_m < 1\%$. 'FX Sim' denotes FX simulations and 'Bound' denotes evaluation of the upper bound in Theorem 1. Layer indices 19, 20, and 21 correspond to the weight skip connection layers.

2.7 Numerical Results - Large Models and Datasets

Next, we apply our analyses and methods to significantly more complex networks and larger datasets.

2.7.1 ResNet-18 on CIFAR-10

We deploy the ResNet-18 model [3] on the CIFAR-10 dataset. This network is much more complex than all previously considered models. Indeed, it has a depth of 18 feedforward layers and 3 weighted skip connection layers. The model is pre-trained using the SGD algorithm and the floating-point accuracy achieved is 94.53%.

In Figure 2.6 (a), we plot the fixed-point test error as a function of precision when our analysis is applied. It is found that, for an identical precision assignment for all activations and weights, a precision of 6 bit is required to achieve $p_m \leq 1\%$. The corresponding fixed-point accuracy is 93.74%. Our bound in Theorem 1 predicts that a precision of 8 bit yields $p_m \leq 0.56\%$.

When the method of per-layer precision assignment is employed, a minimum precision $B_{\min}$ of 3 bit suffices to achieve $p_m \leq 1\%$. In this case, the

fixed-point accuracy is of 94.35%. Furthermore, our bound in Theorem 1 predicts $p_m \leq 0.91\%$ for this precision assignment.

The above results solidify two aspects of our work: 1) our derived theoretical bounds on the fixed-point accuracy are tight up to a 2 bit conservative estimate of actual precision requirements, and 2) the use of noise equalization leads to further precision reduction compared to the best uniform assignment with no loss of accuracy.

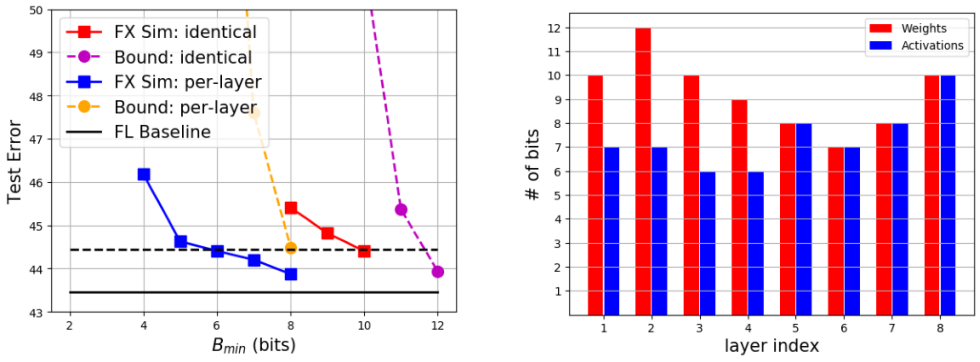
Furthermore, we note that when precision is lowered below the aforementioned values of $B_{\min}$, the test error increases exponentially. We observe an approximate quadrupling of mismatch rate when precision is decremented. This is in accordance with our theoretical results and can be important in the applications where accuracy requirements are tunable.

In Figure 2.6 (b), we plot the per-layer precisions required to achieve $p_m \leq 1\%$ when the method of noise equalization is used. It is found that higher precision is typically required for early layers as opposed to layers closer to the output. Additionally, weight precision requirements are found to be highest. Such findings are consistent with our aforementioned results on smaller networks.

Related works on quantization-aware training have shown that accuracy can be maintained for such networks using a 4 bit uniform precision assignment [53, 58]. These works rely on ad-hoc optimization techniques and provide no accuracy guarantees. Our results indicate precision values ranging from 3 to 9 bit in order to guarantee accuracy. Thus, we claim that the price to pay for such guarantees is up to 5 additional bits in the fixed-point network precision assignment.

2.7.2 AlexNet on ImageNet

Next, we demonstrate the applicability of our work using the AlexNet network [1] deployed on the ImageNet dataset [2]. The network structure is of 8 feedforward layers, 5 of which are convolutional and 3 are fully connected. Overall, it requires storage of over 60 million parameters and 800 million multiply-accumulate operation to process a single input [21]. The ImageNet dataset is significantly larger than any other dataset used thus far. It has over a million images for training and 50,000 images for validation/testing. These



(a) (b)

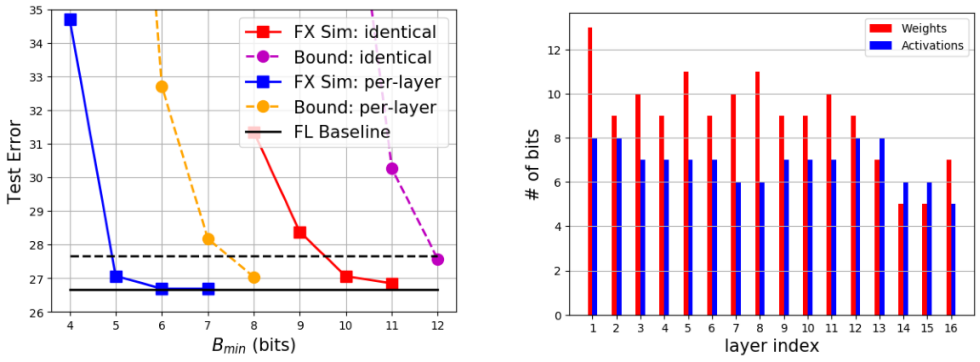
Figure 2.7: Precision analysis for AlexNet on ImageNet: (a) Test error vs. B_{min} for an identical weight/activation precision assignment for all layers and a per-layer precision assignment based on the noise equalization method, and (b) per-layer weight and activation precisions required to achieve $p_m < 1\%$. 'FX Sim' denotes FX simulations and 'Bound' denotes evaluation of the upper bound in Theorem 1. The black solid and dashed lines correspond to the floating-point test error and its 1% increase threshold, respectively.

natural images are organized into 1,000 classes. The pre-trained floating-point network is downloaded from the Pytorch repository [93] and its Top-1 test error is of 43.45%.

In Figure 2.7 (a), we plot the fixed-point test error as a function of precision when our analysis is applied. First, for an identical weight/activation precision assignment, it is found that a precision of 10 bit is required to achieve $p_m \leq 1\%$. The corresponding fixed-point test error is of 44.40%. Using the bound in Theorem 1, we predict this accuracy is achieved for a 12 bit precision assignment. Thus, in this case, the looseness of the bound is found to be 2 bit.

In contrast, when the method of noise equalization is used to obtain a per-layer precision assignment, the same accuracy can achieved while lowering B_{min} to 6 bit. In this case the fixed-point test error is of 44.41%. Hence, we once more demonstrate how our methods can be used to lower precision by up to 4 bit while maintaining accuracy. Furthermore, using the bound in Theorem 1, it is found that a $B_{\text{min}} = 8$ is required to achieve the desired accuracy. Hence, we once again find the looseness of the bound to be of 2 bit.

Figure 2.7 (a) also indicates that precision reduction leads to $\sim 3\times$ increase



(a) (b)

Figure 2.8: Precision analysis for VGG-16 on ImageNet: (a) Test error vs. B_{min} for an identical weight/activation precision assignment for all layers and a per-layer precision assignment based on the noise equalization method, and (b) per-layer weight and activation precisions required to achieve $p_m < 1\%$. 'FX Sim' denotes FX simulations and 'Bound' denotes evaluation of the upper bound in Theorem 1. The black solid and dashed lines correspond to the floating-point test error and its 1% increase threshold, respectively.

in test error. Our bounds predict a $4\times$ increase in p_m when precision is decremented. Hence, our theoretical results are found to be slightly conservative compared to actual empirical findings. Nonetheless, the trends are similar and can be important in applications having tunable precision requirements.

In Figure 2.7 (b), we plot the per-layer precisions required to achieve $p_m \leq 1\%$ when the method of noise equalization is used. It is found that weights require more precision than activations and that highest requirements are attributed to the early layers. These findings are consistent with our earlier results on small networks and datasets.

2.7.3 VGG-16 on ImageNet

Next, we demonstrate the applicability of our work using the VGG-16 network [39] deployed on the ImageNet dataset. The VGG-16 network is even larger and more complex than AlexNet. It consists of 16 feedforward layers, 13 of which are convolutional and 3 are fully connected. It has approximately 138 million parameters and requires 15 billion multiply-accumulate operations to process one input. This network is known to boast one of the best accuracies on the ImageNet dataset. The pre-trained floating-point net-

work is downloaded from the Pytorch repository and its Top-1 test error is 26.66%.

In Figure 2.8 (a), we plot the fixed-point test error as a function of precision when our analysis is applied. First, for an identical weight/activation precision assignment across all layers, it is found that 10 bit are required to achieve $p_m \leq 1\%$. The corresponding fixed-point test error is of 27.06%. Using the bound in Theorem 1, we predict this accuracy is achieved for a 12 bit precision assignment. Thus, in this case, the looseness of the bound is found to be of 2 bit.

When the method of noise equalization is applied, the same accuracy can be achieved while reducing the precision by up to 5 bit. In this, it is found that when $B_{\min}$ is equal to 5 bit, the fixed-point test error is equal to 27.05%. This result is further evidence that our methods allow for significant precision reduction with no loss in accuracy. Furthermore, using the bound in Theorem 1, it is found that a $B_{\min} = 8$ is required to achieve the desired accuracy. Hence, this time, the looseness of the bound is found to be 3 bit.

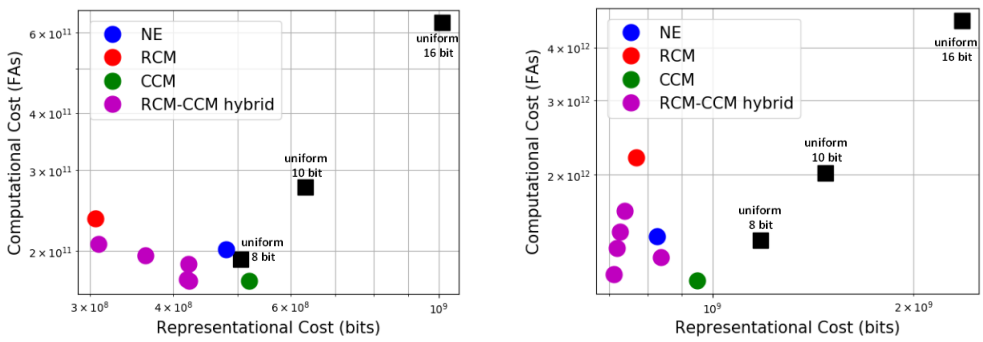
Figure 2.8 (a) also indicates that precision reduction leads to $\sim 4\times$ increase in test error which corroborates our theoretical results. Such trends are important in designing fixed-point networks with tunable accuracy requirements to be exploited by the application.

In Figure 2.8 (b), we plot the per-layer precision required to achieve $p_m \leq 1\%$. Our findings are consistent with all previously reported results. Weights usually require most precision, particularly in the early layers.

2.7.4 Accuracy-aware Complexity Reduction

Finally, we demonstrate how our precision analysis above leads to accuracy-aware complexity reduction. We use the AlexNet and VGG-16 networks discussed above and study their computational vs. representational cost trade-offs. To do so, we select, for both, design points that iso-accurate, i.e., having $p_m \leq 1\%$. These designs are:

- Uniform precision assignment of 16 bit across all activation and weight layers.
- Minimum uniform precision assignment to achieve $p_m \leq 1\%$.



(a) (b)

Figure 2.9: Computational vs. Representational cost trade-off for various precision assignments for: (a) AlexNet and (b) VGG-16. NE, RCM, and CCM correspond to noise equalization, representational cost minimizer, and computational cost minimizer, respectively. All design points are iso-accurate, i.e., have $p_m \leq 1\%$, except for the 8 bit uniform one, which for both networks has $p_m \geq 2\%$.

- Precision assignment obtained via noise equalization (NE) such that $p_m \leq 1\%$.
- Representational Cost Minimizer (RCM) obtained by explicitly minimizing the representational cost in (2.4) subject to $p_m \leq 1\%$.
- Computational Cost Minimizer (CCM) obtained by explicitly minimizing the computational cost in (2.3) subject to $p_m \leq 1\%$.
- RCM-CCM hybrid corresponding to solutions to the minimization of linear combinations of computational and representational costs subject to $p_m \leq 1\%$.

Note that the problem of minimizing the complexity costs subject to the accuracy constraint can be solved efficiently using a grid search. This is because the Hessian matrix of the associated problem is always block diagonal. The 16 bit uniform design point is chosen representative of common practices [21]. We also consider the popular choice of 8 bit often used by ad-hoc designers.

In Figure 2.9, we plot representational vs. computational costs for the above design points applied to both our AlexNet and ImageNet networks.

The following insights are derived. The common choice of uniform 16 bit is an overly conservative one since 10 bit suffice to achieve the same accuracy. This difference of precision is equivalent to $\sim 40\%$ and $\sim 50\%$ reduction in representational and computational costs, respectively.

Furthermore, NE yields an additional $\sim 30\%$ and $\sim 50\%$ in representational and computational costs, respectively, compared to the best uniform precision assignment. This constitutes further evidence as to the importance of our analysis.

Interestingly, NE is found to exhibit an excellent representational vs. computational cost trade-off. It is found that CCM only reduces the computational cost by $\sim 10\%$ compared to NE for both networks. On the other hand, RCM only yields a $\sim 5\%$ decrease in representational cost in the case of the VGG-16 network.

When a linear combination of both costs is minimized (RCM-CCM hybrid), a family of pareto-optimal design points is obtained. These points optimize the trade-off between the two costs. We do observe that NE lies in close proximity to this family on the representational vs. computational cost grid. This proves that NE is a nearly optimal accuracy-aware complexity reduction technique

Finally, we make an interesting observation regarding the commonly employed 8 bit uniform precision assignment. The corresponding design point, which fails to maintain accuracy ($p_m \geq 2\%$), has higher complexity than NE, RCM, CCM, and their linear combinations. This is an important result indicating how ad-hoc designs can be both inaccurate and sub-optimal. It is thus of utmost importance to design reduced-complexity networks in an accuracy-aware manner, using the methods and analyses derived in our work.

2.8 Summary

In this chapter we analyzed the quantization tolerance of neural networks. We used our analysis to efficiently reduce weight and activation precisions while maintaining fidelity similar to that of the floating-point initiation. Specifically, we obtained bounds on the mismatch probability between a fixed-point network and its floating-point counterpart in terms of precision.

We showed that a neural network's accuracy degradation due to quantization decreases double exponentially as a function of precision. Our analysis provides a straightforward method to obtain an upper bound on the network's error probability as a function of precision. We used these results on real datasets to minimize the computational and representational costs of a fixed-point network while maintaining accuracy.

Additionally, we have presented an analytical approach to fine-grained precision assignment in deep neural networks (DNNs). The benefits of the proposed approach in terms of minimum precision and complexity reduction have been shown. A performance comparison with state-of-the-art binary and fixed-point neural networks was illustrated and highlighted considerable savings. The presented work allows DNN designers to determine minimum precision requirements in DNNs and estimate their complexities without needing to run lengthy simulations. The proposed method can be employed to efficiently explore other dimensions in the design of low-complexity DNNs such as the trade-off between precision vs. depth vs. width, and between precision and pruning.

Our work addresses the general problem of resource-constrained machine learning. One takeaway is that it is imperative to understand the trade-offs between accuracy and complexity. In our work, we used precision as a parameter to analytically characterize this trade-off. Nevertheless, additional aspects of complexity in neural networks such as their structure and their sparsity can also be accounted for. In fact, more work can be done in that regard. Our work may be viewed as a first step in developing a unified and principled framework to understand complexity vs. accuracy trade-offs in deep neural networks and other machine learning algorithms.

2.9 Addendum: Proofs of Bounds on Mismatch Probability

The main purpose of this section is to provide proofs for Theorems 1 and 2.

Preliminaries

Here we shall give a proof of (2.1) as well as preliminary results that will be needed to complete the proofs of Theorems 1 and 2.

Proposition 1. *The fixed point error probability $p_{e,fx}$ is upper bounded as shown in* (2.1).

Proof. From the definitions of $p_{e,fx}$, $p_{e,fl}$, and p_m,

$$\begin{aligned} p_{e,fx} &= \Pr\{\hat{Y}_{fx} \neq Y\} \\ &= \Pr\{\hat{Y}_{fx} \neq Y, \hat{Y}_{fx} = \hat{Y}_{fl}\} + \Pr\{\hat{Y}_{fx} \neq Y, \hat{Y}_{fx} \neq \hat{Y}_{fl}\} \\ &= \Pr\{\hat{Y}_{fl} \neq Y, \hat{Y}_{fx} = \hat{Y}_{fl}\} + \Pr\{\hat{Y}_{fx} \neq Y, \hat{Y}_{fx} \neq \hat{Y}_{fl}\} \\ &\leq p_{e,fl} + p_m. \end{aligned}$$

□

Next is a simple result that allows us to replace the problem of upper bounding p_m by several smaller and easier problems by virtue of the union bound.

Proposition 2. *In an M-class classification problem, the total mismatch probability can be upper bounded as follows:*

$$p_m \leq \sum_{j=1}^{M} \sum_{i=1, i\neq j}^{M} \Pr(\hat{Y}_{fx} = i | \hat{Y}_{fl} = j) \Pr(\hat{Y}_{fl} = j). \tag{2.19}$$

Proof.

$$
\begin{aligned}
p_m = \Pr(\hat{Y}_{fx} \neq \hat{Y}_{fl}) &= \Pr\left(\bigcup_{j=1}^{M} (\hat{Y}_{fx} \neq j, \hat{Y}_{fl} = j)\right) \\
&\leq \sum_{j=1}^{M} \Pr(\hat{Y}_{fx} \neq j, \hat{Y}_{fl} = j) \\
&= \sum_{j=1}^{M} \Pr(\hat{Y}_{fx} \neq j | \hat{Y}_{fl} = j) \Pr(\hat{Y}_{fl} = j) \\
&= \sum_{j=1}^{M} \Pr\left(\bigcup_{i=1, i\neq j}^{M} \hat{Y}_{fx} = i \middle| \hat{Y}_{fl} = j\right) \Pr(\hat{Y}_{fl} = j) \\
&\leq \sum_{j=1}^{M} \sum_{i=1, i\neq j}^{M} \Pr(\hat{Y}_{fx} = i | \hat{Y}_{fl} = j) \Pr(\hat{Y}_{fl} = j),
\end{aligned}
$$

where both inequalities are due to the union bound. □

The next result is also straightforward, but quite useful in obtaining upper bounds that are fully determined by averages.

Proposition 3. *Given a random variable X and an event $\mathcal{E}$, we have:*

$$
\mathbb{E}\left[X \cdot \mathbb{1}_{\mathcal{E}}\right] = \mathbb{E}\left[X | \mathcal{E}\right] \Pr(\mathcal{E}), \tag{2.20}
$$

where $\mathbb{1}_{\mathcal{E}}$ denotes the indicator function of the event $\mathcal{E}$.

Proof. By the law of total expectation,

$$
\begin{aligned}
\mathbb{E}\left[X \cdot \mathbb{1}_{\mathcal{E}}\right] &= \mathbb{E}\left[X \cdot \mathbb{1}_{\mathcal{E}} \mid \mathcal{E}\right] \Pr(\mathcal{E}) + \mathbb{E}\left[X \cdot \mathbb{1}_{\mathcal{E}} \mid \mathcal{E}^c\right] \Pr(\mathcal{E}^c) \\
&= \mathbb{E}\left[X \cdot 1 \mid \mathcal{E}\right] \Pr(\mathcal{E}) + \mathbb{E}\left[X \cdot 0 \mid \mathcal{E}^c\right] \Pr(\mathcal{E}^c) \\
&= \mathbb{E}\left[X | \mathcal{E}\right] \Pr(\mathcal{E}).
\end{aligned}
$$

□

Proof of Theorem 1

Let us define $p_{m,j\to i}$ for $i \neq j$ as follows:

$$p_{m,j\to i} = \Pr\{\hat{Y}_{fx} = i \mid \hat{Y}_{fl} = j\}. \tag{2.21}$$

We first prove the following Lemma.

Lemma 1. *Given B_X and B_F, if the output of the floating-point network is $\hat{Y}_{fl} = j$, then that of the fixed-point network would be $\hat{Y}_{fx} = i$ with a probability upper bounded as follows:*

$$\begin{aligned} p_{m,j\to i} \leq \frac{\Delta_A^2}{24}\mathbb{E}\left[\frac{\sum_{h\in\mathcal{A}}\left|\frac{\partial(Z_i-Z_j)}{\partial A_h}\right|^2}{|Z_i-Z_j|^2}\middle|\hat{Y}_{fl}=j\right] \\ + \frac{\Delta_W^2}{24}\mathbb{E}\left[\frac{\sum_{h\in\mathcal{W}}\left|\frac{\partial(Z_i-Z_j)}{\partial w_h}\right|^2}{|Z_i-Z_j|^2}\middle|\hat{Y}_{fl}=j\right]. \end{aligned} \tag{2.22}$$

Proof. We can claim that, if $i \neq j$:

$$p_{m,j\to i} \leq \Pr\{Z_i + q_{Z_i} > Z_j + q_{Z_j} \mid \hat{Y}_{fl} = j\}, \tag{2.23}$$

where the equality holds for $M = 2$.

From the law of total probability,

$$p_{m,j\to i} \leq \int f_{\mathbf{X}}(\mathbf{x}) \Pr\left(z_i + q_{z_i} > z_j + q_{z_j} \mid \hat{Y}_{fl} = j, \mathbf{x}\right) d\mathbf{x}, \tag{2.24}$$

where $\mathbf{x}$ denotes the input of the network, or equivalently an element from the dataset and $f_{\mathbf{X}}()$ is the distribution of the input data. But for one specific $\mathbf{x}$ given $\hat{Y}_{fl} = j$, we have:

$$\Pr\left(z_i + q_{z_i} > z_j + q_{z_j}\right) = \frac{1}{2}\Pr\left(\left|q_{z_i} - q_{z_j}\right| > |z_j - z_i|\right),$$

where the $\frac{1}{2}$ term is due to the symmetry of the distribution of the quantization noise around zero per output. By (2.7), we can claim that

$$q_{z_i} - q_{z_j} = \sum_{h\in\mathcal{A}} q_{a_h}\frac{\partial(z_i - z_j)}{\partial a_h} + \sum_{h\in\mathcal{W}} q_{w_h}\frac{\partial(z_i - z_j)}{\partial w_h}. \tag{2.25}$$

Note that $q_{z_i} - q_{z_j}$ is a zero mean random variable with the following variance:

$$\frac{\Delta_A^2}{12} \sum_{h \in \mathcal{A}} \left| \frac{\partial (z_i - z_j)}{\partial a_h} \right|^2 + \frac{\Delta_W^2}{12} \sum_{h \in \mathcal{W}} \left| \frac{\partial (z_i - z_j)}{\partial w_h} \right|^2 .$$

By Chebyshev's inequality, we obtain

$$\Pr\left(z_i + q_{z_i} > z_j + q_{z_j}\right) \leq \frac{\Delta_A^2 \sum_{h \in \mathcal{A}} \left| \frac{\partial (z_i - z_j)}{\partial a_h} \right|^2 + \Delta_W^2 \sum_{h \in \mathcal{W}} \left| \frac{\partial (z_i - z_j)}{\partial w_h} \right|^2}{24 \left| z_i - z_j \right|^2}. \tag{2.26}$$

From (2.24) and (2.26), we can derive (2.22). □

Substituting (2.22) of Lemma 1 into (2.19) and using (2.20),

$$\begin{aligned} p_m \leq \sum_{j=1}^{M} \sum_{i=1, i \neq j}^{M} \Bigg(& \frac{\Delta_A^2}{24} \mathbb{E}\left[\frac{\sum_{h \in \mathcal{A}} \left| \frac{\partial (Z_i - Z_j)}{\partial A_h} \right|^2}{|Z_i - Z_j|^2} \mathbb{1}_{\hat{Y}_{fl} = j} \right] \\ & + \frac{\Delta_W^2}{24} \mathbb{E}\left[\frac{\sum_{h \in \mathcal{W}} \left| \frac{\partial (Z_i - Z_j)}{\partial w_h} \right|^2}{|Z_i - Z_j|^2} \mathbb{1}_{\hat{Y}_{fl} = j} \right] \Bigg), \end{aligned} \tag{2.27}$$

which can be simplified into (2.8) in Theorem 1.

Proof of Theorem 2

We start with the following lemma.

Lemma 2. *Given B_A and B_W, $p_{m,j \to i}$ is upper bounded as follows:*

$$p_{m,j \to i} \leq \mathbb{E}\left[e^{-T \cdot V} \prod_{h \in \mathcal{A}} \frac{\sinh\left(T \cdot D_{A,h}\right)}{T \cdot D_{A,h}} \cdot \prod_{h \in \mathcal{W}} \frac{\sinh\left(T \cdot D_{W,h}\right)}{T \cdot D_{W,h}} \middle| \hat{Y}_{fl} = j \right], \tag{2.28}$$

where $T = \frac{3V}{\sum_{h \in \mathcal{A}} \Delta_{A,h}^2 + \sum_{h \in \mathcal{W}} \Delta_{W,h}^2}$, $V = Z_j - Z_i$, $D_{A,h} = \frac{\Delta_A}{2} \cdot \frac{\partial (Z_i - Z_j)}{\partial A_h}$, and $D_{W,h} = \frac{\Delta_W}{2} \cdot \frac{\partial (Z_i - Z_j)}{\partial W_h}$.

Proof. The setup is similar to that of Lemma 1. Denote $v = z_j - z_i$. By the

Chernoff bound,

$$\Pr\left(q_{z_i} - q_{z_j} > v\right) \leq e^{-tv}\mathbb{E}\left[e^{t(q_{z_i}-q_{z_j})}\right]$$

for any $t > 0$. Because quantization noise terms are independent, by (2.25),

$$\mathbb{E}\left[e^{t(q_{z_i}-q_{z_j})}\right] = \prod_{h\in\mathcal{A}}\mathbb{E}\left[e^{tq_{a_h}d'_{a_h}}\right]\prod_{h\in\mathcal{W}}\mathbb{E}\left[e^{tq_{w_h}d'_{w_h}}\right],$$

where $d'_{a_h} = \frac{\partial(z_i-z_j)}{\partial a_h}$ and $d'_{w_h} = \frac{\partial(z_i-z_j)}{\partial w_h}$. Also, $\mathbb{E}\left[e^{tq_{a_h}d'_{a_h}}\right]$ is given by

$$\mathbb{E}\left[e^{tq_{a_h}d'_{a_h}}\right] = \frac{1}{\Delta_A}\int_{-\frac{\Delta_A}{2}}^{\frac{\Delta_A}{2}} e^{tq_{a_h}d'_{a_h}}dq_{a_h} = \frac{2}{td'_{a_h}\Delta_A}\sinh\left(\frac{td'_{a_h}\Delta_A}{2}\right) = \frac{\sinh\left(td_{a_h}\right)}{td_{a_h}},$$

where $d_{a_h} = \frac{d'_{a_h}\Delta_A}{2}$. Similarly, $\mathbb{E}\left[e^{tq_{w_h}d'_{w_h}}\right] = \frac{\sinh\left(td_{w_h}\right)}{td_{w_h}}$ where $d_{w_h} = \frac{d'_{w_h}\Delta_W}{2}$.

Hence,

$$\Pr\left(q_{z_i} - q_{z_j} > v\right) \leq e^{-tv}\prod_{h\in\mathcal{A}}\frac{\sinh\left(td_{a,h}\right)}{td_{a,h}}\prod_{h\in\mathcal{W}}\frac{\sinh\left(td_{w,h}\right)}{td_{w,h}}. \tag{2.29}$$

By taking logarithms, the right-hand side is given by

$$-tv + \sum_{h\in\mathcal{A}}\big(\ln\sinh\left(td_{a,h}\right) - \ln\left(td_{a,h}\right)\big) + \sum_{h\in\mathcal{W}}\big(\ln\sinh\left(td_{w,h}\right) - \ln\left(td_{w,h}\right)\big).$$

This term corresponds to a linear function of t added to a sum of log-moment generating functions. It is hence convex in t. By taking the derivative with respective to t and setting to zero,

$$v + \frac{|\mathcal{A}| + |\mathcal{W}|}{t} = \sum_{h\in\mathcal{A}}\frac{d_{a,h}}{\tanh(td_{a,h})} + \sum_{h\in\mathcal{W}}\frac{d_{w,h}}{\tanh(td_{w,h})}.$$

But $\tanh(x) = x - \frac{1}{3}x^3 + \mathbf{o}\left(x^5\right)$, so dropping fifth-order terms yields:

$$v + \frac{|\mathcal{A}| + |\mathcal{W}|}{t} = \sum_{h\in\mathcal{A}}\frac{1}{t(1-\frac{(td_{a,h})^2}{3})} + \sum_{h\in\mathcal{W}}\frac{1}{t(1-\frac{(td_{w,h})^2}{3})}.$$

Note, for the terms inside the summations, we divided numerator and denominator by $d_{a,h}$ and $d_{w,h}$, respectively, then factored the denominator by

t. Now, me multiply both sides by t to get:

$$tv + |\mathcal{A}| + |\mathcal{W}| = \sum_{h \in \mathcal{A}} \frac{1}{1 - \frac{(td_{a,h})^2}{3}} + \sum_{h \in \mathcal{W}} \frac{1}{1 - \frac{(td_{w,h})^2}{3}}.$$

Also $\frac{1}{1-x^2} = 1 + x^2 + \mathbf{o}(x^4)$, so we drop fourth-order terms:

$$tv + |\mathcal{A}| + |\mathcal{W}| = \sum_{h \in \mathcal{A}} \left(1 + \frac{(td_{a,h})^2}{3}\right) + \sum_{h \in \mathcal{W}} \left(1 + \frac{(td_{w,h})^2}{3}\right)$$

which yields:

$$t = \frac{3v}{\sum_{h \in \mathcal{A}} (d_{a,h})^2 + \sum_{h \in \mathcal{W}} (d_{w,h})^2}. \tag{2.30}$$

Substituting (2.29) into (2.30) and using the similar method of Lemma 1, we can derive (2.28) of Lemma 2. □

Theorem 2 is obtained by substituting (2.28) of Lemma 2 into (2.19) and using (2.20). Of course, $D_{A_h}^{(i,j)}$ is the random variable of $d_{a,h}$ when $\hat{y}_{fx} = i$ and $\hat{y}_{fl} = j$, and the same applies to $D_{w_h}^{(i,j)}$ and $d_{w,h}$. We dropped the superscript (i, j) in the Lemma as it was not needed for the consistency of the definitions.

CHAPTER 3

MINIMIZING ADC PRECISION FOR INFERENCE ON IN-MEMORY ARCHITECTURE

The previous chapter investigated the precision requirements of weights and activations in a DNN realization. In this chapter, we study the precision requirements of DNN dot-products, when implemented using in-memory computing. Specifically, we propose the optimal clipping criterion for minimizing the column ADC precision. Applying bit slicing techniques, we prove that accurate dot-products can be realized with a single memory access. Finally, we demonstrate that, as a consequence of our work, significant energy reduction can be achieved while maintaining network level accuracy.

3.1 Motivation

In-memory computing (IMC) architecture [94, 95, 96] strives to eliminate the separation of storage and compute. It does so by realizing functional operations (such as dot-products) within the bitcell array (BCA) during memory accesses. In the process, the energy-delay product (EDP) of inference tasks can be reduced by up to two orders of magnitude compared to the conventional von Neumann architecture [28]. Since IMCs address the memory wall problem, it is particularly attractive for memory-centric workloads such as machine learning algorithms. In recent years, a large number of IMC implementations of DNNs have been proposed [27, 28, 29, 30, 31, 14, 32, 33, 34, 35, 36, 37, 38].

In spite of recent advances, IMCs face several challenges. First, functions mapped to IMCs have been restricted to simple (binary) operations and algorithms [76, 51, 52, 75]. Methods to increase IMC precision remain elusive. Second, mapping of high-dimensional dot-products onto IMCs is often limited by analog noise sources, which are not fully understood [97, 98]. Third, the energy and latency costs of column analog-digital converters (ADCs)

dominate [98] the system energy efficiency. Finally, the dense BCA layout imposes strict area and hence precision constraints on the ADCs [99]. Unfortunately, matching application-level accuracy requirements with such precision constraints on ADCs is made challenging.

Efforts to address some of the above mentioned limitations have relied on ad-hoc trial-and-error methods. The lack of an analytical framework to guide the design of IMCs has led to designs that are overly conservative and therefore sub-optimal in terms of efficiency. For example, the use of the bit-growth criterion (BGC) to set ADC precision [100] avoids loss in fidelity of bitline computations in the BCA much higher precision that necessary. Many IMC designs employ ad-hoc methods to assign fewer ADC bits than suggested by BGC and justify it via extensive simulations to show that DNN accuracy is preserved. However, such methods do not provide any theoretical guarantees.

Our work addresses the above mentioned limitations by making the following contributions:

- We propose the optimal clipping criterion (OCC) to minimize the column ADC precision requirements. The signal-to-quantization ratio (SQNR) of OCC is shown to be within 0.8 dB of the well-known optimal Lloyd-Max (LM) quantizer [101]. OCC improves the SQNR by 14 dB compared to the commonly employed full range (FR) quantizer which translates to a 3 bit reduction in ADC precision.

- We study the quantization noise in a input-serial weight-parallel (ISWP) IMC which generalizes the popular bit-serial weight-parallel IMC in [33]. We show that, using bit slicing techniques, significant energy savings can be achieved with minimal accuracy lost. Indeed, we prove that a dot-product can be achieved with a single memory access while suffering no more than 2 dB SQNR drop.

- We derive an analytical methodology for proper IMC design subject to DNN accuracy constraint. Combining the proposed OCC for reduced ADC precision and ISWP noise analysis, we demonstrate that per-layer energy consumption can be reduced by a factor of $\sim 6\times$.

This chapter is organized as follows: Our problem setup with the corresponding IMC model and architecture is introduced in Section 3.2. The op-

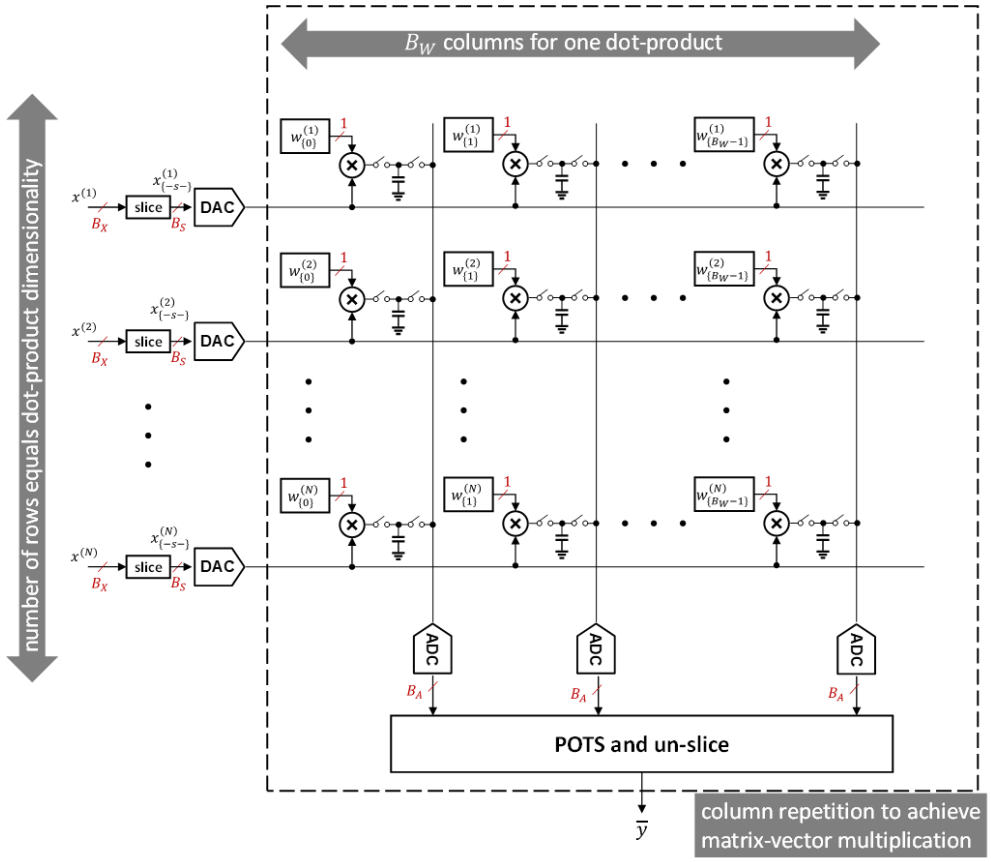


Figure 3.1: The input-serial weight-parallel (ISWP) architecture

timal clipping criterion is presented in Section 3.3. An analysis of bit slicing is included in Section 3.4. Section 3.5 contains our proposed optimization of DNN implementation using IMC. Numerical results are presented in Section 3.6. Finally, Section 3.7 summarizes this chapter.

3.2 Problem Setup

Consider an N-dimensional dot-product $y = \mathbf{w}^T\mathbf{x}$ of real valued (signed) weight and (unsigned) input vectors of precision B_W and B_X bit, respectively,

and given by:

$$\mathbf{w} = \begin{bmatrix} w^{(1)} \\ \vdots \\ w^{(N)} \end{bmatrix}; \quad w^{(i)} = w_m \left(-w_{\{0\}}^{(i)} + \sum_{b=1}^{B_W-1} w_{\{b\}}^{(i)} 2^{-b} \right)$$

$$\mathbf{x} = \begin{bmatrix} x^{(1)} \\ \vdots \\ x^{(N)} \end{bmatrix}; \quad x^{(i)} = x_m \sum_{b=0}^{B_X-1} x_{\{b\}}^{(i)} 2^{-b-1},$$

where $w_{\{b\}}^{(i)} \in \{0,1\}$ and $x_{\{b\}}^{(i)} \in \{0,1\}$ are the b^{th} bits of $w^{(i)} \in [-w_m, w_m]$ and $x^{(i)} \in [0, x_m]$, respectively. The choice of unsigned inputs is to account for the use of activations (e.g., ReLU) in DNNs.

3.2.1 The Input-Serial Weight-Parallel (ISWP) IMC

We consider the input-serial bit-parallel (ISBP) architecture (see Figure 3.1) [97] which generalizes the architecture in [33] by allowing for multi-bit inputs per read cycle.

The ISWP architecture stores $\mathbf{w}$ in the columns of the BCA where the B_W bit of $w^{(i)}$ are arrayed across B_W columns in the i^{th} row. During computation, ISWP serializes the B_X bit input vector $\mathbf{x}$ into $N_S = \left\lceil \frac{B_X}{B_S} \right\rceil$ input slices of B_S bit where

$$x^{(i)} = x_m \sum_{s=0}^{N_S-1} x_{\{-s-\}}^{(i)} 2^{-sB_s}$$

with $x_{\{-s-\}}^{(i)}$ being the s^{th} slice given by:

$$x_{\{-s-\}}^{(i)} = \sum_{b=0}^{B_S-1} x_{\{sB_S+b\}}^{(i)} 2^{-b-1}.$$

Processing the inputs at one slice per read cycle, the multi-bit dot-product $y = \mathbf{w}^T\mathbf{x}$ is realized using the following powers-of-two (POT) combination:

$$y = x_m w_m \sum_{s=0}^{N_S-1} \left(-y_{s,0} + \sum_{b=1}^{B_W-1} y_{s,b} 2^{-b} \right) 2^{-sB_S}, \tag{3.1}$$

Table 3.1: Values of analog noise parameters in a 65 nm process

Parameter	Value	Parameter	Value
ρ_1	6.40×10^{-18}F	ρ_3	6.01×10^{-33}F^2
ρ_2	4.14×10^{-21}F	V_{DD}	1V

where bitline (BL) dot-product $y_{s,b}$ is computed as:

$$y_{s,b} = \sum_{i=1}^{N} w_{\{b\}}^{(i)} x_{\{-s-\}}^{(i)}. \tag{3.2}$$

3.2.2 Quantization and Analog Noise Effects

The BL dot-product $y_{s,b}$ in (3.2) is computed using analog circuits. Due to noise, the observed BL dot-product $\overline{y}_{s,b}$ is given by:

$$\overline{y}_{s,b} = y_{s,b} + q_{A_{s,b}} + \eta_{a_{s,b}}, \tag{3.3}$$

where $q_{A_{s,b}}$ and $\eta_{a_{s,b}}$ are the column ADC quantization noise and BL analog noise, respectively. The expression for the variance of $q_{A_{s,b}}$ will be presented in Section 3.3 since it depends on the quantization method employed in the ADC.

The analog noise term $\eta_{a_{s,b}}$ includes capacitor mismatch, thermal effects, and charge injection. The variance of analog noise term $\eta_{a_{s,b}}$ [97] is given by:

$$\sigma^2_{\eta_{a_{s,b}}} = N \left(\frac{\mathbb{E}\left[(w_{\{\}} x_{\{-\}})^2\right] \rho_1}{(1 - 2^{-B_S})^2 C_o} + \frac{\rho_2}{C_o} + \frac{\rho_3}{C_o^2} \right), \tag{3.4}$$

where $w_{\{\}}$ and $x_{\{-\}}$ are unindexed weight bits and input slices, respectively, C_o is the nominal bitcell (BC) capacitance, and ρ_1, ρ_2, and ρ_3 are technology- and layout-dependent parameters. For a 65 nm process [97], the values of these parameters are listed in Table 3.1. The BC capacitance C_o is an important design parameter trading off complexity and accuracy. In (3.4), the noise variance decreases when C_o increases. However, large values of C_o cause higher energy consumption and reduce memory density.

The impact of the noise sources in (3.3) on the accuracy of the dot-product in (3.1) will be derived in Section 3.4.

3.2.3 Energy Consumption

An IMC's energy efficiency is quantified by the energy per 1 bit multiply-accumulate (MAC) operation E_{OP} [98]:

$$E_{OP} = \frac{N_S}{B_X}\left(E_{BC} + \frac{E_{ADC}}{N}\right), \tag{3.5}$$

where E_{BC} is the energy consumed by the 1 bit MAC within the bitcell given by

$$E_{BC} = \mathbb{E}\left[x_{\{-\}}\right] C_o V_{DD}^2, \tag{3.6}$$

where $\mathbb{E}\left[x_{\{-\}}\right]$ is the mean value of an input slice and V_{DD} is the supply voltage.

For a B_A bit ADC, E_{ADC} is given by [102, 103]:

$$E_{ADC} = k_1\left(B_A + \log_2\left(\frac{y_m}{R_Y}\right)\right) + k_2\left(\frac{y_m}{R_Y}\right)^2 4^{B_A}, \tag{3.7}$$

where $k_1 = 10^{-13}\,\mathrm{J}$ and $k_2 = 10^{-13}\,\mathrm{J}$ are empirical parameters, y_m is to the largest dot -product that can be accumulated on the bitline, and R_Y is the ADC quantization range. Equation (3.7) also indicates that the ADC energy quadruples per bit of increase in its precision B_A, emphasizing the need for minimizing it without impacting accuracy.

3.3 The Optimal Clipping Criterion

Quantization of a signal $x \in [x_{\min}, x_{\max}]$ to B bit is the process of mapping its value to one of 2^B pre-defined levels $\{x_{q_i}\}_{i=1}^{2^B}$. The quantized signal is obtained as $x_q = \arg\min_{\{x_{q_i}\}_{i=1}^{2^B}} |x - x_{q_i}|$. The choice of quantization levels sets the quantizer's mean-squared error (MSE), desired to be small, and defined as:

$$\mathrm{MSE}(x_q) = \mathbb{E}\left[(x - x_q)^2\right]. \tag{3.8}$$

Further, it is useful to employ an additive model of quantization noise [104, 105]. We write $x_q = x + q_x$, where q_x is a random variable assumed to be

zero-mean and independent from x. Its variance $\sigma_{q_x}^2$ is equal to the MSE in (3.8).

Given a signal distribution $f_x()$, the Lloyd-Max (LM) algorithm [101, 106, 107] finds a set of quantization levels $\{x_{q_i}\}_{i=1}^{2^B}$ minimizing the quantizer's MSE in (3.8).

A conventional full range (FR) uniform quantizer simply sets $x_{q_i} = x_{\min} + (i-1)\Delta_x$, for $i = 1, \ldots, 2^B$, where $\Delta_x = (x_{\max} - x_{\min})2^{-B}$ is the quantization step. It is useful to approximate its quantization noise term q_x as a uniformly distributed random variable [108, 82], i.e., $q_x \sim U\left[-\frac{\Delta_x}{2}, \frac{\Delta_x}{2}\right]$. Consequently, we obtain $\sigma_{q_x}^2 = \frac{\Delta_x^2}{12}$, an expression for the quantizer's MSE.

3.3.1 Clipped Quantization

Recently, we have shown that a uniform quantizer's accuracy can be improved by allowing for signal clipping [109]. Specifically, all quantization levels are placed in a narrow interval $[x_L, x_R]$, with $x_L > x_{\min}$ and $x_R < x_{\max}$. The resulting quantizer has an MSE consisting of quantization and clipping noise terms [109]:

$$\text{MSE}(x_q) = \frac{\Delta_x^2}{12} + \sigma_c^2, \tag{3.9}$$

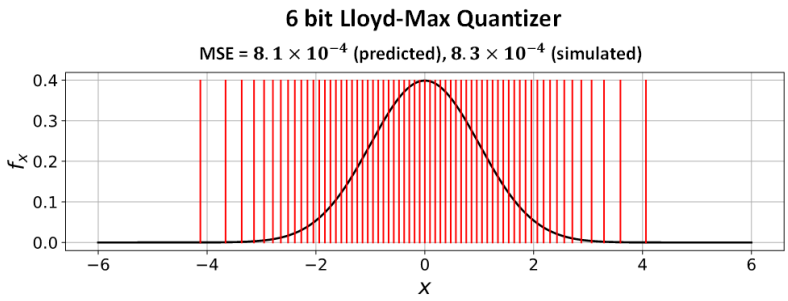
where, by virtue of the new quantization range, the step size is given by $\Delta_x = (x_R - x_L)2^{-B}$ and the clipping noise variance equals:

$$\sigma_c^2 = \mathbb{E}\left[(x - x_L)^2 \mathbb{1}_{\{x<x_L\}}\right] + \mathbb{E}\left[(x - x_R)^2 \mathbb{1}_{\{x>x_R\}}\right]. \tag{3.10}$$

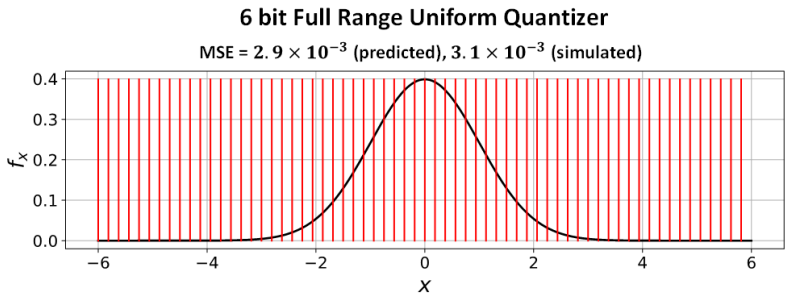
Thus, a clipped uniform quantizer exhibits a fundamental trade-off between its quantization and clipping noise. Hereafter, we demonstrate how to optimally clip a signal.

3.3.2 Optimally Clipped Quantization

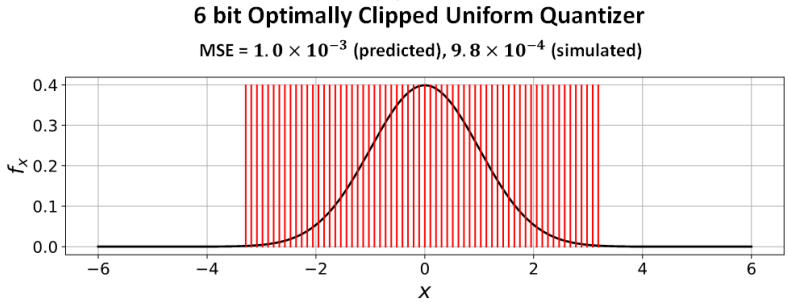
We present the optimal clipping criterion (OCC) for signals with a Gaussian distribution. Such signals are very prominent in machine learning systems, particularly in dot-product outputs by virtue of the Central Limit Theorem [110]. The following is our first main result:



(a)



(b)



(c)

Figure 3.2: Illustration of various quantization strategies for a unit Gaussian signal: (a) Lloyd-Max, (b) full range uniform, and (c) optimally clipped uniform quantizers. The predicted and simulated MSEs are obtained via evaluation of (3.8) using numerical integration and Monte Carlo simulations, respectively.

Theorem 3. *Given a Gaussian signal $x \sim \mathcal{N}(\mu_x, \sigma_x^2)$ and a B bit uniform quantizer, the optimal quantization range is $[\mu_x - \zeta^{(OCC)}\sigma_x, \mu_x + \zeta^{(OCC)}\sigma_x]$ where the optimal clipping factor $\zeta^{(OCC)}$ is the converging point of the following recursive expression:*

$$\zeta_{n+1} = \frac{\sqrt{\frac{2}{\pi}} e^{-\frac{\zeta_n^2}{2}}}{\frac{4^{-B}}{3} + 2Q(\zeta_n)}, \tag{3.11}$$

where $Q()$ represents the Q-function of a standard Gaussian.

Table 3.2: Comparison of MSE between OCC and LM

B	$\zeta^{(\mathrm{OCC})}$	$\sigma^2_{(\mathrm{OCC})}$	$\sigma^2_{(\mathrm{LM})}$	$\frac{\sigma^2_{(\mathrm{OCC})}}{\sigma^2_{(\mathrm{LM})}}$
2	1.71	1.26×10^{-1}	1.17×10^{-1}	1.077
3	2.15	3.79×10^{-2}	3.45×10^{-2}	1.099
4	2.55	1.16×10^{-2}	9.50×10^{-3}	1.221
5	2.94	3.50×10^{-3}	2.50×10^{-3}	1.560
6	3.29	1.04×10^{-3}	8.14×10^{-4}	1.278
7	3.61	3.04×10^{-4}	2.13×10^{-4}	1.427
8	3.92	8.77×10^{-5}	7.15×10^{-5}	1.227
9	4.21	2.49×10^{-5}	2.02×10^{-5}	1.232
10	4.49	6.99×10^{-6}	5.11×10^{-6}	1.368

Proof. See Section 3.8. □

An important consequence of Theorem 3 is that $\zeta^{(\mathrm{OCC})}$ depends on the number of bits B. Second, (3.11) does not explicitly compute $\zeta^{(\mathrm{OCC})}$ and requires an initial guess ζ_0. The process is computationally simple. We found that no more than 10 iterations are needed when $\zeta_0 = 4$.

The use of OCC and its comparison to LM and FR are illustrated in Figure 3.2. A unit Gaussian signal, confined to the interval $[-6, 6]$, is quantized to 6 bit. Corresponding quantization levels are shown and quantizer MSE is reported.

First, LM places most quantization levels near the mean as shown in Figure 3.2 (a). Intuitively, most of the representation is allocated to high-density regions which minimizes the MSE. Unfortunately, the quantization levels are non-uniform which can lead to design difficulty due to the need for complex circuitry [111, 112].

In contrast, the non-parametric FR has a large MSE. Indeed, many of its quantization levels are placed on the tails of the distribution which is data inefficient as shown in Figure 3.2(b).

Figure 3.2(c) illustrates the improvements OCC provides. Via clipping, the region of high signal density is pinpointed. As a result, the quantizer's accuracy is almost as good as LM's and it maintains FR's desirable property of having uniformly placed quantization levels.

In Table 3.2, we tabulate $\zeta^{(\mathrm{OCC})}$ for varying values of B of interest. We also list $\sigma^2_{(\mathrm{OCC})}$ and $\sigma^2_{(\mathrm{LM})}$, the quantization noise variances for the OCC and LM quantizers, respectively, when applied to a unit Gaussian random variable.

These are obtained by evaluating (3.8) using numerical integration.

We also include the ratio $\frac{\sigma^2_{(\mathrm{OCC})}}{\sigma^2_{(\mathrm{LM})}}$. It is found that $\sigma^2_{(\mathrm{OCC})}$ is usually about $\sim$20% higher than $\sigma^2_{(\mathrm{LM})}$ and at worst 56% when $B = 5$. Equivalently, the OCC has an SQNR within 0.8 dB of LM. Thus, the OCC, being a uniform quantizer, constitutes a practical, yet reliable, alternative to LM.

3.3.3 Application of the OCC

Though OCC is meant for quantizing IMC dot-products it can be employed for quantizing digitally computed dot-products as well. We study the application of OCC in quantizing digital dot-products first and then IMC dot-products.

OCC in digital dot-products

For a digital realization of the dot-product described in Section 3.2, there are three sources of noise: input ($q_{x\to y}$), weight ($q_{w\to y}$), and output (q_y) quantization. The resulting signal-to-quantization-noise ratio (SQNR) is given by:

$$\mathrm{SQNR} = \frac{\sigma_y^2}{\sigma^2_{q_{x\to y}} + \sigma^2_{q_{w\to y}} + \sigma^2_{q_y}} \tag{3.12}$$

with

$$\sigma^2_{q_{x\to y}} = \frac{N x_m^2 \sigma_w^2 4^{-B_X}}{12}; \quad \sigma^2_{q_{w\to y}} = \frac{N w_m^2 \mathbb{E}[x^2] 4^{-B_W}}{3}, \tag{3.13}$$

and $\sigma^2_{q_y}$ depending on the output quantization strategy.

We consider a dot-product of dimension $N = 256$ where inputs and weights are uniformly distributed in $[0, 1]$ and $[-1, 1]$, respectively. Input and weight precisions are chosen as $B_X = B_W = 4$. The upper bound on the SQNR in (3.12) is 22.5 dB, obtained by setting $\sigma^2_{q_y} = 0$.

Output precision is typically set using the bit-growth criterion (BGC) [113]:

$$B_Y = B_X + B_W + \log_2(N). \tag{3.14}$$

This criterion is known to be overly conservative. We consider three out-

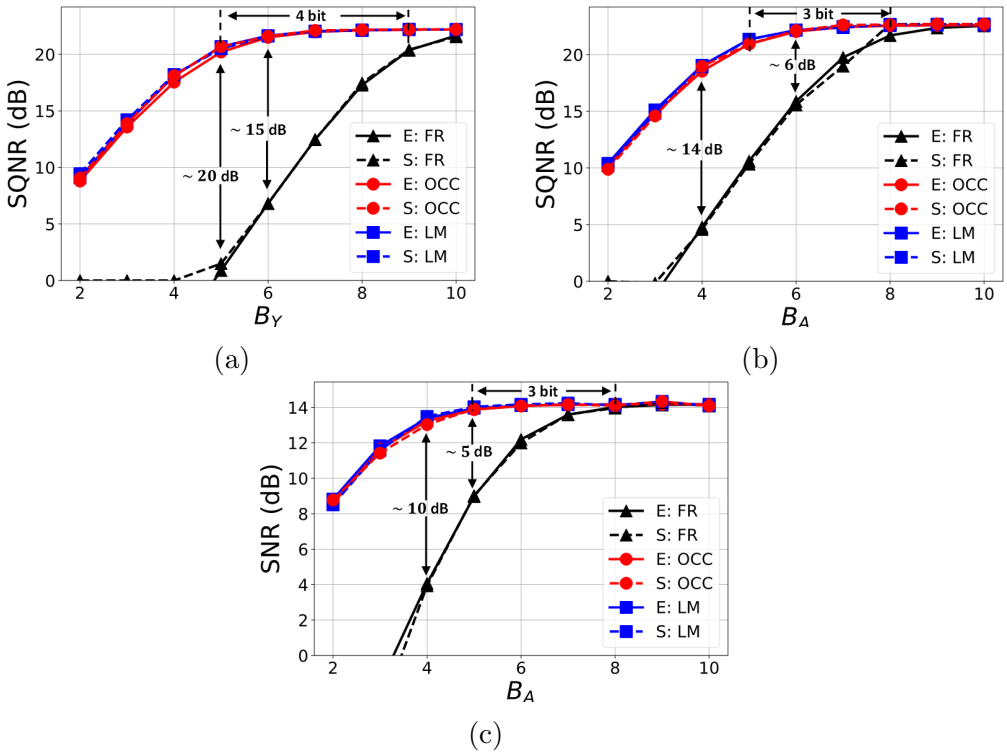


Figure 3.3: Comparison of FR, OCC, and LM for output and ADC quantization: (a) SQNR vs. B_Y in digital dot-products, (b) SQNR vs. B_A in IMC dot-products, and (c) SNR vs. B_A in IMC dot-products. The dot-product dimension is $N = 256$ and input/weight precisions are set as $B_X = B_W = 4$. The maximum achievable SQNR in (a) and (b) is 22.5dB. The bitcell capacitance used in (c) is $C_o = 1fF$ and the maximum achievable SNR is 14.5dB. Solid lines 'E' are obtained via evaluation of (3.12), (3.16), (3.15), (3.17), and (3.18); dashed lines 'S' are obtained using Monte Carlo simulations.

put quantization strategies: (1) FR employing the range $[-N, N]$, (2) OCC, and (3) LM. For each, B_Y is swept and the SQNR is evaluated both analytically (using (3.12), (3.8), and (3.9)) and empirically (using Monte Carlo simulations).

The results are shown in Figure 3.3. The following is observed: First, OCC's accuracy matches the optimal LM. The asymptotic SQNR of $\sim 22\,\text{dB}$ is attained for a value of $B_Y = 6$ for both quantizers. On the other hand, the commonly employed FR has a much smaller SQNR. Its gap with respect to LM and OCC is as high as 20 dB for $B_Y = 5$. In addition, it requires $B_Y = 10$ to reach the SQNR asymptote. OCC thus boasts a 4 bit reduction over FR, which is significant.

OCC in IMC dot-products

For an IMC realization of the dot-product described in Section 3.2, there are three quantization noise sources: input ($q_{x\to y}$), weight ($q_{w\to y}$), total column ADC operations ($q_{A\to y}$). In addition, circuit non-idealities contribute to a fourth term ($\eta_{a\to y}$). The resulting signal-to-noise ratio (SNR) is given by:

$$\mathrm{SNR} = \frac{\sigma_y^2}{\sigma_{q_{x\to y}}^2 + \sigma_{q_{w\to y}}^2 + \sigma_{q_{A\to y}}^2 + \sigma_{\eta_{a\to y}}^2}, \tag{3.15}$$

where $\sigma_{q_{x\to y}}^2$ and $\sigma_{q_{w\to y}}^2$ are given by (3.13), while $\sigma_{q_{A\to y}}^2$ and $\sigma_{\eta_{a\to y}}^2$ are derived hereafter.

Additionally, it is useful to consider an ideal scenario where analog noise does not exist (i.e., $\eta_{a\to y} = 0$). In this case, we define the SQNR of the IMC dot-product as:

$$\mathrm{SQNR} = \frac{\sigma_y^2}{\sigma_{q_{x\to y}}^2 + \sigma_{q_{w\to y}}^2 + \sigma_{q_{A\to y}}^2}. \tag{3.16}$$

The SQNR in (3.16) sets an upper bound on the SNR in (3.15).

We consider the bit-serial, bit-parallel (BSBP) architecture, a special case of the ISWP architecture discussed in Section 3.2 using $B_S = 1$ so that $N_S = B_X$. The BSBP architecture is popular [33] because it is believed to have the best accuracy (a claim we refute in Section 3.4). We consider the same setup as above and first study the SQNR in (3.16). The asymptote of 22.5 dB is identical to the digital dot-product case and can be obtained by setting $\sigma_{q_{A\to y}}^2 = 0$. For the BSBP architecture, it can be shown that:

$$\sigma_{q_{A\to y}}^2 = \frac{4}{9} x_m^2 w_m^2 \left(1 - 4^{-B_X}\right)\left(1 - 4^{-B_W}\right) \sigma_{q_{A_{s,b}}}^2, \tag{3.17}$$

where $\sigma_{q_{A_{s,b}}}^2$ is the intermediate column ADC quantization noise variance from (3.3). This quantity depends on the quantization strategy and can be evaluate using (3.8) or (3.9).

Figure 3.3 (b) illustrates how the BSBP architecture's SQNR varies with B_A. Once again, LM and OCC perform nearly identically. Compared to FR, OCC yields a 14 dB improvement when $B_A = 4$. Furthermore, OCC reaches the SQNR asymptote for $B_A = 5$ as opposed to FR which requires $B_A = 8$. Hence, OCC reduces ADC precision requirements by 3 bit.

The SNR in (3.15) is also studied. In the BSBP setup, the total analog noise variance is given by:

$$\sigma^2_{\eta_{a\to y}} = \frac{4}{9} x_m^2 w_m^2 \left(1 - 4^{-B_X}\right)\left(1 - 4^{-B_W}\right)\sigma^2_{\eta_{a_{s,b}}} \tag{3.18}$$

with $\sigma^2_{\eta_{a_{s,b}}}$ given by (3.4). We use a value of $C_o = 1\,\text{fF}$, resulting in an asymptotic SNR of 14.5 dB obtained by setting $\sigma^2_{q_{A\to y}} = 0$ in (3.15). Then, B_A is swept and the resulting SNR is included in Figure 3.3 (c). Once more, OCC performs as accurately as LM. Furthermore, it improves on FR by up to 10 dB for $B_A = 4$. The asymptote of $\sim 14\,\text{dB}$ is attained when $B_A = 5$, implying a 3 bit reduction compared to FR which requires $B_A = 8$.

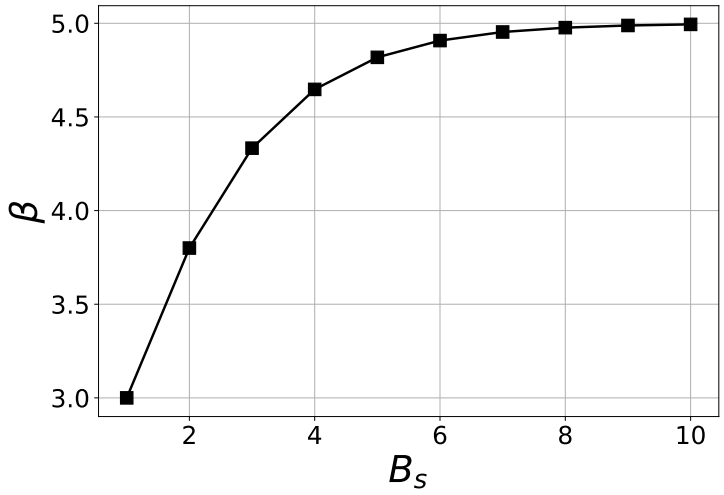
3.4 Bit Slicing Analysis

3.4.1 Noise Propagation and Futility of Bit Serialization

IMC precision limits [109], though not widely employed by hardware designers, are known to exist [98]. When the column ADC precision B_A is fixed (for instance due to area constraints), designers attempt to improve IMC accuracy by reducing the input precision. The usual argument is to re-purpose the BGC in (3.14).

When applied to a column accumulation within an IMC architecture, B_A replaces B_Y, B_S replaces B_X, $B_W = 1$, and N is set by the number of active rows in the array. As B_S is the only free design parameter, IMC designers, usually attempt to satisfy BGC by choosing the smallest value for $B_S = 1$, i.e., use the BSBP architecture [33, 114]. This leads to $N_S = B_X$, i.e., a maximal number of array accesses.

In reality, a trade-off between quantization noise from individual ADC operations and accumulation of noise in POTs exists. We prove that the common choice of $B_S = 1$ is sub-optimal in terms of accuracy and energy efficiency. We show that the use of a higher value for B_S minimally impacts $\sigma^2_{q_{A\to y}}$ and $\sigma^2_{\eta_{a\to y}}$. First, we consider the impact of bit slicing on ADC quantization noise. We prove in Section 3.8 that, when individual input and weight



(a)

Figure 3.4: Bit slicing gain β in (3.20).

bits are equally likely to be 0 or 1, and when the OCC is employed, then:

$$\sigma^2_{q_{A\to y}} = \frac{N\sigma^2_{(\mathrm{OCC})}\left(1-4^{-B_X}\right)\left(1-4^{-B_W}\right)}{36x_m^{-2}w_m^{-2}} \times \beta, \tag{3.19}$$

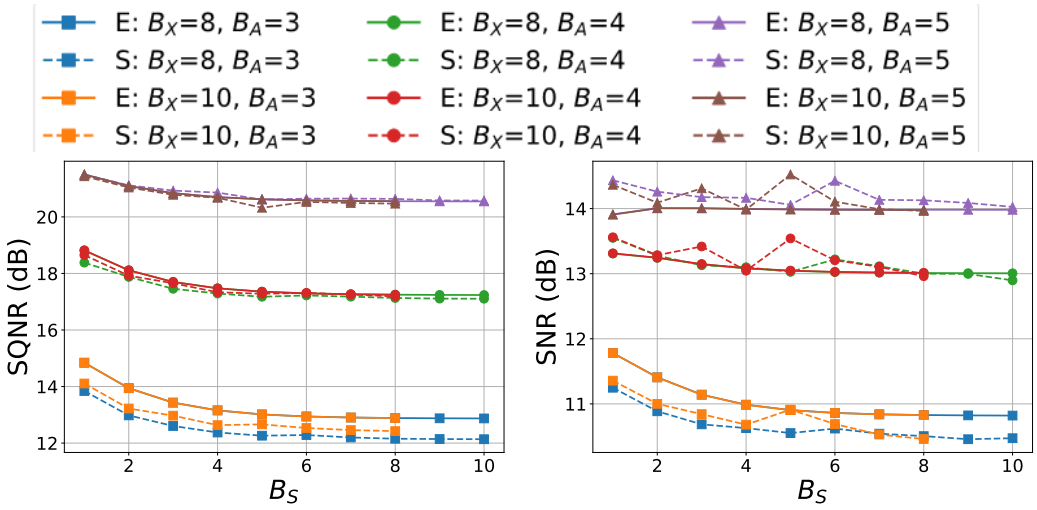
where β is the *ADC quantization noise bit slicing gain* given by:

$$\beta = \frac{5-2^{-B_S}}{1+2^{-B_S}}. \tag{3.20}$$

In Figure 3.4, we plot β as a function of B_S. The smallest value for β is equal to 3 and corresponds to $B_S = 1$, validating that the BSBP architecture has the best accuracy. However, the largest value for β is equal to 5 when $B_S \to \infty$. Thus, the impact of bit slicing causes a 1.6× increase in total ADC quantization noise variance. This corresponds to a 2 dB SQNR drop at worst which is equivalent to a third of an LSB [81]. Thus, in principle, the accuracy benefits from using a single bit per slice are very limited and arguably do not outweigh the price of accessing the array multiple times.

We also analyze the impact of bit slicing on analog noise. In Section 3.8, we prove that:

$$\sigma^2_{\eta_{a\to y}} = \frac{4x_m^2w_m^2\left(1-4^{-B_X}\right)\left(1-4^{-B_W}\right)}{3\left(1-4^{-B_S}\right)}\sigma^2_{\eta_{a_{s,b}}} \tag{3.21}$$



(a) (b)

Figure 3.5: Impact of bit slicing on the accuracy of IMC dot-products: (a) SNQR vs. B_S and (b) SNR vs. B_S. The legend is included at the top of the figure and lists various values of B_X and B_A used. The dot-product dimension is $N = 256$ and weight precision is set as $B_W = 4$. The bitcell capacitance used in (c) is $C_o = 1\,\text{fF}$. Solid lines 'E' are obtained via evaluation of (3.12), (3.16), (3.15), (3.19), (3.21), and (3.22); dashed lines 'S' are obtained using Monte Carlo simulations.

with:

$$\sigma^2_{\eta_{a_{s,b}}} = N\left(\frac{\rho_1\left(2-2^{-B_S}\right)}{12\left(1-2^{-B_S}\right)C_o} + \frac{\rho_2}{C_o} + \frac{\rho_3}{C_o^2}\right). \tag{3.22}$$

Thus, when B_S increases, $\sigma^2_{\eta_{a\to y}}$ decreases, though not drastically. This is not surprising since higher B_S leads to fewer memory accesses, meaning fewer instances when circuit non-idealities are accumulated.

Summarizing our findings above, no clear advantages can be associated with the use of $B_S = 1$. Thus, we recommend choosing $B_S = B_X$ whenever possible. Such fully sliced (FS) designs improve energy efficiency and do not hinder accuracy. Generally, FS hardware is available for moderate input precision values ($B_X \leq 8$) [30]. For applications necessitating high input precision ($B_X > 8$), we recommend choosing the largest value of B_S supported by the hardware.

3.4.2 Impact on Dot-product Accuracy

We confirm our findings above by investigating the impact of bit-slicing on the accuracy of IMC dot-products.

We use the same setup as in Section 3.6.1, but consider higher input precision B_X to increase the choices for B_S. Specifically, we keep $B_W = 4$, $N = 256$ but use $B_X = 8$ and $B_X = 10$. For each case, we sweep the value of $B_S = 1, \ldots, B_X$. The column ADC precision B_A is also fixed to 3, 4, or 5 bit, and always uses the OCC.

Figure 3.5 (b) shows plots of the SQNR as a function of B_S for these various cases. Validating our analysis, results indicate that the choice of B_S has a minor effect on the SQNR. When $B_A = 3$, the SQNR lies between $\sim$ 14 dB for $B_S = 1$ and $\sim$ 12 dB for $B_S \to B_X$. Thus, our contention that single bit slicing offers no more than a 2 dB SQNR boost is verified. Results when $B_A > 3$ also corroborate this finding. In general, when B_A increases (and ADC quantization noise no longer dominates), the SQNR sensitivity to B_S decreases.

Figure 3.5 (c) illustrates the impact of bit slicing on the SNR of IMC dot-products given by (3.15). The setup is identical to the above SQNR study, but analog noise is also enabled with $C_o = 1$ fF. The SNR is minimally affected by the choice of B_S, as predicted. Its value is lower than the SQNR for various configurations due to the presence of analog noise. When $B_A = 3$, the SNR lies between $\sim$ 11.5 dB for $B_S = 1$ and $\sim$ 10.5 dB for $B_S \to B_X$. Thus, in the case of the SNR, the benefits of single-bit slicing are reduced to just 1 dB. In fact, when $B_A = 5$, the SNR does not vary as a function of B_S, but remains constant and equal to $\sim$ 14 dB.

3.5 DNN Implementation Methodology

3.5.1 Setup and Accuracy Analysis

We apply our analysis above for DNN inference using IMC architectures. Direct mapping is considered. More precisely, we assume each layer's computation is mapped to a separate memory bank of appropriate size (matched to the dot-product dimension at that layer). Thus, for each of the L layers,

there are four design parameters to be determined: activation ($\{B_{X,l}\}_{l=1}^{L}$), weight ($\{B_{W,l}\}_{l=1}^{L}$), and ADC ($\{B_{A,l}\}_{l=1}^{L}$) precisions, as well as bitcell capacitance ($\{C_{o,l}\}_{l=1}^{L}$).

In order to analyze the accuracy of the implemented DNN, we use the *mismatch probability* metric [89], $p_m = \Pr\{\hat{C}_{FL} \neq \hat{C}_{IMC}\}$, measuring the probability of disagreement in floating-point ($\hat{C}_{FL}$) and IMC ($\hat{C}_{IMC}$) predicted classes.

Following a similar approach to [115], we prove in Section 3.8 that the following upper bound holds:

$$p_m \leq \sum_{l=1}^{L} \left(\sigma^2_{q_{x\to y_l}} + \sigma^2_{q_{w\to y_l}} + \sigma^2_{q_{A\to y_l}} + \sigma^2_{\eta_{a\to y_l}} \right) E_{Y,l}, \tag{3.23}$$

where, at layer l, $\sigma^2_{q_{x\to y_l}}$, $\sigma^2_{q_{w\to y_l}}$, and $\sigma^2_{q_{A\to y_l}}$ are the output referred variances from input, weight, and ADC quantization noise, respectively; $\sigma^2_{\eta_{a\to y_l}}$ is the output referred variance from analog noise; and $E_{Y,l}$ is the *pre-activation noise gain* given by:

$$E_{Y,l} = \mathbb{E}\left[\sum_{\substack{i=1 \\ i\neq \hat{C}_{FL}}}^{M} \frac{\sum\limits_{h\in\mathcal{Y}_l} \left| \frac{\partial (Z_i - Z_{\hat{C}_{FL}})}{\partial Y_h} \right|^2}{2|Z_i - Z_{\hat{C}_{FL}}|^2} \right] \tag{3.24}$$

with M being the number of classes, $\{Z_i\}_{i=1}^{M}$ being the network soft outputs, and $\mathcal{Y}_l$ being the index set of pre-activations (dot-product outputs) at layer l.

3.5.2 Mismatch Budget Distribution

Consider the problem of IMC implementation, where a complexity metric $f()$ needs to be minimized subject to an accuracy constraint:

$$\begin{aligned} \min_{\{B_{X,l},B_{W,l},B_{A,l},C_{o,l}\}_{l=1}^{L}} \; & f\left(\{B_{X,l}, B_{W,l}, B_{A,l}, C_{o,l}\}_{l=1}^{L}\right) \\ \text{such that: } & p_m \leq p_t, \end{aligned} \tag{3.25}$$

where p_t is the target worst-case mismatch allowed. One choice for $f()$ can be E_{OP} in (3.5). The solution of the resulting optimization problem is beyond

the scope of this dissertation.

Meaningful complexity metrics often monotonically increase as a function of the various design parameters in (3.25). Such problems are typically solved by finding extremal points satisfying the constraints [116, 117]. Thus, keeping $f()$ in (3.25) general, we show hereafter how to find such solutions via *mismatch budget distribution*.

Note that the upper bound on p_m in (3.23) can be written as:

$$p_m \leq p_{m_{X,W}} + p_{m_A} + p_{m_\eta}, \tag{3.26}$$

where

$$p_{m_{X,W}} = \sum_{l=1}^{L} \left(\sigma^2_{q_{x \to y_l}} + \sigma^2_{q_{w \to y_l}} \right) E_{Y,l}$$
$$p_{m_A} = \sum_{l=1}^{L} \sigma^2_{q_{A \to y_l}} E_{Y,l} \quad \text{and} \quad p_{m_\eta} = \sum_{l=1}^{L} \sigma^2_{\eta_{a \to y_l}} E_{Y,l}$$

correspond to mismatches caused by input/weight quantization, ADC quantization, and analog noise, respectively.

Thus, in order to satisfy the accuracy requirements, it is enough to find extremal values for the $4L$ design parameters such that:

$$p_{m_{X,W}} \leq \alpha_1 p_t \quad \text{and} \quad p_{m_A} \leq \alpha_2 p_t \quad \text{and} \quad p_{m_\eta} \leq \alpha_3 p_t, \tag{3.27}$$

where $\alpha_1 + \alpha_2 + \alpha_3 = 1$. Thus, each of the three mismatch contributors can be optimized separately.

The specific choice for $(\alpha_1, \alpha_2, \alpha_3)$ relates to the complexity metric $f()$ in (3.25), i.e., the application. If the bitcell capacitance has to be kept as small as possible due to area constraints, α_3 should be made largest, allowing for more analog noise. If the BCA is dense, the column ADC needs to be physically small to enable pitch-matched design. In this case, α_2 should be made largest allowing for more ADC quantization noise but smaller precision. Finally, α_1 can be made largest if it is required to minimize activation and weight precisions as much as possible. This is useful to reduce the number of required columns and potentially the number of memory accesses.

Once the choice for $(\alpha_1, \alpha_2, \alpha_3)$ is made, activation and weight per-layer precision can be solved for via the method of noise equalization in [115, 118].

In what follows, we derive a method to explicitly solve for per-layer ADC precision and bitcell capacitance.

3.5.3 Per-layer ADC Precision via Noise Equalization

To solve $p_{m_A} \leq \alpha_2 p_t$ in (3.27), we formulate a per-layer precision assignment following a noise equalization. As discussed in Sections 3.3 and 3.4, we recommend FS dot-products using the OCC. For this strategy, we prove in Section 3.8 that a noise equalized (NE) ADC precision assignment can be obtained as:

$$B_{A,l} = B_{\min} + \gamma \ln \frac{F_{Y,l}^{(\mathrm{OCC})}}{F_{\min}^{(\mathrm{OCC})}} \tag{3.28}$$

for $l = 1, \ldots, L$, where $\gamma = \frac{1}{\ln(3.4)}$, $B_{\min}$ is a reference precision, $F_{Y,l}^{(\mathrm{OCC})}$ is the *OCC-normalized pre-activation noise gain* at layer l, given by:

$$F_{Y,l}^{(\mathrm{OCC})} = \frac{N_l \left(1 - 2^{-B_{X,l}}\right) \left(1 - 4^{-B_{W,l}}\right) \left(5 - 2^{-B_{X,l}}\right)}{36 x_{m,l}^{-2} w_{m,l}^{-2}} E_{Y,l} \tag{3.29}$$

with N_l being the dot-product dimension at layer l and $x_{m,l}$, $w_{m,l}$ being the activation/weight peaks at layer l, respectively; and $F_{\min}^{(\mathrm{OCC})}$ is the least of $\{F_{Y,l}^{(\mathrm{OCC})}\}_{l=1}^{L}$. Thus, the search for per-layer ADC precisions is reduced in (3.28) to a single axis (linear complexity) and consists of finding a suitable value for $B_{\min}$. The resulting design will be referred to as FS/OCC/NE.

Similarly, we derive a per-layer precision assignment for bit-serial (BS) dot-products using FR ADC quantization. The corresponding NE assignment is given by (see Section 3.8):

$$B_{A,l} = B_{\min} + \log_4 \frac{F_{Y,l}^{(\mathrm{FR})}}{F_{\min}^{(\mathrm{FR})}} \tag{3.30}$$

for $l = 1, \ldots, L$, where $F_{Y,l}^{(\mathrm{FR})}$ is the *FR-normalized pre-activation noise gain* at layer l, given by:

$$F_{Y,l}^{(\mathrm{FR})} = \frac{N_l^2 \left(1 - 4^{-B_{X,l}}\right) \left(1 - 4^{-B_{W,l}}\right)}{27 x_{m,l}^{-2} w_{m,l}^{-2}} E_{Y,l}. \tag{3.31}$$

And once again we obtain a search over a single axis (linear complexity). We term this design BS/FR/NE.

Note that BS/FR/NE is an improvement over the commonly employed bit-serial, bit-growth (BS/BG) [33] which attempts to suppress all ADC quantization noise using the following per-layer assignments:

$$B_{A,l} = \lfloor \log_2(N_l) \rfloor + 1 \tag{3.32}$$

for $l = 1, \ldots, L$. Finally, though not commonly employed, we also consider FS/BG, the natural extension of BS/BG that applies to FS dot-products, with per-layer assignment:

$$B_{A,l} = B_{X,l} + \lceil \log_2(N_l) \rceil \tag{3.33}$$

for $l = 1, \ldots, L$. We use BS/BG and FS/BG to demonstrate the usefulness of noise equalization. A comparison between FS/OCC/NE and BS/FR/NE highlights the benefits of our proposed OCC and bit slicing analyses in Sections 3.3 and 3.4.

3.5.4 Per-layer Bitcell Capacitance Design

Depending on the available technology, the implementation could dictate identical bitcells across all IMC arrays. In this case $C_{o,l} = C_o$ for $l = 1, \ldots, L$ making p_{m_η} a second-order polynomial in $\frac{1}{C_o}$ so that the upper bound $p_{m_\eta} \leq \alpha_3 p_t$ in (3.27) can be solved exactly. We do assume hereafter that different BCAs can be used for different layers. Still, the upper bound on p_{m_η} can be solved exactly since the latter is a polynomial in $\{\frac{1}{C_{o,l}}\}_{l=1}^{L}$. Notwithstanding, a noise equalization method can be formulated leading to an extremal solution.

Indeed, provided $C_{o,l} \geq 1\,\text{fF}$ for $l = 1, \ldots, L$, then, as proven in the Section 3.8, the following per-layer bitcell capacitance value assignment equalizes analog noise contributions:

$$C_{o,l} = C_{\min} \frac{G_{Y,l}}{G_{\min}} \tag{3.34}$$

for $l = 1, \ldots, L$, where $C_{\min}$ is a reference capacitance value, $G_{Y,l}$ is the

Table 3.3: Topological details of the VGG-9 DNN

Layer index	Tensor dimension	$B_{X,l}$	$B_{W,l}$	D_l	N_l	# of banks
1	(3,64,3,3)	7	8	27	27	1
Max Pool: 2×2						
2	(64,128,3,3)	6	7	576	512	2
Max Pool: 2×2						
3	(128,256,3,3)	6	6	1152	512	3
Max Pool: 2×2						
4	(256,256,3,3)	6	6	2304	512	5
Max Pool: 2×2						
5	(256,512,3,3)	6	6	2304	512	5
6	(512,512,3,3)	6	5	4608	512	9
Max Pool: 2×2						
7	(512,512,3,3)	6	5	4608	512	9
8	(512,512,3,3)	5	4	4608	512	9
9	(512,10)	4	4	512	512	1

The required activation/weight precisions ($B_{X,l}/B_{W,l}$) to satisfy the bound on $p_{m_{X,W}}$ in (3.23) via noise equalization are listed. Additionally, per-layer tensor dimensions and corresponding dot-product dimension D_l are included. Memory arrays of at most 512 rows are assumed to be utilizable. For each layer, the assumed array size N_l is listed, and if need be, the computation is partitioned across multiple banks. Layers with 4D tensors implement 2D convolutions whereas layers with 2D tensors implement matrix vector multiplications.

capacitance-normalized preactivation noise gain at layer l, given by:

$$G_{Y,l} = \frac{\rho_1 N_l \left(1 - 4^{-B_{X,l}}\right)\left(1 - 4^{-B_{W,l}}\right)\left(2 - 2^{-B_{S,l}}\right)}{9 x_{m,l}^{-2} w_{m,l}^{-2} \left(1 - 4^{-B_{S,l}}\right)\left(1 - 2^{-B_{S,l}}\right)} E_{Y,l} \tag{3.35}$$

with $B_{S,l}$ being the number of bits per slice used at layer l ($B_{S,l} = B_{X,l}$ for FS and $B_{S,l} = 1$ for BS), and $G_{\min}$ is the least of $\{G_{Y,l}\}_{l=1}^{L}$. Much like noise equalization for precision assignment, (3.34) reduces the search for per-layer capacitance value into the single axis of $C_{\min}$. This search can thus be performed in linear time.

Table 3.4: Achieved accuracy under various implementations

Design	Accuracy (%)
Floating-point	87.71
Fixed-point (activation and weight quantization only)	87.47
FS/OCC/NE without analog noise	87.14
BS/FR/NE without analog noise	87.18
FS/OCC/NE with analog noise	86.73±0.108
BS/FR/NE with analog noise	86.74±0.113

3.6 Numerical Results

3.6.1 Experimental Setup

We illustrate the application of our analyses in Sections 3.3, 3.4, and 3.5 using a VGG-9 [119] network deployed on the CIFAR-10 [65] dataset. The floating-point accuracy achieved is 87.79%. We choose $p_t = 1\%$, for the worst-case accuracy drop, and $\alpha_1 = \alpha_2 = \alpha_3 = \frac{1}{3}$. Table 3.3 lists the network details, including values for $\{B_{X,l}\}_{l=1}^{L}$ and $\{B_{W,l}\}_{l=1}^{L}$ required to achieve $p_{m_{X,W}} \leq \alpha_1 p_t$, as well as the dot-product dimension at each layer $\{D_l\}_{l=1}^{L}$. We assume the maximum dot-product dimension realizable in IMC is 512. For each layer, we list the array size N_l used. If need be, dot-products are partitioned into multiple banks as indicated. Note that the analysis of Section 3.5 still holds under this setup with the caveat that pre-activation noise gains are computed on partial dot-products.

Furthermore, Table 3.4 lists the inference accuracy for various implementations. By design, all accuracies fall within 1% of the floating-point baseline. Thus, the comparisons and insights derived hereafter apply to *iso-accurate designs*.

3.6.2 Per-layer ADC Precision Assignment

For each of the four implementations considered in Section 3.5.3, the corresponding per-layer ADC precision assignments ($\{B_{A,l}\}_{l=1}^{L}$) are listed in Table

Table 3.5: Per-layer ADC precision and bitcell capacitance values for various implementations considered

ADC Precision $B_{A,l}$

Layer Index	FS/OCC/NE	BS/FR/NE	BS/BG	FS/BG
1	8	**5**	**5**	12
2	**8**	9	10	15
3	**8**	9	10	15
4	**7**	9	10	15
5	**8**	9	10	15
6	**7**	9	10	15
7	**8**	9	10	15
8	**6**	8	10	14
9	**7**	9	10	13

Capacitance Value $C_{o,l}$ (fF)

Layer Index	FS/OCC/NE	BS/FR/NE
1	**1.25**	1.32
2	**23.8**	26.6
3	**35.0**	39.2
4	**32.3**	36.1
5	**56.5**	63.3
6	**75.2**	84.1
7	**86.7**	101.4
8	**11.5**	12.9
9	**17.8**	19.9

3.5.

We first compare the commonly employed BS/BG to BS/FR/NE. The latter supposedly improves upon the former thanks to the use of noise equalization. Nevertheless, the resulting precision values for $\{B_{A,l}\}_{l=1}^{L}$ are only 1 bit lower. It is only at layer 8 that BS/FR/NE improves on BS/BG by 2 bit. This marginal improvement is explained next.

The FR-normalized pre-activation noise gain in (3.31) increases quadratically as a function of the dot-product dimension. Therefore, the impact of ADC quantization on the network accuracy is most sensitive to $\{N_l\}_{l=1}^{L}$, the sole parameters accounted for by BS/BG in (3.32). This explains the similarity between these two precision assignment strategies.

Next, let us compare FS/OCC/NE to FS/BG. The former ensures zero ADC quantization noise for FS dot-products. Our proposed methods relax this constraint by allowing for tolerable quantization noise in an accuracy-optimal manner. This relaxation leads to massive ADC precision reduction

of 7 to 8 bit, except for the first and last layers.

Finally, we compare FS/OCC/NE to BS/FR/NE, both of which use noise equalization. Hence, this comparison serves as evidence supporting our earlier contention in Sections 3.3 and 3.4 that (1) full slicing does not adversely impact ADC quantization noise, and (2) the use of OCC leads to significantly lower ADC precision requirements compared to FR.

Results do validate our claims. Save for the first layer, FS/OCC/NE always leads to lower ADC precision than BS/FR/NE. The precision reduction is 1 to 2 bit for most layers. Note that the OCC-normalized pre-activation noise gain only grows linearly with dot-product in (3.28), as opposed to the aforementioned quadratic dependence of its FR-normalized counterpart. Thus, we expect lower precision for large dot-product dimensions. This is indeed observed in Table 3.5.

The first layer constitutes an exception whereby BS/FR/NE assigns only 5 bit compared to 8 bit for FS/OCC/NE. This is due to the very short dot-product dimension of 27. Indeed, the OCC is intended to be used with Gaussian distributions. Column outputs in IMC arrays typically follow such distributions for large row-wise accumulations [110]. In fact, IMC is most appropriate for high dimensionality problems where the inherent parallelism and attractive density features of BCAs can be leveraged [96, 99]. Because of its short dot-product, we argue that the first layer is in fact unfit to be implemented using IMCs and could instead be handled using digital circuits.

3.6.3 Per-layer Bitcell Capacitance

Using noise equalization in (3.34), we determine per-layer bitcell capacitance values ($\{C_{o,l}\}_{l=1}^{L}$) to be used for both FS/OCC/NE and BS/FR/NE. The bitcell capacitance is a knob that only sets the analog noise which does not depend on the ADC quantization strategy. Hence, the use of FS vs. BS is only relevant in the comparison of the two implementations.

We claimed in Section 3.4 that the analog noise variance in (3.22) is minimally impacted by the number of bits per slice. We did find that FS leads to smaller analog variance compared to BS, but the reduction is minor. Should this claim be true, the required capacitance values under both implementations are expected to be similar. Our results do validate this claim as per

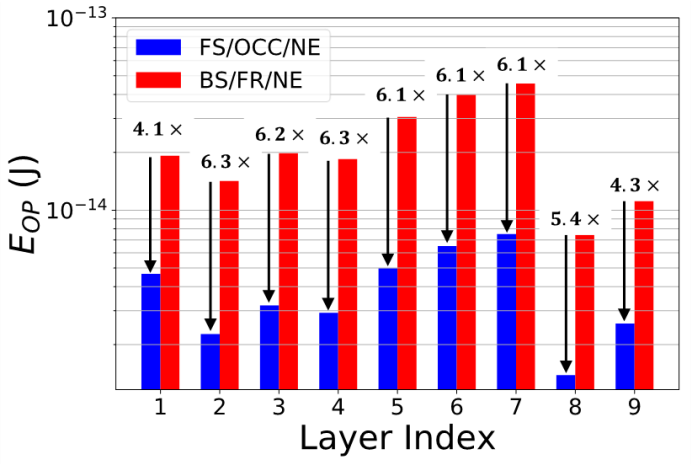


Figure 3.6: Per-layer precision normalized energy per operation E_{OP} in (3.5) for the FS/OCC/NE and BS/FR/NE implementations.

the listed values of $\{C_{o,l}\}_{l=1}^{L}$ in Table 3.5. It is found that FS/OCC/NE can operate with smaller bitcell capacitance values compared to BS/FR/NE, but the differences between the two are small and do not exceed 16%.

3.6.4 Per-layer Energy Consumption

We compare implementations FS/OCC/NE and BS/FR/NE in terms of energy consumption. We use the precision normalized energy per operation in (3.5). There are two sources of energy savings that FS/OCC/NE benefits from: (1) lower ADC precisions, and (2) smaller number of BCA accesses. These benefits translate to $\sim 6\times$ energy reduction compared to BS/FR/NE as shown in Figure 3.6.

In general, FS on its own leads to an energy reduction factor approximately equal to the activation precision. For instance, at layer 8, since $B_{X,l} = 5$, a reduction of $5\times$ is expected upfront. In addition, the smaller ADC precision of $B_{A,l} = 6$ for FS/OCC/NE compared to $B_{A,l} = 8$ for BS/FR/NE accentuates these savings by virtue of lower ADC energy E_{ADC} in (3.5). In this case, reduction of ADC precision increases the energy savings factor from $5\times$ to $\sim 5.4\times$. Interestingly, even for the first layer, where BS/FR/NE uses a lower precision for the ADC, FS/OCC/NE still exhibits $\sim 4.2\times$ energy reduction due to the use of FS.

3.7 Summary

In this chapter, an optimal clipping criterion for uniform quantization has been derived. It was shown that even with uniformly placed quantization levels, maximum SQNR/SNR (as achieved by the Lloyd-Max algorithm) can be achieved. Thus, the trade-off between hardware compatibility and quantization accuracy is eliminated. An analysis of bit slicing was performed and it was found that full slicing suffers no clear disadvantages compared to the bit-serial approach, believed to be most accurate. Thus, full slicing is recommended potentially leading to significant energy reduction. Finally, neural network implementation using IMC was investigated. An analytical method for per-layer parameter optimization was proposed. Combining all our contributions, iso-accurate inference at a much reduced cost was enabled.

The presented work allows IMC designers to push the available hardware limits and target highly efficient DNN implementations. Future work includes derivation of similar analyses for different compute models, as well as parameter design via constrained optimization using a complexity objective. In general, expanding the proposed framework can lead to automated tools assisting IMC designers in the seamless implementation of DNNs in hardware operating at the limits of energy efficiency.

3.8 Addendum: Proofs of Theoretical Results

Proof of Theorem 3:
Without loss of generality, we consider B bit uniform quantization of a unit Gaussian signal $x \sim \mathcal{N}(0,1)$ in the range $[x_L, x_R]$. A necessary condition for optimality is $x_R = -x_L = \zeta$ by virtue of the distribution's symmetry. The MSE in (3.9) can be written as the following function of ζ:

$$f(\zeta) = \frac{\zeta^2 2^{-2B}}{3} + 2\int_{\zeta}^{\infty} \frac{1}{\sqrt{2\pi}}(x-\zeta)^2 e^{-\frac{x^2}{2}} dx, \tag{3.36}$$

where we used $\Delta_x = \zeta 2^{-B}$ and $\sigma_c^2 = 2\mathbb{E}\left[(x-\zeta)^2 \mathbb{1}_{\{x>\zeta\}}\right]$. Our task is to find $\zeta^{(\mathrm{OCC})}$ minimizing $f(\zeta)$ in (3.36) which can be written as:

$$f(\zeta) = f_0(\zeta) + \sqrt{\frac{2}{\pi}}\left(f_1(\zeta) + f_2(\zeta) + f_3(\zeta)\right) \tag{3.37}$$

with $f_0(\zeta) = \frac{\zeta^2 2^{-2B}}{3}$, $f_1(\zeta) = \int_\zeta^\infty x^2 e^{-\frac{x^2}{2}} dx$, $f_2(\zeta) = -2\zeta \int_\zeta^\infty x e^{-\frac{x^2}{2}} dx$, and $f_3(\zeta) = \zeta^2 \int_\zeta^\infty e^{-\frac{x^2}{2}} dx$. It follows that:

$$f_0'(\zeta) = 2\zeta \frac{2^{-2B}}{3} \quad \text{and} \quad f_1'(\zeta) = -\zeta^2 e^{-\frac{\zeta^2}{2}} \tag{3.38}$$

$$f_2'(\zeta) = -2\int_\zeta^\infty x e^{-\frac{x^2}{2}} dx + 2\zeta^2 e^{-\frac{\zeta^2}{2}} = 2(\zeta^2 - 1)e^{-\frac{\zeta^2}{2}} \tag{3.39}$$

$$f_3'(\zeta) = 2\zeta \int_\zeta^\infty e^{-\frac{x^2}{2}} dx - \zeta^2 e^{-\frac{\zeta^2}{2}}. \tag{3.40}$$

Combining (3.37), (3.38), (3.39), and (3.40) yields:

$$\begin{aligned} f'(\zeta) &= 2\zeta \frac{2^{-2B}}{3} + \sqrt{\frac{2}{\pi}}\left(2\zeta \int_\zeta^\infty e^{-\frac{x^2}{2}} dx - 2e^{-\frac{\zeta^2}{2}}\right) \\ &= 2\left[g_0(\zeta) + \sqrt{\frac{2}{\pi}}\left(g_1(\zeta) + g_2(\zeta)\right)\right], \end{aligned} \tag{3.41}$$

where $g_0(\zeta) = \zeta \frac{2^{-2B}}{3}$, $g_1(\zeta) = \zeta \int_\zeta^\infty e^{-\frac{x^2}{2}} dx$, and $g_2(\zeta) = -e^{-\frac{\zeta^2}{2}}$. It follows that:

$$g_0'(\zeta) = \frac{2^{-2B}}{3} \quad \text{and} \quad g_2'(\zeta) = \zeta e^{-\frac{\zeta^2}{2}} \tag{3.42}$$

$$g_1'(\zeta) = \int_\zeta^\infty e^{-\frac{x^2}{2}} dx - \zeta e^{-\frac{\zeta^2}{2}}. \tag{3.43}$$

Combining (3.41), (3.42), and (3.43) yields:

$$f''(\zeta) = 2\left(\frac{2^{-2B}}{3} + \sqrt{\frac{2}{\pi}} \int_\zeta^\infty e^{-\frac{x^2}{2}} dx\right), \tag{3.44}$$

which is strictly positive for any ζ. Hence, $f(\zeta)$ is convex and can be minimized using Newton's algorithm [120] via the following recursion:

$$\zeta_{n+1} = \zeta_n - \frac{f'(\zeta_n)}{f''(\zeta_n)}. \tag{3.45}$$

Replacing (3.41) and (3.44) into (3.45) and substituting $\sqrt{\frac{2}{\pi}}\int_{\zeta}^{\infty} e^{-\frac{x^2}{2}}dx = 2Q(\zeta)$ yields (3.11) in Theorem 3 which concludes our proof.

Derivation of eq. (3.19)**:**
Combining (3.1) and (3.3), we have:

$$q_{A\to y} = x_m w_m \sum_{s=0}^{N_S-1}\left(-q_{A_{s,0}} + \sum_{b=1}^{B_W-1} q_{A_{s,b}} 2^{-b}\right) 2^{-sB_S}$$

and it follows that:

$$\begin{aligned}\sigma^2_{q_{A\to y}} &= x_m^2 w_m^2 \sum_{s=0}^{N_S-1}\sum_{b=0}^{B_W-1} \sigma^2_{q_{A_{s,b}}} 4^{-b} 4^{-sB_S} \\ &= \frac{4x_m^2 w_m^2}{3}\sigma^2_{q_{A_{s,b}}} \frac{\left(1-4^{-B_W}\right)\left(1-4^{-B_X}\right)}{1-4^{-B_S}}.\end{aligned} \tag{3.46}$$

Recall the column ADC uses the OCC so that:

$$\sigma^2_{q_{A_{s,b}}} = \mathrm{Var}(y_{s,b})\sigma^2_{(\mathrm{OCC})} = N\mathrm{Var}(x_{\{-s-\}}w_{\{b\}})\sigma^2_{(\mathrm{OCC})}. \tag{3.47}$$

From the equiprobable bitwise representation assumption we have $w_{\{b\}} \sim Be(0.5)$ is a Bernoulli random variable and $x_{\{-s-\}} = \frac{u_s}{2^{B_S}}$ where $u_s \sim U(0, 2^{B_S}-1)$ is a discrete uniform random variable. Hence, it can be shown that:

$$\mathrm{Var}(x_{\{-s-\}}w_{\{b\}}) = \frac{\left(1-2^{-B_S}\right)\left(5-2^{-B_S}\right)}{48}. \tag{3.48}$$

Substituting (3.48) and (3.47) into (3.46) yields (3.19) which concludes our proof.

Derivation of eq. (3.21) **and eq.** (3.22)**:**
First, (3.21) follows from combining (3.1) and (3.3) in a similar fashion as was done to obtain (3.46). Then, (3.22) is obtained from (3.4) by evaluating:

$$\mathbb{E}\left[\left(x_{\{-s-\}}w_{\{b\}}\right)^2\right] = \frac{1}{12}\left(1-2^{-B_S}\right)\left(2-2^{-B_S}\right).$$

This result itself is a consequence of the equiprobable bitwise representation assumption discussed in the derivation of (3.19).

Derivation of eq. (3.23)**:**
The proof can be obtained in an identical manner as that of Theorem 1 in [89], but by considering perturbations on the pre-activations. Specifically, it can be shown that:

$$p_m \leq \sum_{l=1}^{L} \sigma^2_{\eta_{y_l}} E_{Y,l}, \tag{3.49}$$

where $\sigma^2_{\eta_{y_l}}$ is the variance of a perturbation per pre-activation at layer l. In the IMC setup considered, we have

$$\sigma^2_{\eta_{y_l}} = \sigma^2_{q_{x\to y_l}} + \sigma^2_{q_{w\to y_l}} + \sigma^2_{q_{A\to y_l}} + \sigma^2_{\eta_{a\to y_l}}. \tag{3.50}$$

And (3.23) is obtained by substituting (3.50) into (3.49).

Derivation of eq. (3.28)**:**
The following observation from Table 3.2 is first made:

$$\sigma^2_{(\mathrm{OCC})}(B) \approx 1.46 \times 3.4^{-B}.$$

It follows, by virtue of (3.19), that, for FS/OCC/NE:

$$\sigma^2_{q_{A\to y_l}} E_{Y,l} = 1.46 \times 3.4^{-B_{A,l}} F^{(\mathrm{OCC})}_{Y,l}. \tag{3.51}$$

To achieve noise equalization, (3.51) should be identical for all $l = 1, \ldots, L$. This condition is equivalent to (3.28).

Derivation of eq. (3.30)**:** When B bit FR quantization is used, $\sigma^2_{q_{A_{s,b}}}$ in (3.17) evaluates to $N4^{-B}/12$. Consequently, for BS/FR/NE, we obtain:

$$\sigma^2_{q_{A\to y_l}} E_{Y,l} = 4^{-B_{A,l}} F^{(\mathrm{FR})}_{Y,l}. \tag{3.52}$$

As above, (3.52) is equalized across all layers $l = 1, \ldots, L$, which is equivalent to (3.30).

Derivation of eq. (3.34)**:**
For sufficiently large capacitance (e.g., $C_o > 1$ fF), (3.22) can be approxi-

mated as:

$$\sigma^2_{\eta_{a_{s,b}}} \approx \frac{1}{C_o} \times \frac{N\rho_1 \left(2 - 2^{-B_S}\right)}{12\left(1 - 2^{-B_S}\right)}. \tag{3.53}$$

Then, by virtue of (3.21) and (3.53), we obtain:

$$\sigma^2_{\eta_{a \to y_l}} E_{Y,l} = \frac{C_{o,l}}{G_{Y,l}}. \tag{3.54}$$

Equalizing (3.54) across all layers $l = 1, \ldots, L$ yields (3.34).

CHAPTER 4

FIXED-POINT TRAINING WITH CLOSE-TO-MINIMAL PRECISION

Chapters 2 and 3 have focused on network inference. In this chapter, we study precision requirements of fixed-point training where every tensor in the back-propagation loop is quantized. To do so, we address the challenges of quantization noise, inter-layer and intra-layer precision trade-offs, dynamic range, and stability. We derive a methodology to obtain close-to-minimal per-layer precision requirements for guaranteed statistical similarity between fixed-point and floating-point training. The methodology is applied to several benchmarks and fixed-point training is shown to achieve high fidelity to the baseline. The implication of our work in terms of training complexity is finally quantified.

4.1 Motivation

A fundamental problem contributing to the high computational and parameter complexity of DNNs is their realization using 32 bit floating-point (FL) arithmetic in GPUs and CPUs. Reduced-precision representations such as *quantized FL* (QFL) and *fixed-point* (FX) have been employed in various combinations to both training and inference. Many employ FX during inference but train in FL, e.g., fully binarized neural networks [51] use 1 bit FX in the forward inference path but the network is trained in 32 bit FL. Similarly, [72] employs 16 bit FX for all tensors except for the internal accumulators which use 32 bit FL, and 3-level QFL gradients were employed [121, 122] to accelerate training in a distributed setting. Note that while QFL reduces storage and communication costs, it does not reduce the computational complexity as the arithmetic remains in 32 bit FL.

Thus, none of the previous works address the fundamental problem of realizing *true fixed-point DNN training*, i.e., an SGD algorithm in which

all parameters/variables and all computations are implemented in FX with *minimum precision* required to *guarantee the network's inference/prediction accuracy* and *training convergence*. The reasons for this gap are numerous including: (1) quantization errors propagate to the network output thereby directly affecting its accuracy [45]; (2) precision requirements of different variables in a network are interdependent and involve hard-to-quantify trade-offs [89]; (3) proper quantization requires the knowledge of the dynamic range which may not be available [123]; and (4) quantization errors may accumulate during training and can lead to stability issues [72, 124].

Our work makes a major advance in closing this gap by proposing a systematic methodology to obtain *close-to-minimum* per-layer precision requirements of an FX network that guarantees statistical similarity with full precision training. In particular, we jointly address the challenges of quantization noise, inter-layer and intra-layer precision trade-offs, dynamic range, and stability. As in [89], we do assume that a fully-trained baseline FL network exists and one can observe its learning behavior. While, in principle, such assumption requires extra FL computation prior to FX training, it is to be noted that much of training is done in FL anyway. For instance, FL training is used in order to establish benchmarking baselines such as AlexNet [1], VGG-Net [119], and ResNet [3], to name a few. Even if that is not the case, in practice, this assumption can be accounted for via a warm-up FL training on a small held-out portion of the dataset [125].

Applying our methodology to three benchmarks reveals several lessons. First and foremost, our work shows that it is possible to FX quantize all variables including back-propagated gradients even though their dynamic range is unknown [25]. Second, we find that the per-layer weight precision requirements decrease from the input to the output while those of the activation gradients and weight accumulators increase. Furthermore, the precision requirements for residual networks are found to be uniform across layers. Finally, hyper-precision reduction techniques such as weight and activation binarization [51] or gradient ternarization [121] are not as efficient as our methodology since these do not address the fundamental problem of realizing true fixed-point DNN training.

We demonstrate FX training on three deep learning benchmarks (CIFAR-10, CIFAR-100, SVHN) achieving *high fidelity* to our FL baseline in that we observe no loss of accuracy higher than **0.56%** in all of our experiments.

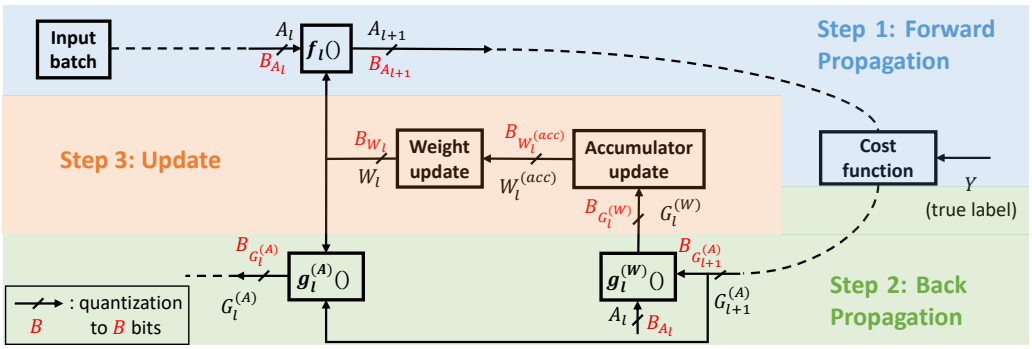


Figure 4.1: Problem setup: FX training at layer l of a DNN showing the quantized tensors and the associated precision configuration $C_l = (B_{W_l}, B_{A_l}, B_{G_l^{(W)}}, B_{G_{l+1}^{(A)}}, B_{W_l^{(acc)}})$.

Our precision assignment is further shown to be *within 1 bit per tensor* of the minimum. We show that our precision assignment methodology reduces representational, computational, and communication costs of training by up to **6**×, **8**×, and **4**×, respectively, compared to the FL baseline and related works.

4.2 Problem Setup, Notation, and Metrics

We consider an L-layer DNN deployed on a M-class classification task using the setup in Figure 4.1. We denote the *precision configuration* as the $L \times 5$ matrix $C = (B_{W_l}, B_{A_l}, B_{G_l^{(W)}}, B_{G_{l+1}^{(A)}}, B_{W_l^{(acc)}})_{l=1}^{L}$ whose l^{th} row consists of the precision (in bits) of weight W_l (B_{W_l}), activation A_l (B_{A_l}), weight gradient $G_l^{(W)}$ ($B_{G_l^{(W)}}$), activation gradient $G_{l+1}^{(A)}$ ($B_{G_{l+1}^{(A)}}$), and internal weight accumulator $W_l^{(acc)}$ ($B_{W_l^{(acc)}}$) tensors at layer l. This DNN quantization setup is summarized in Section 4.6.

4.2.1 Fixed-point Constraints and Definitions

We present definitions/constraints related to fixed-point arithmetic based on the design of fixed-point adaptive filters and signal processing systems [105]:

- A *signed* fixed-point scalar a with precision B_A and binary representation $R_A = (a_0, a_1, \ldots, a_{B_A-1}) \in \{0,1\}^{B_A}$ is equal to:

$a = r_A \left(-a_0 + \sum_{i=1}^{B_A-1} 2^{-i} a_i\right)$, where r_A is the predetermined dynamic range (PDR) of a. The PDR is constrained to be a **constant power of 2** to minimize hardware overhead.

- An *unsigned* fixed-point scalar a with precision B_A and binary representation $R_A = (a_0, a_1, \ldots, a_{B_A-1}) \in \{0,1\}^{B_A}$ is equal to:
  $a = r_A \sum_{i=0}^{B_A-1} 2^{-i} a_i$.

- A fixed-point scalar a is called *normalized* if $r_A = 1$.

- The precision B_A is determined as: $B_A = \log_2 \frac{r_A}{\Delta_A} + 1$, where Δ_A is the *quantization step size* which is the value of the *least significant bit (LSB)*.

- An *additive model* for quantization is assumed: $a = \tilde{a} + q_a$, where a is the fixed-point number obtained by quantizing the floating-point scalar $\tilde{a}$, q_a is a random variable uniformly distributed on the interval $\left[-\frac{\Delta_A}{2}, \frac{\Delta_A}{2}\right]$, and the quantization noise variance is $Var(q_a) = \frac{\Delta_A^2}{12}$. The notion of quantization noise is most useful when there is limited knowledge of the distribution of $\tilde{a}$.

- The *relative quantization bias* η_A is the offset: $\eta_A = \frac{|\Delta_A - \mu_A|}{\mu_A}$, where the first unbiased quantization level $\mu_A = \mathbb{E}\left[\tilde{a} \middle| \tilde{a} \in I_1\right]$ and $I_1 = \left[\frac{\Delta_A}{2}, \frac{3\Delta_A}{2}\right]$. The notion of quantization bias is useful when there is some knowledge of the distribution of $\tilde{a}$.

- The *reflected quantization noise variance* from a tensor T to a scalar $\alpha = f(T)$, for an arbitrary function $f()$, is: $V_{T\to\alpha} = E_{T\to\alpha} \frac{\Delta_T^2}{12}$, where Δ_T is the quantization step of T and $E_{T\to\alpha}$ is the *quantization noise gain* from T to α.

- The *clipping rate* β_T of a tensor T is the probability:
  $\beta_T = \Pr\left(\{|t| \geq r_T : t \in T\}\right)$, where r_T is the PDR of T.

4.2.2 Complexity Metrics

We use a set of metrics inspired by those introduced by [89] which have also been used by [67]. These metrics are algorithmic in nature which makes them easily reproducible.

- *Representational Cost* for weights ($\mathcal{C}_W$) and activations ($\mathcal{C}_A$):
  $\mathcal{C}_W = \sum_{l=1}^{L} |W_l| \left(B_{W_l} + B_{G_l^{(W)}} + B_{W_l^{(acc)}} \right)$ and
  $\mathcal{C}_A = \sum_{l=1}^{L} |A_l| \left(B_{A_l} + B_{G_{l+1}^{(A)}} \right)$, which equals the total number of bits needed to represent the weights, weight gradients, and internal weight accumulators (C_W), and those for activations and activation gradients (C_A). [1]

- *Computational Cost* of training:
  $\mathcal{C}_M = \sum_{l=1}^{L} |A_{l+1}| \, D_l \left(B_{W_l} B_{A_l} + B_{W_l} B_{G_{l+1}^{(A)}} + B_{A_l} B_{G_{l+1}^{(A)}} \right)$, where D_l is the dimensionality of the dot product needed to compute one output activation at layer l. This cost is a measure of the number of 1 bit full adders (FAs) utilized for all multiplications in one back-prop iteration. [2]

- *Communication Cost*: $\mathcal{C}_C = \sum_{l=1}^{L} |W_l| \, B_{G_l^{(W)}}$, which represents cost of communicating weight gradients in a distributed setting [121, 122].

4.3 Precision Assignment Methodology and Analysis

We aim to obtain a minimal or close-to-minimal precision configuration C_o of a FX network such that the mismatch probability $p_m = \Pr\{\hat{Y}_{fl} \neq \hat{Y}_{fx}\}$ between its predicted label ($\hat{Y}_{fx}$) and that of an associated FL network ($\hat{Y}_{fl}$) is bounded, and the convergence behavior of the two networks is similar.

Hence, we require that: (1) all quantization noise sources in the forward path contribute identically to the mismatch budget p_m [89], (2) the gradients be properly clipped in order to limit the dynamic range [123], (3) the accumulation of quantization noise bias in the weight updates be limited [72], (4) the quantization noise in activation gradients be limited as these are back-propagated to calculate the weight gradients, and (5) the precision of weight accumulators should be set so as to avoid premature stoppage of convergence [81]. The above insights can be formally described via the following five quantization criteria.

[1] We use the notation $|T|$ to denote the number of elements in tensor T. Unquantized tensors are assumed to have a 32 bit FL representation, which is the single-precision in a GPU.

[2] When considering 32 bit FL multiplications, we ignore the cost of exponent addition thereby favoring the FL (conventional) implementation. Boundary effects (in convolutions) are neglected.

Criterion 1. *Equalizing Feedforward Quantization Noise (EFQN) Criterion.* The reflected quantization noise variances onto the mismatch probability p_m from all feedforward weights ($\{V_{W_l \to p_m}\}_{l=1}^{L}$) and activations ($\{V_{A_l \to p_m}\}_{l=1}^{L}$) should be equal:

$$V_{W_1 \to p_m} = \ldots = V_{W_L \to p_m} = V_{A_1 \to p_m} = \ldots = V_{A_L \to p_m}.$$

Criterion 2. *Gradient Clipping (GC) Criterion.* The clipping rates of weight ($\{\beta_{G_l^{(W)}}\}_{l=1}^{L}$) and activation ($\{\beta_{G_{l+1}^{(A)}}\}_{l=1}^{L}$) gradients should be less than a maximum value β_0:

$$\beta_{G_l^{(W)}} < \beta_0 \quad \text{and} \quad \beta_{G_{l+1}^{(A)}} < \beta_0 \quad \text{for } l = 1 \ldots L.$$

Criterion 3. *Relative Quantization Bias (RQB) Criterion.* The relative quantization bias of weight gradients ($\{\eta_{G_l^{(W)}}\}_{l=1}^{L}$) should be less than a maximum value η_0:

$$\eta_{G_l^{(W)}} < \eta_0 \quad \text{for } l = 1 \ldots L.$$

Criterion 4. *Back-propagated Quantization Noise (BQN) Criterion.* The reflected quantization noise variance $V_{G_{l+1}^{(A)} \to \Sigma_l}$, i.e., the total sum of element-wise variances of $G_l^{(W)}$ reflected from quantizing $G_{l+1}^{(A)}$, should be less than $V_{G_l^{(W)} \to \Sigma_l}$:

$$V_{G_{l+1}^{(A)} \to \Sigma_l} \leq V_{G_l^{(W)} \to \Sigma_l} \quad \text{for } l = 1 \ldots L,$$

where Σ_l is the total sum of element-wise variances of $G_l^{(W)}$.

Criterion 5. *Accumulator Stopping (AS) Criterion.* The quantization noise of the internal accumulator should be zero, equivalently:

$$V_{W_l^{(acc)} \to \Sigma_l^{(acc)}} = 0 \quad \text{for } l = 1 \ldots L,$$

where $V_{W_l^{(acc)} \to \Sigma_l^{(acc)}}$ is the reflected quantization noise variance from $W_l^{(acc)}$ to $\Sigma_l^{(acc)}$, its total sum of element-wise variances.

Further explanations and motivations behind the above criteria are presented in Section 4.6. The following claim ensures the satisfiability of the

above criteria. This leads to closed form expressions for the precision requirements we are seeking and completes our methodology. The validity of the claim is proved in Section 4.6.

Claim 1. *Satisfiability of Quantization Criteria.* The five quantization criteria (EFQN, GC, RQB, BQN, AS) are satisfied if:

- The precisions B_{W_l} and B_{A_l} are set as follows:

$$B_{W_l} = rnd\left(\log_2\left(\sqrt{\frac{E_{W_l\to p_m}}{E^{(\min)}}}\right)\right) + B^{(\min)}$$
$$B_{A_l} = rnd\left(\log_2\left(\sqrt{\frac{E_{A_l\to p_m}}{E^{(\min)}}}\right)\right) + B^{(\min)} \tag{4.1}$$

  for $l = 1 \dots L$, where $rnd()$ denotes the rounding operation, $E_{W_l\to p_m}$ and $E_{A_l\to p_m}$ are the weight and activation quantization noise gains at layer l, respectively, $B^{(\min)}$ is a reference minimum precision, and $E^{(\min)} = \min\left(\{E_{W_l\to p_m}\}_{l=1}^{L}, \{E_{A_l\to p_m}\}_{l=1}^{L}\right)$.

- The weight and activation gradients PDRs are lower bounded as follows:

$$r_{G_l^{(W)}} \geq 2\sigma^{(\max)}_{G_l^{(W)}} \quad \text{and} \quad r_{G_{l+1}^{(A)}} \geq 4\sigma^{(\max)}_{G_{l+1}^{(A)}} \quad \text{for } l = 1 \dots L, \tag{4.2}$$

  where $\sigma^{(\max)}_{G_l^{(W)}}$ and $\sigma^{(\max)}_{G_{l+1}^{(A)}}$ are the largest recorded estimates of the weight and activation gradients standard deviations $\sigma_{G_l^{(W)}}$ and $\sigma_{G_{l+1}^{(A)}}$, respectively.

- The weight and activation gradients quantization step sizes are upper bounded as follows:

$$\Delta_{G_l^{(W)}} < \frac{\sigma^{(\min)}_{G_l^{(W)}}}{4} \quad \text{and} \quad \Delta_{G_{l+1}^{(A)}} < \frac{\Delta_{G_l^{(W)}}}{\sqrt{\lambda^{(\max)}_{G_{l+1}^{(A)}\to G_l^{(W)}}}} \left(\frac{\left|G_l^{(W)}\right|}{\left|G_{l+1}^{(A)}\right|}\right)^{1/4} \tag{4.3}$$

  for $l = 1 \dots L$, where $\sigma^{(\min)}_{G_l^{(W)}}$ is the smallest recorded estimate of $\sigma_{G_l^{(W)}}$ and $\lambda^{(\max)}_{G_{l+1}^{(A)}\to G_l^{(W)}}$ is the largest singular value of the square-Jacobian (Jacobian matrix with squared entries) of $G_l^{(W)}$ with respect to $G_{l+1}^{(A)}$.

- The accumulator PDR and step size satisfy:

$$r_{W_l^{(acc)}} \geq 2^{-B_{W_l}} \quad \& \quad \Delta_{W_l^{(acc)}} < \gamma^{(\min)} \Delta_{G_l^{(W)}} \quad \text{for } l = 1 \dots L, \tag{4.4}$$

  where $\gamma^{(\min)}$ is the smallest value of the learning rate used during training.

Practical considerations: Note that one of the $2L$ feedforward precisions will equal $B^{(\min)}$. The formulas to compute the quantization noise gains are given in Section 4.6 and require only one forward-backward pass on an estimation set. We would like the EFQN criterion to hold upon convergence; hence, (4.1) is computed using the converged model from the FL baseline. For backward signals, setting the values of PDR and LSB is sufficient to determine the precision using the identity $B_A = \log_2 \frac{r_A}{\Delta_A} + 1$, as explained in Section 4.2.1. As per Claim 1, estimates of the second-order statistics, e.g., $\sigma_{G_l^{(W)}}$ and $\sigma_{G_{l+1}^{(A)}}$, of the gradient tensors, are required. These are obtained via tensor spatial averaging, so that one estimate per tensor is required, and updated in a moving window fashion, as is done for normalization parameters in BatchNorm [47]. Furthermore, it might seem that computing the Jacobian in (4.3) is a difficult task; however, the values of its elements are already computed by the back-prop algorithm, requiring no additional computations (see Section 4.6). Thus, the Jacobians (at different layers) are also estimated during training. Due to the typical very large size of modern neural networks, we average the Jacobians spatially, i.e., the activations are aggregated across channels and mini-batches while weights are aggregated across filters. This is again inspired by the work on Batch Normalization [47] and makes the probed Jacobians much smaller.

4.4 Numerical Results

We conduct numerical simulations in order to illustrate the validity of the predicted precision configuration C_o and investigate its minimality and benefits. We employ three deep learning benchmarking datasets: CIFAR-10, CIFAR-100 [65], and SVHN [126]. All experiments were done using a Pascal P100 NVIDIA GPU. We train the following networks:

- CIFAR-10 ConvNet: a 9-layer convolutional neural network trained on

the CIFAR-10 dataset described as $2 \times (64C3) - MP2 - 2 \times (128C3) - MP2 - 2 \times (256C3) - 2 \times (512FC) - 10$ where $C3$ denotes 3×3 convolutions, $MP2$ denotes 2×2 max pooling operation, and FC denotes fully connected layers.

- SVHN ConvNet: the same network as the CIFAR-10 ConvNet, but trained on the SVHN dataset.

- CIFAR-10 ResNet: a wide deep residual network [127] with ResNet-20 architecture but having 8 times as many channels per layer compared to [3].

- CIFAR-100 ResNet: same network as CIFAR-10 ResNet save for the last layer to match the number of classes (100) in CIFAR-100.

A step by step description of the application of our method to the above four networks is provided in Section 4.6. We hope the inclusion of these steps would: (1) clarify any ambiguity the reader may have from the previous section and (2) facilitate the reproduction of our results.

4.4.1 Precision Configuration C_o and Convergence

The precision configuration C_o, with target $p_m \leq 1\%$, $\beta_0 \leq 5\%$, and $\eta_0 \leq 1\%$, via our proposed method is depicted in Figure 4.2 for each of the four networks considered. We observe that C_o is dependent on the network *type*. Indeed, the precisions of the two ConvNets follow similar trends as do those of the two ResNets. Furthermore, the following observations are made for the ConvNets:

- Weight precision B_{W_l} *decreases as depth increases*. This is consistent with the observation that weight perturbations in the earlier layers are the most destructive [92].

- The precisions of activation gradients ($B_{G_l^{(A)}}$) and internal weight accumulators ($B_{W_l^{(acc)}}$) *increase as depth increases* which we interpret as follows: (1) the back-propagation of gradients is the *dual* of the forward-propagation of activations, and (2) accumulators store the *most information* as their precision is the highest.

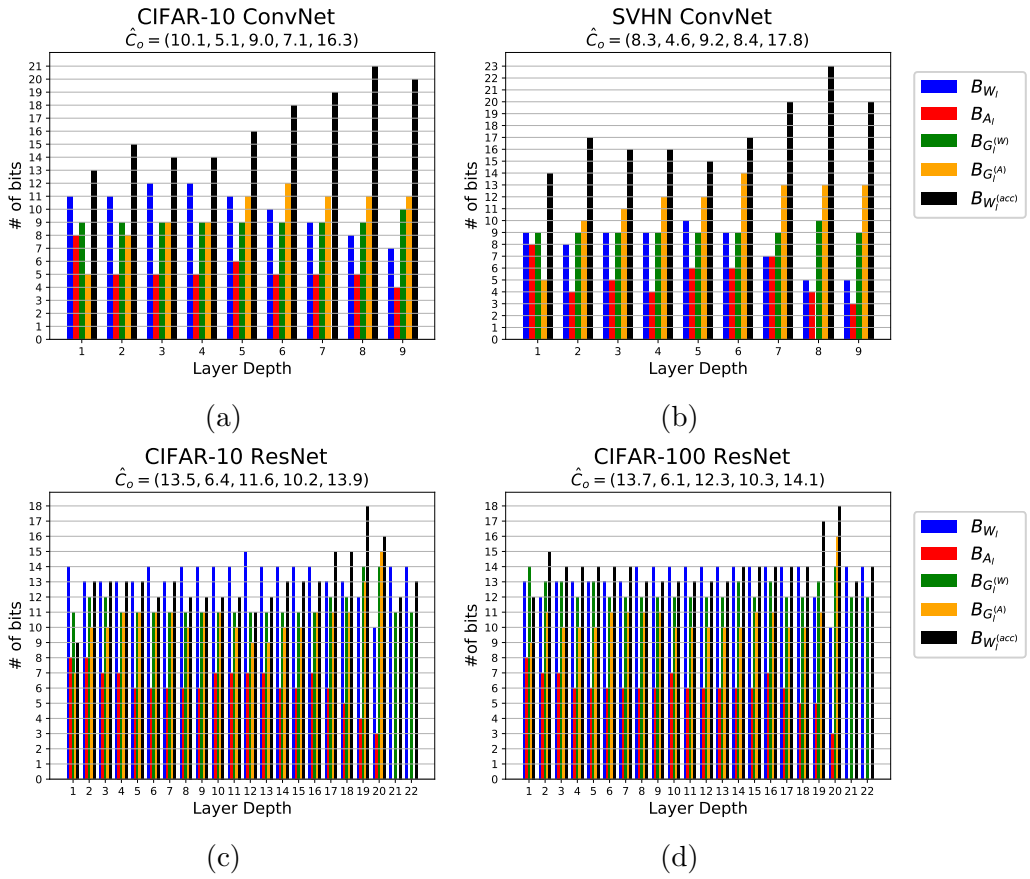


Figure 4.2: The predicted precision configurations C_o for the CIFAR-10 ConvNet (a), SVHN ConvNet (b), CIFAR-10 ResNet (c), and CIFAR-100 ResNet (d). For each network, the 5-tuple $\hat{C}_o$ represents the average number of bits per tensor type. For the ResNets, layer depths 21 and 22 correspond to the strided convolutions in the shortcut connections of residual blocks 4 and 7, respectively. Activation gradients go from layer 2 to $L+1$ and are "shifted to the left" in order to be aligned with the other tensors.

- The precisions of the weight gradients ($B_{G_l^{(W)}}$) and activations (B_{A_l}) are *relatively constant across layers*.

Interestingly, for ResNets, the precision is mostly uniform across the layers. Furthermore, the gap between $B_{W_l^{(acc)}}$ and the other precisions is not as pronounced as in the case of ConvNets. This suggests that information is spread equally among all signals which we speculate is due to the shortcut connections preventing the *shattering* of information [128].

FX training curves in Figure 4.3 indicate that C_o leads to *convergence* and consistently track FL curves with *close fidelity*. This validates our analysis

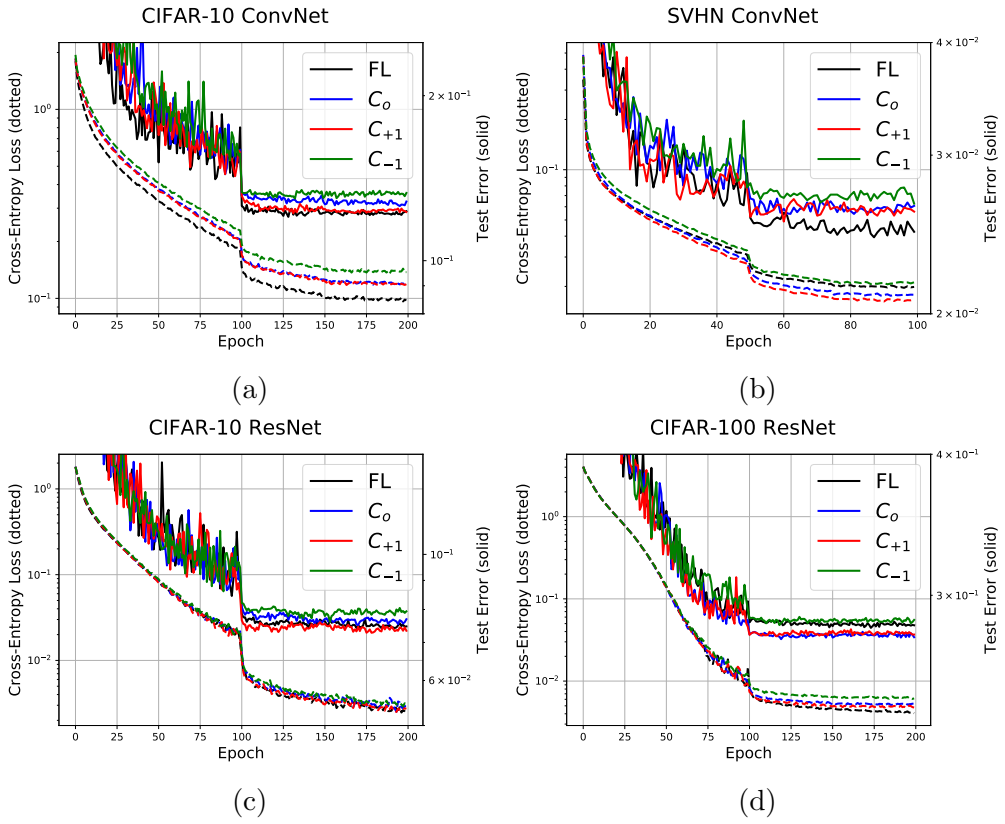


Figure 4.3: Convergence curves for the CIFAR-10 ConvNet (a), SVHN ConvNet (b), CIFAR-10 ResNet (c), and CIFAR-100 ResNet (d) including FL training as well as FX training with precision configurations C_o, C_1, and C_{-1}.

and justifies the choice of C_o.

4.4.2 Near Minimality of C_o

To determine that C_o is a close-to-minimal precision assignment, we compare it with: (a) $C_{+1} = C_o + \mathbf{1}_{L\times 5}$, and (b) $C_{-1} = C_o - \mathbf{1}_{L\times 5}$, where $\mathbf{1}_{L\times 5}$ is an $L \times 5$ matrix with each entry equal to 1[3], i.e., we perturb C_o by 1 bit in either direction. Figure 4.3 also contains the convergence curves for the two new configurations. As shown, C_{-1} always results in a noticeable gap compared to C_o for both the loss function (except for the CIFAR-10 ResNet) and the test error. Furthermore, C_{+1} offers no observable improvements over

[3]PDRs are unchanged across configurations, except for $r_{W_l^{(acc)}}$ as per (4.4).

Table 4.1: Complexity ($\mathcal{C}_W$, $\mathcal{C}_A$, $\mathcal{C}_M$, and $\mathcal{C}_C$) and accuracy (test error) for the floating-point (FL), fixed-point (FX) with precision configuration C_o, binarized network (BN), stochastic quantization (SQ), and TernGrad (TG) training schemes.

	$\mathcal{C}_W$ (10^6b)	$\mathcal{C}_A$ (10^6b)	$\mathcal{C}_M$ (10^9FA)	$\mathcal{C}_C$ (10^6b)	Test Error	$\mathcal{C}_W$ (10^6b)	$\mathcal{C}_A$ (10^6b)	$\mathcal{C}_M$ (10^9FA)	$\mathcal{C}_C$ (10^6b)	Test Error
	CIFAR-10 ConvNet					**SVHN ConvNet**				
FL	148	9.3	94.4	49	12.02%	148	9.3	94.4	49	**2.43**%
FX (C_o)	**56.5**	**1.7**	11.9	14	12.58%	**54.3**	**1.9**	10.5	14	2.58%
BN	100	4.7	**2.8**	49	18.50%	100	4.7	**2.8**	49	3.60%
SQ	78.8	**1.7**	11.9	14	**11.32**%	76.3	**1.9**	10.5	14	2.73%
TG	102	9.3	94.4	**3.1**	12.49%	102	9.3	94.4	**3.1**	3.65%
	CIFAR-10 ResNet					**CIFAR-100 ResNet**				
FL	1784	96	4319	596	7.42%	1789	97	4319	597	28.06%
FX (C_o)	**726**	**25**	785	216	7.51%	**750**	**25**	776	216	**27.43**%
BN	1208	50	**128**	596	**7.24**%	1211	50	**128**	597	29.35%
SQ	1062	**25**	785	216	7.42%	1081	**25**	776	216	28.03%
TG	1227	96	4319	**37.3**	7.94%	1230	97	4319	**37.3**	30.62%

C_o (except for the test error of CIFAR-10 ConvNet). These results support our contention that C_o is *close-to-minimal* in that increasing the precision above C_o leads to diminishing returns while reducing precision below C_o leads to a noticeable degradation in accuracy. Additional experimental results provided in Section 4.6 support our contention regarding the near minimality of C_o. Furthermore, by studying the impact of quantizing specific tensors we determine that that the accuracy is most sensitive to the precision assigned to weights and activation gradients.

4.4.3 Complexity vs. Accuracy

We would like to quantify the reduction in training cost and expense in terms of accuracy resulting from our proposed method and compare them with those of other methods. Importantly, for a fair comparison, the *same* network architecture and training procedure are used. We report $\mathcal{C}_W$, $\mathcal{C}_A$, $\mathcal{C}_M$, $\mathcal{C}_C$, and test error, for each of the four networks considered for the following training methods:

- Baseline FL training and FX training using C_o.
- Binarized network (BN) training, where feedforward weights and activations are binary (constrained to ± 1) while gradients and accumulators are in floating-point and activation gradients are back-propagated

via the straight through estimator [50] as was done in [51].

- Fixed-point training with stochastic quantization (SQ). As was done in [72], we quantize feedforward weights and activations as well as all gradients, but accumulators are kept in floating-point. The precision configuration (excluding accumulators) is inherited from C_o (hence we determine exactly how much stochastic quantization helps).

- Training with ternarized gradients (TG) as was done in TernGrad [121]. All computations are done in floating-point but weight gradients are ternarized according to the instantaneous tensor spatial standard deviations $\{-2.5\sigma, 0, 2.5\sigma\}$ as was suggested by [121]. To compute costs, we assume all weight gradients use two bits although they are not really fixed-point and do require computation of 32 bit floating-point scalars for every tensor.

The comparison is presented in Table 4.1. The first observation is a massive complexity reduction compared to FL. For instance, for the CIFAR-10 ConvNet, the complexity reduction is $2.6\times$ $(= 148/56.5)$, $5.5\times$ $(= 9.3/1.7)$, $7.9\times$ $(= 94.4/11.9)$, and $3.5\times$ $(= 49/14)$ for $\mathcal{C}_W$, $\mathcal{C}_A$, $\mathcal{C}_M$, and $\mathcal{C}_C$, respectively. Similar trends are observed for the other four networks. Such complexity reduction comes at the expense of no more than **0.56%** increase in test error. For the CIFAR-100 network, the accuracy when training in fixed-point is even better than that of the baseline.

The representational and communication costs of BN are significantly greater than those of FX because the gradients and accumulators are kept in full precision, which masks the benefits of binarizing feedforward tensors. However, benefits are noticeable when considering the computational cost which is lowest as binarization eliminates multiplications. Furthermore, binarization causes a severe accuracy drop for the ConvNets but surprisingly not for the ResNets. We speculate that this is due to the high-dimensional geometry of ResNets [129].

As for SQ, since C_o was inherited, all costs are identical to FX, save for $\mathcal{C}_W$ which is larger due to full precision accumulators. Furthermore, SQ has a positive effect only on the CIFAR-10 ConvNet where it clearly acted as a regularizer.

TG does not provide complexity reductions in terms of representational

and computational costs which is expected as it only compresses weight gradients. Additionally, the resulting accuracy is slightly worse than that of all other considered schemes, including FX. Naturally, it has the lowest communication cost as weight gradients are quantized to just 2 bit.

4.5 Discussion

4.5.1 Comparison with quantized training methods

Many works have addressed the general problem of reduced precision deep learning. Finite precision training was explored in [72] which employed stochastic quantization in order to counter quantization bias accumulation in the weight updates. This was done by quantizing all tensors to 16 bit FX, except for the internal accumulators which were stored in a 32 bit floating-point format. An important distinction our work makes is the circumvention of the overhead of implementing stochastic quantization [51]. Similarly, DoReFa-Net [53] stores internal weight representations in 32 bit FL, but quantizes the remaining tensors more aggressively. Thus arises the need to re-scale and re-compute in floating-point format, which our work avoids. Finally, [25] suggests a new number format – Flexpoint – and was able to train neural networks using slightly more than 16 bit per tensor element, with 5 shared exponent bits and a per-tensor dynamic range tracking algorithm. Such tracking causes a hardware overhead bypassed by our work since the arithmetic is purely FX. Augmenting Flexpoint with stochastic quantization effectively results in WAGE [73], and enables integer quantization of each tensor.

As seen above, none of the prior works address the problem of predicting precision requirements of all training signals. Furthermore, the choice of precision is made in an ad-hoc manner. In contrast, we propose a *systematic methodology* to determine *close-to-minimal* precision requirements for FX-only training of deep neural networks.

4.5.2 Conclusion

In this chapter, we have presented a study of precision requirements in a typical back-propagation based training procedure of neural networks. Using a set of quantization criteria, we have presented a precision assignment methodology for which FX training is made statistically similar to the FL baseline, known to converge a priori. We realized FX training of four networks on the CIFAR-10, CIFAR-100, and SVHN datasets and quantified the associated complexity reduction gains in terms costs of training. We also showed that our precision assignment is nearly minimal.

The presented work relies on the statistics of all tensors being quantized *during training*. This necessitates an initial baseline run in floating-point which can be costly. An open problem is to predict a suitable precision configuration by only observing the data statistics and the network architecture. Future work can leverage the analysis presented in this chapter to enhance the effectiveness of other network complexity reduction approaches. For instance, weight pruning can be viewed as a coarse quantization process (quantize to zero) and thus can potentially be done in a targeted manner by leveraging the information provided by noise gains. Furthermore, parameter sharing and clustering can be viewed as a form of vector quantization which presents yet another opportunity to leverage our method for complexity reduction.

4.6 Addendum: Proofs and Additional Results

Summary of Quantization Setup

The quantization setup depicted in Figure 4.1 is summarized as follows:

- Feedforward computation at layer l:

$$A_{l+1} = f_l(A_l, W_l),$$

  where $f_l()$ is the function implemented at layer l, A_l (A_{l+1}) is the activation tensor at layer l ($l+1$) quantized to a normalized unsigned fixed-point format with precision B_{A_l} ($B_{A_{l+1}}$), and W_l is the weight

tensor at layer l quantized to a normalized signed fixed-point format with precision B_{W_l}. We further assume the use of a ReLU-like activation function with a clipping level of 2 and a max-norm constraint on the weights which are clipped between $[-1, 1]$ at every iteration.

- Back-propagation of activation gradients at layer l:

$$G_l^{(A)} = g_l^{(A)}(W_l, G_{l+1}^{(A)}),$$

where $g_l()^{(A)}$ is the function that back-propagates the activation gradients at layer l, $G_l^{(A)}$ ($G_{l+1}^{(A)}$) is the activation gradient tensor at layer l ($l+1$) quantized to a signed fixed-point format with precision $B_{G_l^{(A)}}$ ($B_{G_{l+1}^{(A)}}$).

- Back-propagation of weight gradient tensor $G_l^{(W)}$ at layer l:

$$G_l^{(W)} = g_l^{(W)}(A_l, G_{l+1}^{(A)}),$$

where $g_l^{(W)}()$ is the function that back-propagates the weight gradients at layer l, and $G_l^{(W)}$ is quantized to a signed fixed-point format with precision $B_{G_l^{(W)}}$.

- Internal weight accumulator update at layer l:

$$W_l^{(acc)} = U(W_l^{(acc)}, G_l^{(W)}, \gamma),$$

where $U()$ is the update function, γ is the learning rate, and $W_l^{(acc)}$ is the internal weight accumulator tensor at layer l quantized to signed fixed-point with precision $B_{W_l^{(acc)}}$. Note that, for the next iteration, W_l is directly obtained from $W_l^{(acc)}$ via quantization to B_{W_l} bit.

Further Explanations and Motivations behind Quantization Criteria

Criterion 1 (EFQN) is used to ensure that all feedforward quantization noise sources contribute equally to the p_m budget. Indeed, if one of the $2L$ reflected quantization noise variances from the feedforward tensors onto p_m, say $V_{W_i \to p_m}$ for $i \in \{1, \ldots, L\}$, largely dominates all others, it would

imply that all tensors but W_i are overly quantized. It would therefore be necessary to either increase the precision of W_i or decrease the precisions of all other tensors. The application of Criterion 1 (EFQN) through the closed form expression (4.1) in Claim 1 solves this issue avoiding the need for a trial-and-error approach.

Because FX numbers require a constant PDR, clipping of gradients is needed since their dynamic range is arbitrary. Ideally, a very small PDR would be preferred in order to obtain quantization steps of small magnitude, and hence less quantization noise. We can draw parallels from signal processing theory, where it is known that for a given quantizer, the signal-to-quantization-noise ratio (SQNR) is equal to $SQNR(dB) = 6B + 4.78 - PAR$ where PAR is the peak-to-average ratio, proportional to the PDR. Thus, we would like to reduce the PDR as much as possible in order to increase the SQNR for a given precision. However, this comes at the risk of overflows (due to clipping). **Criterion 2 (GC)** addresses this trade-off between quantization noise and overflow errors.

Since the back-propagation training procedure is an iterative one, it is important to ensure that any form of bias does not corrupt the weight update accumulation in a positive feedback manner. FX quantization, being a uniform one, is likely to induce such bias when quantized quantities, most notably gradients, are not uniformly distributed. **Criterion 3 (RQB)** addresses this issue by using η as proxy to this bias accumulation as a function of quantization step size and ensuring that its worst-case value is small in magnitude.

Criterion 4 (BQN) is in fact an extension of Criterion 1 (EFQN), but for the back-propagation phase. Indeed, once the precision (and hence quantization noise) of weight gradients is set as per Criterion 3 (RQB), it is needed to ensure that the quantization noise source at the activation gradients would not contribute more noise to the updates. This criterion sets the quantization step of the activation gradients.

Criterion 5 (AS) ties together feedforward and gradient precisions through the weight accumulators. It is required to increment/decrement the feedforward weights whenever the accumulated updates cross over the weight quantization threshold. This is used to set the PDR of the weight accumulators. Furthermore, since the precision of weight gradients has already been designed to account for quantization noise (through Criteria 2-4), the criterion

requires that the accumulators do not cause additional noise.

Proof of Claim 1

The validity of Claim 1 is derived from the following five lemmas. Note that each lemma addresses the satisfiability of one of the five quantization criteria presented in this chapter and corresponds to part of Claim 1.

Lemma 3. *The EFQN criterion holds if the precisions B_{W_l} and B_{A_l} are set as follows:*

$$B_{W_l} = rnd\left(\log_2\left(\sqrt{\frac{E_{W_l\to p_m}}{E^{(\min)}}}\right)\right) + B^{(\min)}$$
$$B_{A_l} = rnd\left(\log_2\left(\sqrt{\frac{E_{A_l\to p_m}}{E^{(\min)}}}\right)\right) + B^{(\min)}$$

for $l = 1 \dots L$, where $rnd()$ denotes the rounding operation, $B^{(\min)}$ is a reference minimum precision, and $E^{(\min)}$ is given by:

$$E^{(\min)} = \min\left(\{E_{W_l\to p_m}\}_{l=1}^{L}, \{E_{A_l\to p_m}\}_{l=1}^{L}\right). \tag{4.5}$$

Proof. By definition of the reflected quantization noise variance, the EFQN, by definition, is satisfied if:

$$\frac{\Delta_{W_1}^2}{12}E_{W_1\to p_m} = \dots = \frac{\Delta_{W_L}^2}{12}E_{W_L\to p_m} = \frac{\Delta_{A_1}^2}{12}E_{A_1\to p_m} = \dots = \frac{\Delta_{A_L}^2}{12}E_{A_L\to p_m},$$

where the quantization noise gains are given by:

$$E_{W_l\to p_m} = \mathbb{E}\left[\sum_{\substack{i=1\\ i\neq\hat{Y}_{fl}}}^{M}\frac{\sum\limits_{w\in W_l}\left|\frac{\partial(Z_i - Z_{\hat{Y}_{fl}})}{\partial w}\right|^2}{2|Z_i - Z_{\hat{Y}_{fl}}|^2}\right]; E_{A_l\to p_m} = \mathbb{E}\left[\sum_{\substack{i=1\\ i\neq\hat{Y}_{fl}}}^{M}\frac{\sum\limits_{a\in A_l}\left|\frac{\partial(Z_i - Z_{\hat{Y}_{fl}})}{\partial a}\right|^2}{2|Z_i - Z_{\hat{Y}_{fl}}|^2}\right] \tag{4.6}$$

for $l = 1 \dots L$, where $\{Z_i\}_{i=1}^{M}$ are the soft outputs and $Z_{\hat{Y}_{fl}}$ is the soft output corresponding to $\hat{Y}_{fl}$. The expressions for these quantization gains are obtained by linearly expanding (across layers) those used in [89]. Note that a second-order upper bound is used as a surrogate expression for p_m.

From the definition of quantization step size, the above is equivalent to:

$$2^{-2B_{W_1}} E_{W_1 \to p_m} = \ldots = 2^{-2B_{A_1}} E_{A_1 \to p_m} = \ldots = 2^{-2B_{A_L}} E_{A_L \to p_m}.$$

Let $E^{(\min)}$ be as defined in (4.5):

$$E^{(\min)} = \min \left(\{E_{W_l \to p_m}\}_{l=1}^{L}, \{E_{A_l \to p_m}\}_{l=1}^{L} \right).$$

We can divide each term by $E^{(\min)}$:

$$2^{-2B_{W_1}} \frac{E_{W_1 \to p_m}}{E^{(\min)}} = \ldots = 2^{-2B_{A_1}} \frac{E_{A_1 \to p_m}}{E^{(\min)}} = \ldots = 2^{-2B_{A_L}} \frac{E_{A_L \to p_m}}{E^{(\min)}},$$

where each term is positive, so that we can take square roots and logarithms such that:

$$\begin{aligned} &B_{W_1} - \log_2 \left(\sqrt{\frac{E_{W_1 \to p_m}}{E^{(\min)}}} \right) = \ldots = B_{W_L} - \log_2 \left(\sqrt{\frac{E_{W_L \to p_m}}{E^{(\min)}}} \right) \\ =&B_{A_1} - \log_2 \left(\sqrt{\frac{E_{A_1 \to p_m}}{E^{(\min)}}} \right) = \ldots = B_{A_L} - \log_2 \left(\sqrt{\frac{E_{A_L \to p_m}}{E^{(\min)}}} \right). \end{aligned}$$

Thus we equate all of the above to a reference precision $B^{(\min)}$ yielding:

$$B_{W_l} = \log_2 \left(\sqrt{\frac{E_{W_l \to p_m}}{E^{(\min)}}} \right) + B^{(\min)} \quad \text{and} \quad B_{A_l} = \log_2 \left(\sqrt{\frac{E_{A_l \to p_m}}{E^{(\min)}}} \right) + B^{(\min)}$$

for $l = 1 \ldots L$. Note that because $E^{(\min)}$ is the least quantization noise gain, it is equal to one of the above quantization noise gains so that the corresponding precision actually equates $B^{(\min)}$. As precisions must be integer valued, each of $B^{(\min)}$, $\{B_{W_l}\}_{l=1}^{L}$, and $\{B_{A_l}\}_{l=1}^{L}$ have to be integers, and thus a rounding operation is to be applied on all logarithm terms. Doing so results in (4.1) from Lemma 3 which completes this proof. □

Lemma 4. *The GC criterion holds for $\beta_0 = 5\%$ provided the weight and activation gradient pre-defined dynamic ranges (PDRs) are lower bounded as follows:*

$$r_{G_l^{(W)}} \geq 2\sigma^{(\max)}_{G_l^{(W)}} \quad \textit{and} \quad r_{G_{l+1}^{(A)}} \geq 4\sigma^{(\max)}_{G_{l+1}^{(A)}} \quad \textit{for } l = 1 \ldots L,$$

where $\sigma^{(\max)}_{G_l^{(W)}}$ *and* $\sigma^{(\max)}_{G_{l+1}^{(A)}}$ *are the largest ever recorded estimates of the weight and activation gradient standard deviations* $\sigma_{G_l^{(W)}}$ *and* $\sigma_{G_{l+1}^{(A)}}$*, respectively.*

Proof. Let us consider the case of weight gradients. The GC criterion by definition requires:

$$\beta_{G_l^{(W)}} = \Pr\left(\left\{|g| \geq r_{G_l^{(W)}} : g \in G_l^{(W)}\right\}\right) < 0.05.$$

Typically, weight gradients are obtained by computing the derivatives of a loss function with respect to a mini-batch. By linearity of derivatives, weight gradients are themselves averages of instantaneous derivatives and are hence expected to follow a Gaussian distribution by application of the Central Limit Theorem. Furthermore, the gradient mean was estimated during baseline training and was found to oscillate around zero.

Thus

$$\beta_{G_l^{(W)}} = 2Q\left(\frac{r_{G_l^{(W)}}}{\sigma_{G_l^{(W)}}}\right),$$

where we used the fact that a Gaussian distribution is symmetric and $Q()$ is the elementary Q-function, which is a decreasing function. Thus, in the worst case, we have:

$$\beta_{G_l^{(W)}} \leq 2Q\left(\frac{r_{G_l^{(W)}}}{\sigma^{(\max)}_{G_l^{(W)}}}\right).$$

Hence, for a PDR as suggested by the lower bound in (4.2)

$$r_{G_l^{(W)}} \geq 2\sigma^{(\max)}_{G_l^{(W)}}$$

in Lemma 4, we obtain the upper bound:

$$\beta_{G_l^{(W)}} \leq 2Q(2) = 0.044 < 0.05,$$

which means the GC criterion holds and completes the proof.

For activation gradients, the same reasoning applies, but the choice of a larger PDR in (4.2)

$$r_{G_{l+1}^{(A)}} \geq 4\sigma^{(\max)}_{G_{l+1}^{(A)}}$$

than for weight gradients is due to the fact that the true dynamic range of the activation gradients is larger than the value indicated by the second moment.

This stems from the use of activation functions such as ReLU which make the activation gradients sparse. We also recommend increasing the PDR even more when using regularizers that sparsify gradients such as Dropout [44] or Maxout [42]. □

Lemma 5. *The RQB criterion holds for $\eta_0 = 1\%$ provided the weight gradient quantization step size is upper bounded as follows:*

$$\Delta_{G_l^{(W)}} < \frac{\sigma^{(\min)}_{G_l^{(W)}}}{4} \quad \textit{for } l = 1 \dots L,$$

where $\sigma^{(\min)}_{G_l^{(W)}}$ is the smallest ever recorded estimate of $\sigma_{G_l^{(W)}}$.

Proof. For the Gaussian distributed (see proof of Lemma 4) weight gradient at layer l, the true mean conditioned on the first non-zero quantization region is given by:

$$\begin{aligned}
\mu_{G_l^{(W)}} &= \frac{\int_{\frac{\Delta_{G_l^{(W)}}}{2}}^{\frac{3\Delta_{G_l^{(W)}}}{2}} x \exp\left(-\frac{x^2}{2\sigma^2_{G_l^{(W)}}}\right) dx}{\left(Q\left(\frac{\Delta_{G_l^{(W)}}}{2\sigma_{G_l^{(W)}}}\right) - Q\left(\frac{3\Delta_{G_l^{(W)}}}{2\sigma_{G_l^{(W)}}}\right)\right)\sqrt{2\pi\sigma^2_{G_l^{(W)}}}} \\
&= \frac{\sigma_{G_l^{(W)}}\left(\exp\left(-\frac{\Delta^2_{G_l^{(W)}}}{8\sigma^2_{G_l^{(W)}}}\right) - \exp\left(-\frac{9\Delta^2_{G_l^{(W)}}}{8\sigma^2_{G_l^{(W)}}}\right)\right)}{\left(Q\left(\frac{\Delta_{G_l^{(W)}}}{2\sigma_{G_l^{(W)}}}\right) - Q\left(\frac{3\Delta_{G_l^{(W)}}}{2\sigma_{G_l^{(W)}}}\right)\right)\sqrt{2\pi}},
\end{aligned}$$

where $\sigma_{G_l^{(W)}}$ is the standard deviation of $G_l^{(W)}$. By substituting $\Delta_{G_l^{(W)}} = \frac{\sigma_{G_l^{(W)}}}{4}$ into the above expression of $\mu_{G_l^{(W)}}$ and plugging in the definition of relative quantization bias, we obtain:

$$\eta_{G_l^{(W)}} = \frac{\left|\Delta_{G_l^{(W)}} - \mu_{G_l^{(W)}}\right|}{\mu_{G_l^{(W)}}} = 0.4\% < 1\%.$$

Hence, this choice of the quantization step satisfies the RQB. In order to ensure the RQB holds throughout training, $\sigma^{(\min)}_{G_l^{(W)}}$ is used in Lemma 5. This completes the proof. □

Lemma 6. *The BQN criterion holds provided the activation gradient quantization step size is upper bounded as follows:*

$$\Delta_{G_{l+1}^{(A)}} < \frac{\Delta_{G_l^{(W)}}}{\sqrt{\lambda^{(\max)}_{G_{l+1}^{(A)} \to G_l^{(W)}}}} \left(\frac{\left| G_l^{(W)} \right|}{\left| G_{l+1}^{(A)} \right|} \right)^{1/4} \qquad \textit{for } l = 1 \dots L,$$

where $\lambda^{(\max)}_{G_{l+1}^{(A)} \to G_l^{(W)}}$*, the largest singular value of the square-Jacobian (Jacobian matrix with squared entries) of* $G_l^{(W)}$ *with respect to* $G_{l+1}^{(A)}$*.*

Proof. Let us unroll $G_l^{(W)}$ and $G_{l+1}^{(A)}$ to vectors of size $\left| G_l^{(W)} \right|$ and $\left| G_{l+1}^{(A)} \right|$, respectively. The element-wise quantization noise variance of each weight gradient is $\frac{\Delta^2_{G_l^{(W)}}}{12}$. Therefore we have:

$$V_{G^{(W)} \to \Sigma_l} = \left| G_l^{(W)} \right| \frac{\Delta^2_{G_l^{(W)}}}{12}.$$

The reflected quantization noise variance from an activation gradient $g_a \in G_{l+1}^{(A)}$ onto a weight gradient $g_w \in G_l^{(W)}$ is

$$\left| \frac{\partial g_w}{\partial g_a} \right|^2 \frac{\Delta^2_{G_{l+1}^{(A)}}}{12},$$

where cross products of quantization noise are neglected [89]. Hence, the reflected quantization noise variance element-wise from $G_{l+1}^{(A)}$ onto $G_l^{(W)}$ is given by:

$$\frac{\Delta^2_{G_{l+1}^{(A)}}}{12} J_{G_{l+1}^{(A)} \to G_l^{(W)}} \mathbf{1}_{\left| G_{l+1}^{(A)} \right|},$$

where $J_{G_{l+1}^{(A)} \to G_l^{(W)}}$ is the square-Jacobian of $G_l^{(W)}$ with respect to $G_{l+1}^{(A)}$ and $\mathbf{1}$ denotes the all one vector with size denoted by its subscript. Hence, we

have:

$$
\begin{aligned}
V_{G_{l+1}^{(A)} \to \Sigma_l} &= \frac{\Delta^2_{G_{l+1}^{(A)}}}{12} \left(J_{G_{l+1}^{(A)} \to G_l^{(W)}} \mathbf{1}_{\left|G_{l+1}^{(A)}\right|} \right)^T \mathbf{1}_{\left|G_l^{(W)}\right|} \\
&\leq \frac{\Delta^2_{G_l^{(A)}}}{12} \left\| J_{G_{l+1}^{(A)} \to G_l^{(W)}} \mathbf{1}_{\left|G_{l+1}^{(A)}\right|} \right\| \left\| \mathbf{1}_{\left|G_l^{(W)}\right|} \right\| \\
&\leq \sqrt{\left|G_l^{(W)}\right|} \frac{\Delta^2_{G_{l+1}^{(A)}}}{12} \left\| J_{G_{l+1}^{(A)} \to G_l^{(W)}} \right\| \left\| \mathbf{1}_{\left|G_{l+1}^{(A)}\right|} \right\| \\
&\leq \lambda^{(\max)}_{G_{l+1}^{(A)} \to G_l^{(W)}} \sqrt{\left|G_{l+1}^{(A)}\right| \left|G_l^{(W)}\right|} \frac{\Delta^2_{G_{l+1}^{(A)}}}{12},
\end{aligned}
$$

where we used the Cauchy-Schwarz inequality and the spectral norm of a matrix. Next we set this upper bound on $V_{G_{l+1}^{(A)} \to \Sigma_l}$ to be less than the value of $V_{G_l^{(W)} \to \Sigma_l}$ determined above. This condition, by definition, is enough to satisfy the BQN criterion. Rearranging terms yields Lemma 6 which completes the proof.

Earlier in the chapter, it was mentioned that each entry in the Jacobian matrix above is already computed by the back-propagation algorithm. We now explain how. Let us denote the instantaneous loss function being minimized by ξ. Note that each entry of $J_{G_{l+1}^{(A)} \to G_l^{(W)}}$ is of the form $\left| \frac{\partial g_w}{\partial g_a^{(0)}} \right|^2$ where $g_w = \frac{\partial \xi}{\partial w}$ with $w \in W_l$ and $g_a^{(0)} = \frac{\partial \xi}{\partial a^{(0)}}$ with $a^{(0)} \in A_{l+1}$. The back-propagation algorithm computes g_w using the chain rule as follows:

$$
g_w = \frac{\partial \xi}{\partial w} = \sum_{a^{(i)} \in A_{l+1}} \frac{\partial \xi}{\partial a^{(i)}} \frac{\partial a^{(i)}}{\partial w}.
$$

In particular, note that $g_a^{(0)}$ appears only once in the summation above and is multiplied by $\frac{\partial a^{(0)}}{\partial w}$. Thus $\frac{\partial g_w}{\partial g_a^{(0)}} = \frac{\partial a^{(0)}}{\partial w}$. This establishes that each entry of the Jacobian matrix is already computed via the back-propagation algorithm.

□

Lemma 7. *The AS criterion holds provided the accumulator PDR and quantization step size satisfy:*

$$
r_{W_l^{(acc)}} \geq 2^{-B_{W_l}} \quad \textit{and} \quad \Delta_{W_l^{(acc)}} < \gamma^{(\min)} \Delta_{G_l^{(W)}} \quad \textit{for } l = 1 \ldots L,
$$

where $\gamma^{(\min)}$ is the smallest value of the learning rate used during training.

Proof. The lower bound on the PDR of the weight accumulator, given by

$$r_{W_l^{(acc)}} \geq 2^{-B_{W_l}}$$

for $l = 1 \dots L$, ensures that updates are able to cross over the feedforward weight quantization threshold so that it can be updated. Additionally, the lower bound on the quantization step size, given by

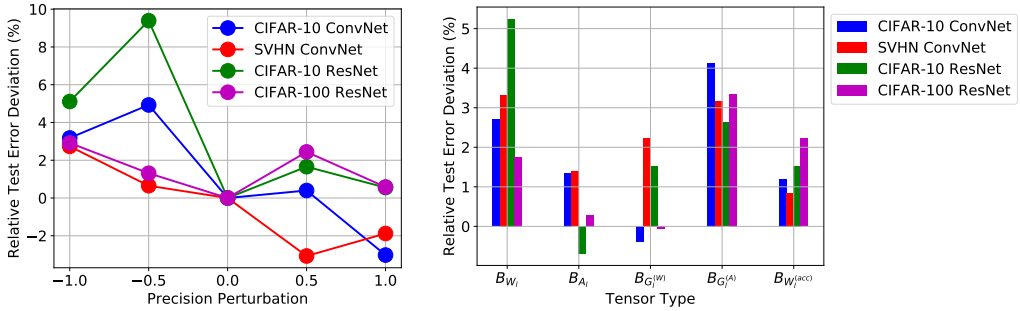
$$\Delta_{W_l^{(acc)}} < \gamma^{(\min)} \Delta_{G_l^{(W)}}$$

for $l = 1 \dots L$, simply ensures that the internal weight accumulator overlaps with the least significant part of the representation of the weight gradient multiplied by the learning rate. Thus, the quantization noise of the internal accumulator is zero, or equivalently,

$$V_{W_l^{(acc)} \to \Sigma_l^{(acc)}} = 0 \quad \text{for } l = 1 \dots L$$

which, by definition, is enough for the AS criterion to hold. Note that this criterion applies to the Vanilla-SGD learning rule (which was used in our experiments). Future work includes extending this criterion to other learning rules such as momentum and ADAM. □

We close this section by discussing the approximation made by invoking the Central Limit Theorem (CLT) in the proofs of Lemmas 4 and 5. This approximation was made because, typically, a back-propagation iteration computes gradients of a loss function being averaged over a mini-batch of samples. By linearity of derivatives, the gradients themselves are averages, which warrants the invocation of the CLT. However, the CLT is an asymptotic result which might be imprecise for a finite number of samples. In typical training of neural networks, the number of samples, or mini-batch size, is in the range of hundreds or thousands [10]. It is therefore important to quantify the preciseness, or lack thereof, of the CLT approximation. On way to do so is via the Berry-Essen Theorem which considers the average of n independent, identically distributed random variables with finite absolute third moment ρ and standard deviation σ. The worst-case deviation of the cumulative distribution of the true average from the of the approximated Gaussian random variable (via the CLT), also known as the Kolmogorov-Smirnov dis-



(a) (b)

Figure 4.4: Additional experiments on minimality and sensitivity of C_o: relative test error deviation with respect to C_o as a function of (a) random fractional precision perturbations, and (b) 1 bit precision reduction per tensor type.

tance, KS, is upper bounded as follows: $KS < \frac{C\rho}{\sqrt{n}\sigma^3}$, where $C < 0.4785$ [130]. Observe that the quantity $\frac{\rho}{\sigma^3}$ is data dependent. To estimate this quantity, we performed a forward-backward pass for all training samples at the start of each epoch for our four networks considered. The statistics ρ and σ were estimated by spatial (over tensors) and sample (over training samples) averages. The maximum value of the ratio $\frac{\rho}{\sigma^3}$ for all gradient tensors was found to be 2.8097. The mini-batch size we used in all our experiments was 256. Hence, we claim that the CLT approximation in Lemmas 4 and 5 is valid in our context up to a worst-case Kolmogorov-Smirnov distance of $KS < \frac{0.4785 \times 2.8097}{\sqrt{256}} = 0.084$.

Additional Results on the Minimality and Sensitivity of C_o

The minimality experiments in Section 4.4 only consider a full 1 bit perturbation to the full precision configuration matrix. We further investigate the minimality of C_o and its sensitivity to precision perturbation per tenor type. The results of this investigation are presented in Figure 4.4.

First, we consider random fractional precision perturbations, meaning perturbations to the precision configuration matrix where only a random fraction p of the $5L$ precision assignments is incremented or decremented. A fractional precision perturbation of 1 (-1) corresponds to C_{+1} (C_{-1}). A fractional precision perturbation of 0.5 (-0.5) means that a randomly chosen half of the precision assignments is incremented (decremented). Figure 4.4 (a)

shows the relative test error deviation compared to the test error associated with C_o for various fractional precision perturbations. The error deviation is taken in a relative fashion to account for the variability of the different networks' accuracies. For instance, an absolute 1% difference in accuracy on a network trained on SVHN is significantly more severe than one on a network trained on CIFAR-100. It is observed that for negative precision perturbations the variation in test error is more important than for the case of positive perturbations. This is further encouraging evidence that C_o is nearly minimal, in that a negative perturbation causes significant accuracy degradation while a positive one offers diminishing returns.

It is also interesting to study which of the $5L$ tensor types is most sensitive to precision reduction. To do so, we perform a similar experiment whereby we selectively decrement the precision of all tensors belonging to the same type (weights, activations, weight gradients, activation gradients, weight accumulators). The results of this experiment are found in Figure 4.4 (b). It is found that the most sensitive tensor types are weights and activation gradients while the least sensitive ones are activations and weight gradients. This is an interesting finding raising further evidence that there exists some form of duality between the forward propagation of activations and back propagation of derivatives as far as numerical precision is concerned.

Illustration of Methodology Usage

We illustrate a step by step application of our precision assignment methodology to the four networks on which we reported results.

CIFAR-10 ConvNet

Feedforward Precisions: The first step in our methodology consists of setting the feedforward precisions B_{W_l} and B_{A_l}. As per Claim 1, this requires using (4.1). To do so, it is first necessary to compute the quantization noise gains using (4.6). Using the converged weights from the baseline run, we obtain the noise gains listed in Table 4.2. Consequently, $E_{(\min)} = 94.7$ and the feedforward precision offsets should be set according to (4.1) as listed in Table 4.2. The value of $B^{(\min)}$ is swept and p_m is evaluated on the validation set. It is found that the smallest value of $B^{(\min)}$ resulting in $p_m < 1\%$ is equal

Table 4.2: Feedforward path precisions in the CIFAR-10 ConvNet

Noise gain values

Layer Index l	1	2	3	4	5
$E_{W_l \to p_m}$	1.52E+06	1.24E+06	4.21E+06	3.57E+06	2.35E+06
$E_{A_l \to p_m}$	5.51E+04	3.27E+02	5.15E+02	6.60E+02	7.78E+02
Layer Index l	6	7	8	9	
$E_{W_l \to p_m}$	5.61E+05	5.97E+04	3.23E+04	8.66E+03	
$E_{A_l \to p_m}$	7.49E+02	6.32E+02	2.37E+02	9.47E+01	

Precision offsets

Layer Index l	1	2	3	4	5
B_{W_l}	$7+B^{(\min)}$	$7+B^{(\min)}$	$8+B^{(\min)}$	$8+B^{(\min)}$	$7+B^{(\min)}$
B_{A_l}	$4+B^{(\min)}$	$1+B^{(\min)}$	$1+B^{(\min)}$	$1+B^{(\min)}$	$2+B^{(\min)}$
Layer Index l	6	7	8	9	
B_{W_l}	$6+B^{(\min)}$	$5+B^{(\min)}$	$4+B^{(\min)}$	$3+B^{(\min)}$	
B_{A_l}	$1+B^{(\min)}$	$1+B^{(\min)}$	$1+B^{(\min)}$	$0+B^{(\min)}$	

Feedforward precisions

Layer Index l	1	2	3	4	5	6	7	8	9
B_{W_l}	11	11	12	12	11	10	9	8	7
B_{A_l}	8	5	5	5	6	5	5	5	4

to 4 bit. Hence the feedforward precisions are set as listed in Table 4.2 and as illustrated in Figure 4.2:

Gradient Precisions: The second step of the methodology is to determine the precisions of weight $B_{G_l^{(W)}}$ and activation $B_{G_{l+1}^{(A)}}$ gradients. As per Claim 1, an important statistic is the spatial variance of the gradient tensors. We estimate these variances via moving window averages, where at each iteration, the running variance estimate $\hat{\sigma}^2$ is updated using the instantaneous variance $\tilde{\sigma}^2$ as follows:

$$\hat{\sigma}^2 \leftarrow (1-\theta)\hat{\sigma}^2 + \theta\tilde{\sigma}^2,$$

where θ is the running average factor, chosen to be 0.1. The running variance estimate of each gradient tensor is dumped every epoch. Using the maximum recorded estimate and (4.2) we compute the PDRs of the gradient tensors (as a reminder, the PDR is forced to be a power of 2). These are listed in Table 4.3.

Furthermore, using the minimum recorded estimates of the weight gradient spatial variances and (4.3), we compute the values of the quantization step

Table 4.3: Backward path precisions in the CIFAR-10 ConvNet

Gradient PDR values

Layer Index l	1	2	3	4	5
$r_{G_l^{(W)}}$	5.00E-01	1.25E-01	1.25E-01	1.25E-01	6.25E-02
$r_{G_{l+1}^{(A)}}$	4.88E-04	9.77E-04	9.77E-04	1.95E-03	7.81E-03
Layer Index l	6	7	8	9	
$r_{G_l^{(W)}}$	3.13E-02	3.13E-02	1.56E-02	1.25E-01	
$r_{G_{l+1}^{(A)}}$	1.56E-02	7.81E-03	7.81E-03	3.13E-02	

Weight gradient step-size values

Layer Index l	1	2	3	4	5
$\Delta_{G_l^{(W)}}$	3.91E-03	1.95E-03	9.77E-04	9.77E-04	4.88E-04
Layer Index l	6	7	8	9	
$\Delta_{G_l^{(W)}}$	2.44E-04	2.44E-04	1.22E-04	4.88E-04	

Square Jacobian matrix largest singular values

Layer Index l	1	2	3	4	5
$\lambda^{(\max)}_{G_{l+1}^{(A)} \to G_l^{(W)}}$	1.44E+02	2.37E+02	4.28E+02	2.03E+02	4.20E+01
Layer Index l	6	7	8	9	
$\lambda^{(\max)}_{G_{l+1}^{(A)} \to G_l^{(W)}}$	9.08E+00	1.37E+01	1.26E+01	3.51E+00	

Activation gradient step-size values

Layer Index l	1	2	3	4	5
$\Delta_{G_{l+1}^{(A)}}$	6.10E-05	3.05E-05	7.63E-06	1.53E-05	1.53E-05
Layer Index l	6	7	8	9	
$\Delta_{G_{l+1}^{(A)}}$	1.53E-05	1.53E-05	7.63E-06	6.10E-05	

Backward path tensor precision values

Layer Index l	1	2	3	4	5	6	7	8	9
$B_{G_l^{(W)}}$	9	9	9	9	9	9	9	9	10
$B_{G_{l+1}^{(A)}}$	5	8	9	9	11	12	11	11	11
$B_{W_l^{(Acc)}}$	13	15	14	14	16	18	19	21	20

sizes of the weight tensors and include them in Table 4.3. Hence the weight gradient precisions $B_{G_l^{(W)}}$ are set as listed in Table 4.3 and as illustrated in Figure 4.2.

In order to compute the activation gradient precisions, (4.3) dictates that we need the values of largest singular values of the of the square-Jacobians of $G_l^{(W)}$ with respect to $G_{l+1}^{(A)}$ for $l = 1 \ldots L$. The square Jacobian matrices are estimated in a moving window fashion as for the variances above. However,

Table 4.4: Feedforward path precisions in the SVHN ConvNet

Noise gain values and precision offsets

Layer Index l	1	2	3	4	5
$E_{W_l \to p_m}$	3.07E+03	4.50E+02	1.54E+03	1.79E+03	6.01E+03
B_{W_l}	$6+B^{(\min)}$	$5+B^{(\min)}$	$6+B^{(\min)}$	$6+B^{(\min)}$	$7+B^{(\min)}$
$E_{A_l \to p_m}$	7.58E+02	2.86E+00	7.09E+00	2.55E+00	8.33E+00
B_{A_l}	$5+B^{(\min)}$	$1+B^{(\min)}$	$2+B^{(\min)}$	$1+B^{(\min)}$	$2+B^{(\min)}$
Layer Index l	6	7	8	9	
$E_{W_l \to p_m}$	1.25E+03	7.91E+01	1.20E+01	9.13E+00	
B_{W_l}	$6+B^{(\min)}$	$4+B^{(\min)}$	$2+B^{(\min)}$	$2+B^{(\min)}$	
$E_{A_l \to p_m}$	8.18E+00	1.78E+01	1.14E+00	3.90E-01	
B_{A_l}	$2+B^{(\min)}$	$3+B^{(\min)}$	$1+B^{(\min)}$	$0+B^{(\min)}$	

Feedforward precisions

Layer Index l	1	2	3	4	5	6	7	8	9
B_{W_l}	9	8	9	9	10	9	7	5	5
B_{A_l}	8	4	5	4	6	6	7	4	3

instead of updating a matrix every iteration, the updates are done every first batch of every epoch. The maximum recorded singular values are listed in Table 4.3.

Using the above values and (4.3) we obtain the values of the quantization step sizes for the activation gradients as listed in Table 4.3. Hence the activation gradient precisions $B_{G_{l+1}^{(A)}}$ are set as listed in Table 4.3 and as illustrated in Figure 4.2.

Internal Weight Accumulator Precisions: By application of (4.4), we use the above results to obtain the internal weight accumulator precisions. The only additional information needed is the value of the smallest learning rate value used in the training, which in our case is 0.0001. We obtain the precisions listed in Table 4.3 which are illustrated in Figure 4.2.

SVHN ConvNet

Feedforward Precisions: The quantization noise gains are used to obtain values for the precisions as a function of $B^{(\min)}$ as summarized in Table 4.4. The value of $B^{(\min)}$ is again swept, and it is found that the $p_m < 1\%$ for $B^{(\min)} = 3$. The feedforward precisions are therefore set as listed in Table 4.4 and as illustrated in Figure 4.2.

Gradient Precisions: The spatial variance of the gradient tensors is used

Table 4.5: Backward path precisions in the SVHN ConvNet

Gradient PDR and step-size, and Square Jacobian singular values

Layer Index l	1	2	3	4	5
$r_{G_l^{(W)}}$	6.25E-02	1.56E-02	1.56E-02	1.56E-02	1.56E-02
$\Delta_{G_l^{(W)}}$	2.44E-04	6.10E-05	6.10E-05	6.10E-05	6.10E-05
$r_{G_{l+1}^{(A)}}$	4.88E-04	4.88E-04	9.77E-04	1.95E-03	3.91E-03
$\lambda^{(\max)}_{G_{l+1}^{(A)} \to G_l^{(W)}}$	5.13E+00	1.48E+02	3.25E+02	1.37E+02	8.84E+01
$\Delta_{G_{l+1}^{(A)}}$	3.05E-05	9.54E-07	9.54E-07	9.54E-07	1.91E-06
Layer Index l	6	7	8	9	
$r_{G_l^{(W)}}$	7.81E-03	3.91E-03	3.91E-03	1.56E-02	
$\Delta_{G_l^{(W)}}$	3.05E-05	1.53E-05	7.63E-06	6.10E-05	
$r_{G_{l+1}^{(A)}}$	1.56E-02	3.91E-03	3.91E-03	3.13E-02	
$\lambda^{(\max)}_{G_{l+1}^{(A)} \to G_l^{(W)}}$	2.20E+01	9.58E+00	1.78E+00	1.71E+00	
$\Delta_{G_{l+1}^{(A)}}$	1.91E-06	9.54E-07	9.54E-07	7.63E-06	

Backward path tensor precision values

Layer Index l	1	2	3	4	5	6	7	8	9
$B_{G_l^{(W)}}$	9	9	9	9	9	9	9	10	9
$B_{G_{l+1}^{(A)}}$	5	10	11	12	12	14	13	13	13
$B_{W_l^{(Acc)}}$	14	17	16	16	15	17	20	23	20

to determine the PDRs and the quantization step sizes of weight gradients. The singular values of the square-Jacobians are needed to determine the quantization step sizes of activation gradients. They were computed as listed in Table 4.5. Hence the gradient precisions are set as listed in Table 4.5 and as illustrated in Figure 4.2.

Internal Weight Accumulator Precisions: The smallest learning rate value for this network is 0.001 which results in the precisions for the internal weight accumulators in Table 4.5 as illustrated in Figure 4.2.

CIFAR-10 ResNet

Feedforward Precisions: The quantization noise gains are used to obtain values for the precisions as a function of $B^{(\min)}$ as summarized in Table 4.6. Note that for weights, layer depths 21 and 22 correspond to the strided convolutions in the shortcut connections of residual blocks 4 and 7, respectively.

Table 4.6: Feedforward path precisions in the CIFAR-10 ResNet

Noise gain values and precision offsets

Layer Index l	1	2	3	4	5	6
$E_{W_l \to p_m}$	2.41E+03	9.80E+02	1.22E+03	1.62E+03	1.52E+03	3.05E+03
B_{W_l}	$11+B^{(\min)}$	$10+B^{(\min)}$	$10+B^{(\min)}$	$10+B^{(\min)}$	$10+B^{(\min)}$	$11+B^{(\min)}$
$E_{A_l \to p_m}$	7.32E-01	5.15E-01	1.29E-01	1.12E-01	7.31E-02	8.98E-02
B_{A_l}	$5+B^{(\min)}$	$5+B^{(\min)}$	$4+B^{(\min)}$	$4+B^{(\min)}$	$3+B^{(\min)}$	$3+B^{(\min)}$
Layer Index l	7	8	9	10	11	12
$E_{W_l \to p_m}$	1.47E+03	2.15E+03	2.74E+03	4.96E+03	4.23E+03	4.20E+03
B_{W_l}	$10+B^{(\min)}$	$11+B^{(\min)}$	$11+B^{(\min)}$	$11+B^{(\min)}$	$11+B^{(\min)}$	$12+B^{(\min)}$
$E_{A_l \to p_m}$	7.70E-02	8.39E-02	6.38E-02	1.92E-01	1.54E-01	1.33E-01
B_{A_l}	$3+B^{(\min)}$	$3+B^{(\min)}$	$3+B^{(\min)}$	$4+B^{(\min)}$	$4+B^{(\min)}$	$4+B^{(\min)}$
Layer Index l	13	14	15	16	17	18
$E_{W_l \to p_m}$	7.25E+03	2.99E+03	2.86E+03	3.00E+03	5.02E+03	4.34E+03
B_{W_l}	$11+B^{(\min)}$	$11+B^{(\min)}$	$11+B^{(\min)}$	$11+B^{(\min)}$	$10+B^{(\min)}$	$10+B^{(\min)}$
$E_{A_l \to p_m}$	1.13E-01	8.51E-02	6.57E-02	1.29E-01	6.51E-02	2.16E-02
B_{A_l}	$4+B^{(\min)}$	$3+B^{(\min)}$	$3+B^{(\min)}$	$4+B^{(\min)}$	$3+B^{(\min)}$	$2+B^{(\min)}$
Layer Index l	19	20	21	22		
$E_{W_l \to p_m}$	1.41E+03	1.30E+03	1.08E+02	8.31E+00		
B_{W_l}	$9+B^{(\min)}$	$7+B^{(\min)}$	$11+B^{(\min)}$	$11+B^{(\min)}$		
$E_{A_l \to p_m}$	4.80E-03	7.82E-04				
B_{A_l}	$1+B^{(\min)}$	$0+B^{(\min)}$				

Feedforward precisions

Layer Index l	1	2	3	4	5	6	7	8	9	10	11
B_{W_l}	14	13	13	13	13	14	13	14	14	14	14
B_{A_l}	8	8	7	7	6	6	6	6	6	7	7
Layer Index l	12	13	14	15	16	17	18	19	20	21	22
B_{W_l}	15	14	14	14	14	13	13	12	10	14	14
B_{A_l}	7	7	6	6	7	6	5	4	3		

The value of $B^{(\min)}$ is again swept, and it is found that the $p_m < 1\%$ for $B^{(\min)} = 3$. The feedforward precisions are therefore set as listed in Table 4.6 and as illustrated in Figure 4.2.

Gradient Precisions: The spatial variance of the gradient tensors is used to determine the PDRs and the quantization step sizes of weight gradients. The singular values of the square-Jacobians are needed to determine the quantization step sizes of activation gradients. They were computed as listed in Table 4.7. Hence the gradient precisions are set as listed in Table 4.7 and as illustrated in Figure 4.2.

Internal Weight Accumulator Precisions: The smallest learning rate value for this network is 0.001 which results in the precisions for the internal weight accumulators in Table 4.7 as illustrated in Figure 4.2.

Table 4.7: Backward path precisions in the CIFAR-10 ResNet

Gradient PDR and step-size, and Square Jacobian singular values

Layer Index l	1	2	3	4	5	6
$r_{G_l^{(W)}}$	2.50E-01	6.25E-02	6.25E-02	3.13E-02	3.13E-02	3.13E-02
$\Delta_{G_l^{(W)}}$	2.44E-04	3.05E-05	3.05E-05	3.05E-05	3.05E-05	3.05E-05
$r_{G_{l+1}^{(A)}}$	4.88E-04	2.44E-04	2.44E-04	2.44E-04	2.44E-04	2.44E-04
$\lambda^{(\max)}_{G_{l+1}^{(A)} \to G_l^{(W)}}$	8.07E+02	2.84E+03	2.84E+03	5.43E+03	5.43E+03	4.94E+03
$\Delta_{G_{l+1}^{(A)}}$	7.63E-06	4.77E-07	4.77E-07	2.38E-07	2.38E-07	2.38E-07
Layer Index l	7	8	9	10	11	12
$r_{G_l^{(W)}}$	3.13E-02	3.13E-02	3.13E-02	3.13E-02	3.13E-02	3.13E-02
$\Delta_{G_l^{(W)}}$	3.05E-05	3.05E-05	3.05E-05	3.05E-05	3.05E-05	3.05E-05
$r_{G_{l+1}^{(A)}}$	2.44E-04	2.44E-04	4.88E-04	4.88E-04	2.44E-04	2.44E-04
$\lambda^{(\max)}_{G_{l+1}^{(A)} \to G_l^{(W)}}$	4.94E+03	1.22E+03	1.22E+03	1.08E+03	1.08E+03	8.07E+02
$\Delta_{G_{l+1}^{(A)}}$	2.38E-07	4.77E-07	4.77E-07	4.77E-07	4.77E-07	9.54E-07
Layer Index l	13	14	15	16	17	18
$r_{G_l^{(W)}}$	3.13E-02	3.13E-02	1.56E-02	1.56E-02	1.56E-02	1.56E-02
$\Delta_{G_l^{(W)}}$	3.05E-05	3.05E-05	1.53E-05	1.53E-05	1.53E-05	1.53E-05
$r_{G_{l+1}^{(A)}}$	2.44E-04	4.88E-04	4.88E-04	4.88E-04	2.44E-04	2.44E-04
$\lambda^{(\max)}_{G_{l+1}^{(A)} \to G_l^{(W)}}$	8.07E+02	1.93E+02	1.93E+02	2.98E+02	2.98E+02	3.01E+02
$\Delta_{G_{l+1}^{(A)}}$	9.54E-07	9.54E-07	9.54E-07	4.77E-07	2.38E-07	2.38E-07
Layer Index l	19	20	21	22		
$r_{G_l^{(W)}}$	1.56E-02	1.56E-02	1.56E-02	2.50E-01		
$\Delta_{G_l^{(W)}}$	7.63E-06	7.63E-06	1.91E-06	3.05E-05		
$r_{G_{l+1}^{(A)}}$	2.44E-04	6.25E-02				
$\lambda^{(\max)}_{G_{l+1}^{(A)} \to G_l^{(W)}}$	3.01E+02	2.32E+01				
$\Delta_{G_{l+1}^{(A)}}$	5.96E-08	3.81E-06				

Backward path tensor precision values

Layer Index l	1	2	3	4	5	6	7	8	9	10	11
$B_{G_l^{(W)}}$	11	12	12	11	11	11	11	11	11	11	11
$B_{G_{l+1}^{(A)}}$	7	10	10	11	11	11	11	10	11	11	10
$B_{W_l^{(Acc)}}$	9	13	13	13	13	12	13	12	12	12	12
Layer Index l	12	13	14	15	16	17	18	19	20	21	22
$B_{G_l^{(W)}}$	11	11	11	11	11	11	11	12	12	14	14
$B_{G_{l+1}^{(A)}}$	9	9	10	10	11	11	11	13	15		
$B_{W_l^{(Acc)}}$	11	12	13	13	13	15	15	18	16	12	13

CIFAR-100 ResNet

Feedforward Precisions: The quantization noise gains are used to obtain values for the precisions as a function of $B^{(\min)}$ as summarized in Table 4.8.

Table 4.8: Feedforward path precisions in the CIFAR-100 ResNet

Noise gain values and precision offsets

Layer Index l	1	2	3	4	5	6
$E_{W_l \to p_m}$	2.32E+03	8.23E+02	1.18E+03	1.28E+03	1.70E+03	2.78E+03
B_{W_l}	$10+B^{(\min)}$	$9+B^{(\min)}$	$10+B^{(\min)}$	$10+B^{(\min)}$	$10+B^{(\min)}$	$10+B^{(\min)}$
$E_{A_l \to p_m}$	1.42E+00	7.84E-01	2.52E-01	1.46E-01	7.68E-02	7.40E-02
B_{A_l}	$5+B^{(\min)}$	$4+B^{(\min)}$	$4+B^{(\min)}$	$3+B^{(\min)}$	$3+B^{(\min)}$	$3+B^{(\min)}$
Layer Index l	7	8	9	10	11	12
$E_{W_l \to p_m}$	3.03E+03	5.80E+03	7.29E+03	9.20E+03	9.81E+03	1.41E+04
B_{W_l}	$10+B^{(\min)}$	$11+B^{(\min)}$	$11+B^{(\min)}$	$11+B^{(\min)}$	$11+B^{(\min)}$	$11+B^{(\min)}$
$E_{A_l \to p_m}$	7.52E-02	8.70E-02	1.38E-01	2.49E-01	2.11E-01	1.51E-01
B_{A_l}	$3+B^{(\min)}$	$3+B^{(\min)}$	$3+B^{(\min)}$	$4+B^{(\min)}$	$3+B^{(\min)}$	$3+B^{(\min)}$
Layer Index l	13	14	15	16	17	18
$E_{W_l \to p_m}$	7.67E+03	1.40E+04	1.13E+04	1.09E+04	5.35E+03	3.97E+03
B_{W_l}	$11+B^{(\min)}$	$11+B^{(\min)}$	$11+B^{(\min)}$	$11+B^{(\min)}$	$11+B^{(\min)}$	$11+B^{(\min)}$
$E_{A_l \to p_m}$	1.54E-01	1.09E-01	1.93E-01	2.36E-01	1.27E-01	3.01E-02
B_{A_l}	$3+B^{(\min)}$	$3+B^{(\min)}$	$3+B^{(\min)}$	$4+B^{(\min)}$	$3+B^{(\min)}$	$2+B^{(\min)}$
Layer Index l	19	20	21	22		
$E_{W_l \to p_m}$	8.35E+02	2.30E+01	6.78E+03	6.03E+03		
B_{W_l}	$9+B^{(\min)}$	$7+B^{(\min)}$	$11+B^{(\min)}$	$11+B^{(\min)}$		
$E_{A_l \to p_m}$	2.01E-02	1.80E-03				
B_{A_l}	$2+B^{(\min)}$	$0+B^{(\min)}$				

Feedforward precisions

Layer Index l	1	2	3	4	5	6	7	8	9	10	11
B_{W_l}	13	12	13	13	13	13	13	14	14	14	14
B_{A_l}	8	7	7	6	6	6	6	6	6	7	6
Layer Index l	12	13	14	15	16	17	18	19	20	21	22
B_{W_l}	14	14	14	14	14	14	14	12	10	14	14
B_{A_l}	6	6	6	6	7	6	5	5	3		

The value of $B^{(\min)}$ is again swept, and it is found that the $p_m < 1\%$ for $B^{(\min)} = 3$. The feedforward precisions are therefore set as listed in Table 4.8 and as illustrated in Figure 4.2.

Gradient Precisions: The spatial variance of the gradient tensors is used to determine the PDRs and the quantization step sizes of weight gradients. The singular values of the square-Jacobians are needed to determine the quantization step sizes of activation gradients. They were computed as listed in Table 4.9. Hence the gradient precisions are set as listed in Table 4.9 and as illustrated in Figure 4.2.

Internal Weight Accumulator Precisions: The smallest learning rate value for this network is 0.001 which results in the precisions for the internal weight accumulators in Table 4.9 as illustrated in Figure 4.2.

Table 4.9: Backward path precisions in the CIFAR-100 ResNet

Gradient PDR and step-size, and Square Jacobian singular values

Layer Index l	1	2	3	4	5	6
$r_{G_l^{(W)}}$	5.00E-01	6.25E-02	6.25E-02	3.13E-02	6.25E-02	3.13E-02
$\Delta_{G_l^{(W)}}$	6.10E-05	1.53E-05	1.53E-05	1.53E-05	1.53E-05	1.53E-05
$r_{G_{l+1}^{(A)}}$	2.44E-04	1.22E-04	6.10E-05	6.10E-05	6.10E-05	6.10E-05
$\lambda^{(\max)}_{G_{l+1}^{(A)} \to G_l^{(W)}}$	6.46E+02	1.86E+03	1.86E+03	3.54E+03	3.54E+03	5.11E+03
$\Delta_{G_{l+1}^{(A)}}$	9.54E-07	1.19E-07	1.19E-07	1.19E-07	1.19E-07	5.96E-08
Layer Index l	7	8	9	10	11	12
$r_{G_l^{(W)}}$	3.13E-02	3.13E-02	3.13E-02	3.13E-02	3.13E-02	3.13E-02
$\Delta_{G_l^{(W)}}$	1.53E-05	1.53E-05	1.53E-05	1.53E-05	1.53E-05	1.53E-05
$r_{G_{l+1}^{(A)}}$	6.10E-05	1.22E-04	1.22E-04	1.22E-04	1.22E-04	1.22E-04
$\lambda^{(\max)}_{G_{l+1}^{(A)} \to G_l^{(W)}}$	5.11E+03	1.05E+03	1.05E+03	8.23E+02	8.23E+02	6.37E+02
$\Delta_{G_{l+1}^{(A)}}$	5.96E-08	1.19E-07	1.19E-07	2.38E-07	2.38E-07	2.38E-07
Layer Index l	13	14	15	16	17	18
$r_{G_l^{(W)}}$	3.13E-02	3.13E-02	1.56E-02	3.13E-02	1.56E-02	1.56E-02
$\Delta_{G_l^{(W)}}$	1.53E-05	7.63E-06	7.63E-06	7.63E-06	7.63E-06	7.63E-06
$r_{G_{l+1}^{(A)}}$	1.22E-04	1.22E-04	2.44E-04	1.22E-04	6.10E-05	6.10E-05
$\lambda^{(\max)}_{G_{l+1}^{(A)} \to G_l^{(W)}}$	6.37E+02	2.31E+02	2.31E+02	2.79E+02	2.79E+02	2.80E+02
$\Delta_{G_{l+1}^{(A)}}$	2.38E-07	2.38E-07	2.38E-07	1.19E-07	1.19E-07	1.19E-07
Layer Index l	19	20	21	22		
$r_{G_l^{(W)}}$	1.56E-02	6.25E-02	3.13E-02	1.56E-02		
$\Delta_{G_l^{(W)}}$	3.81E-06	7.63E-06	1.53E-05	7.63E-06		
$r_{G_{l+1}^{(A)}}$	6.10E-05	7.81E-03				
$\lambda^{(\max)}_{G_{l+1}^{(A)} \to G_l^{(W)}}$	2.80E+02	7.81E+01				
$\Delta_{G_{l+1}^{(A)}}$	5.96E-08	2.38E-07				

Backward path tensor precision values

Layer Index l	1	2	3	4	5	6	7	8	9	10	11
$B_{G_l^{(W)}}$	14	13	13	12	13	12	12	12	12	12	12
$B_{G_{l+1}^{(A)}}$	9	11	10	10	10	11	11	11	11	10	10
$B_{W_l^{(Acc)}}$	12	15	14	14	14	14	14	13	13	13	13
Layer Index l	12	13	14	15	16	17	18	19	20	21	22
$B_{G_l^{(W)}}$	12	12	13	12	13	12	12	13	14	12	12
$B_{G_{l+1}^{(A)}}$	10	10	10	11	11	10	10	11	16		
$B_{W_l^{(Acc)}}$	13	13	14	14	14	14	14	17	18	13	14

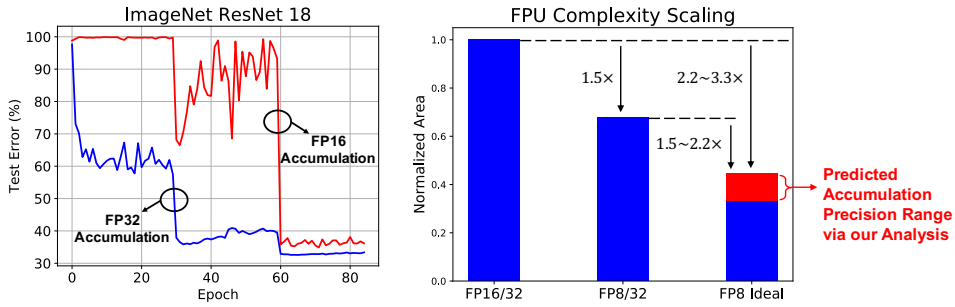
CHAPTER 5

FLOATING-POINT TRAINING WITH ACCUMULATION BIT-WIDTH SCALING

Chapter 4 studied the precision requirements for fixed-point training. In this chapter, we study reduced precision floating-point training. Prior work has investigated the effects of representation quantization, but none has studied the impact of partial sum accumulation quantization. Thus, the hardware benefits from reduced precision training currently face bottlenecks by virtue of the large accumulators used. In this chapter, we analytically obtain mantissa precision requirements of all accumulators in the back-propagation algorithm. To do so, we present the variance retention ratio (VRR), a metric quantifying the suitability, or lack thereof, of a floating-point precision configuration. A formula for the VRR is derived and used to determine accumulation bit-width for precise tailoring of computation hardware. Experimentally, the validity and tightness of our analysis are verified across multiple deep learning benchmarks.

5.1 Motivation

There are several reasons why reduced precision deep learning has attracted the attention of both hardware and algorithms researchers. First, it offers well-defined and scalable hardware efficiency, as opposed to other complexity reduction techniques such as pruning [66, 71], where handling sparse data is needed. Indeed, parameter complexity scales linearly while multiplication hardware complexity scales quadratically with precision bit-width [53]. Thus, any advance towards truly binarized networks [51] corresponds to potentially 30x - 1000x complexity reduction in comparison to single precision floating-point hardware. Second, the mathematics of reduced precision has direct ties with the statistical theory of quantization [104]. In the context of deep learning, this presents an opportunity for theoreticians to derive analytical



(a) (b)

Figure 5.1: The importance of accumulation precision: (a) Convergence curves of an ImageNet ResNet 18 experiment using reduced precision accumulation. The current practice is to keep the accumulation in full precision to avoid such divergence. (b) Estimated area benefits when reducing the precision of a floating-point unit (FPU). The terminology FP$a/b$ denotes an FPU whose multiplier and adder use a and b bit, respectively. Our work enables convergence in reduced precision accumulation and gains an extra $1.5\times \sim 2.2\times$ area reduction.

trade-offs between model accuracy and numerical precision [45, 89].

Most ongoing efforts on reduced precision deep learning solely focus on quantizing representations and always assume wide accumulators, i.e., ideal summations. The reason for this is that reduced precision accumulation can result in severe training instability and accuracy degradation, as illustrated in Figure 5.1 (a) for ResNet 18 (ImageNet) model training. This is especially unfortunate, since the hardware complexity in reduced precision floating-point numbers (needed to represent small gradients during training) [74, 131] is dominated by the accumulator bit-width. To illustrate this dominance we developed a model underpinned by the hardware synthesis of low-precision floating-point units (FPU), that translates precision into area complexity of the FPU. Comparisons obtained from this model are shown in Figure 5.1 (b). We observe that accumulating in high precision severely limits the hardware benefits of reduced precision computations. This presents a new angle to approach the problem of reduced precision deep learning training; this approach concerns determining suitable accumulation precision and forms the basis of our work. Our findings are that the accumulation precision requirements in deep learning training are nowhere near 32 bit, and in fact could enable further complexity reduction of FPUs by a factor of $1.5 \sim 2.2\times$.

Our work is concerned with establishing theoretical foundations for estimating the accumulation bit precision requirements in deep learning. While this topic has never been addressed in the past, there is prior work in both deep learning and high-performance computing communities that aligns well with ours.

Most early work on reduced precision deep learning considers fixed-point arithmetic or a variation of it [72]. However, when considering quantization of signals involving the back-propagation algorithm, finding a suitable fixed-point configuration becomes challenging due to a weak handle on the scalar dynamic range of the back-propagated signals. Thus, hardware solutions have been sought, and, accordingly, other number formats were considered. Flexpoint [25] is a hybrid version between fixed-point and floating-point where scalars in a tensor are quantized to 16 bit fixed-point but share 5 bit of exponent to adjust the dynamic range. Similarly, WAGE [73] augments Flexpoint with stochastic quantization and enables integer quantization. All of these schemes focused on representation precision, but mostly used 32 bit accumulation. Another option is to use reduced precision floating-point as was done in MPT [131], which reduces the precision of most signals to 16 bit floating-point, but incurs accuracy degradation when reducing the accumulation precision from 32 bit. Recently, [74] quantized all representations to 8 bit floating-point and experimentally found that the accumulation could be in 16 bit with algorithmic contrivance, such as chunk-based accumulation, to enable convergence.

The issue of numerical errors in floating-point accumulation has been classically studied in the area of high-performance computing. Robertazzi and Schwartz [132] were among the first to statistically estimate the effects of floating-point accumulation. Assuming a stream of uniformly and exponentially distributed positive numbers, estimates for the mean square error of the floating-point accumulation were derived via quantization noise analysis. Because such analyses are often intractable (due to the multiplicative nature of the noise), later works on numerical stability focus on worst-case estimates of the accumulation error. Higham [133] provides upper bounds on the error magnitude by counting and analyzing round-off errors. Following this style of worst-case analysis, Castaldo et al. [134] provide bounds on the accumulation error for different summing algorithms, notably using chunk-based summations. Different approaches to chunking are considered and their ben-

efits are estimated. It is to be noted that these analyses are often loose as they are agnostic to the application space. To the best of our knowledge, a statistical analysis of the accumulation precision specifically tailored to deep learning training remains elusive.

5.1.1 Contributions

Our contribution is both theoretical and practical. We introduce the variance retention ratio (VRR) of a reduced precision accumulation in the context of the three deep learning dot products. The VRR is used to assess the suitability, or lack thereof, of a precision configuration. Our main result is the derivation of an actual formula for the VRR that allows us to determine accumulation bit-width for precise tailoring of computation hardware. Experimentally, we verify the validity and tightness of our analysis across three benchmarking networks (CIFAR-10 ResNet 32, ImageNet ResNet 18 and ImageNet AlexNet).

5.2 Background on Floating-Point Arithmetic

The following basic floating-point definitions and notations are used in our work:

Floating-point representation: A b bit floating-point number a has a signed bit, e exponent bits, and m mantissa bits so that $b = 1+e+m$. Its binary representation is $(B_s, B'_1, \ldots, B'_e, B''_1, \ldots, B''_m) \in \{0,1\}^b$ and its value is equal to: $a = (-1)^{B_s} \times 2^E \times (1+M)$ where $E = -(2^{e-1}-1) + \sum_{i=1}^{e} B'_i 2^{(e-i)}$ and $M = \sum_{i=1}^{m} B''_i 2^{-i}$. Such a number is called a $(1, e, m)$ floating-point number.

Floating-point operations: One of the most pervasive arithmetic functions used in deep learning is the dot product between two vectors which is the building block of the generalized matrix multiplication (GEMM). A dot product is computed in a multiply-accumulate (MAC) fashion and thus requires two floating-point operations: multiplication and addition. The realization of an ideal floating-point operation requires a certain bit growth at the output to avoid loss of information. For instance, in a typical MAC operation, if $c \leftarrow c+a\times b$ where a is $(1, e_a, m_a)$ and b is $(1, e_b, m_b)$, then c should be

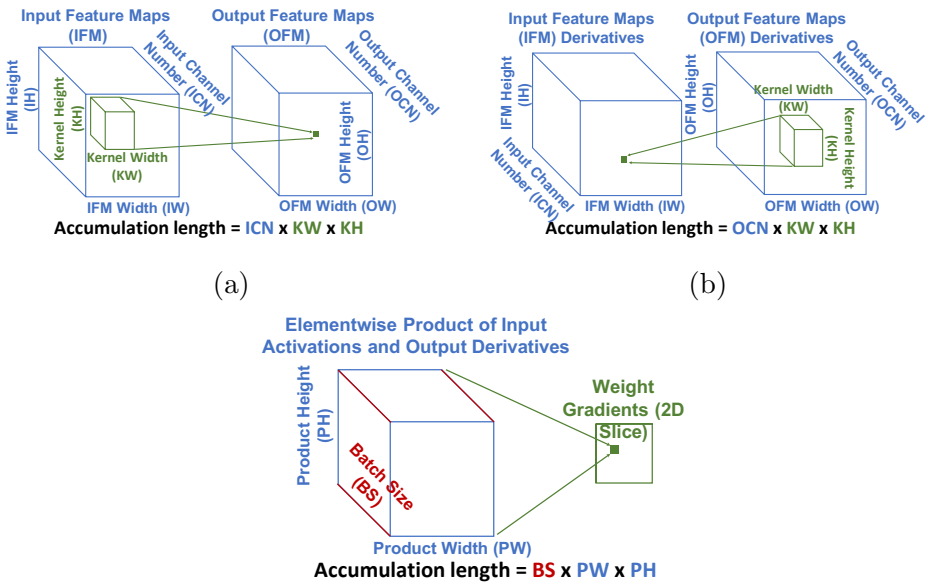


Figure 5.2: The three GEMM calls, and hence accumulations, in one iteration of the back-propagation algorithm: (a) the forward propagation (FWD), (b) the backward propagation (BWD), and (c) gradient computation (GRAD). The accumulation of these three GEMMs is across multiple dimensions (mini-batch size, feature maps, output channels etc.) and their lengths are usually very long.

$(1, \max(e_a, e_b)+2, m_a+m_b+1+\Delta_E)$, which depends on the bit-precision and the relative exponent difference of the operands Δ_E. However, it is often more practical to pre-define the precision of c as $(1, e_c, m_c)$, which requires rounding immediately after computation. Such rounding might cause an operand to be completely or partially truncated out of the addition, a phenomenon called "swamping" [133], which is the primary source of accumulation errors.

5.3 Accumulation Variance

The second-order statistics (variance) of signals are known to be of great importance in deep learning. For instance, in prior works on weight initialization [41, 135], it is customary to initialize random weights subject to a variance constraint designed to prevent vanishing or explosion of activations and gradients. Thus, such variance engineering induces fine convergence of

DNNs. Importantly, in such analyses, the second-order output statistics of a dot product are studied and expressed as a function of that of the accumulated terms, which are assumed to be independent and having similar variance. A fundamental assumption is: $Var(s) = nVar(p)$, where $Var(s)$ and $Var(p)$ are the variances of the sum and individual product terms, respectively, and n is the length of the dot product. One intuition concerning accumulation with reduced precision is that, due to swamping, some product terms may vanish from the summation, resulting in a lower variance than expected: $Var(s) = \tilde{n}Var(p)$, where $\tilde{n} < n$. This constitutes a violation of a key assumption and effectively leads to the re-emergence of the difficulties in training neural networks with improper weight initialization which often harms the convergence behavior [41, 135].

To explain the poor convergence of our ResNet 18 run (Figure 5.1 (a)), we evaluate the behavior of accumulation variance across layers. Specifically, we check the three dot products of a back-propagation iteration: the forward propagation (FWD), the backward propagation (BWD), and the gradient computation (GRAD), as illustrated in Figure 5.2. Indeed, there is an abnormality in reduced precision GRAD as shown in Figure 5.3. It is also observed that the abnormality of variance is directly related to accumulation length. From Figure 5.3, the break point corresponds to the switch from the first to the second residual block. The GRAD accumulation length in the former is much longer ($4\times$) than the latter. Thus, evidence points to the direction that for a given precision, there is an accumulation length for which the expected variance cannot be properly retained due to swamping. Motivated by these observations, we propose to study the trade-offs among accumulation variance, length, and mantissa precision.

Before proceeding, it is important to note that our upcoming analysis differs in style from many works on reduced precision deep learning where it is common to model quantization effects as additive noise causing increased variance [136]. Our work does not contradict such findings, since prior works have considered representation quantization whose effects are by nature different from intermediate roundings in partial sums.

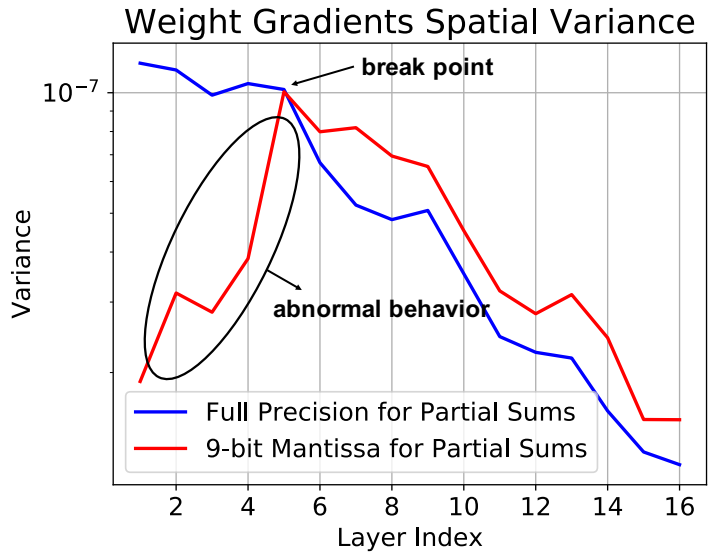


Figure 5.3: Snapshot of measured weight gradient variance as a function of layer index for our ResNet 18 experiment. An abnormal variance is observed for the reduced precision accumulation.

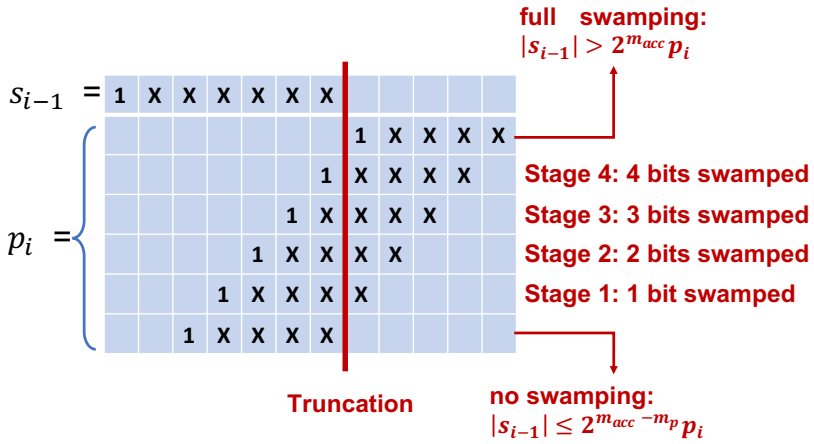


Figure 5.4: Illustration of the difference between full swamping and partial swamping when $m_{acc} = 6$ and $m_p = 4$. The bit-shift of p_i due to exponent difference may cause partial (e.g., stage 1-4) or full truncation of p_i's bits, called swamping.

5.4 Mantissa Precision Requirements Analysis

We assume sufficient exponent precision throughout and treat reduced precision floating-point arithmetic as an unbiased form of approximate computing, as is customary. Thus, our work focuses on associating second-order statistics to mantissa precision.

We consider the accumulation of n terms $\{p_i\}_{i=1}^n$ which correspond to the element-wise product terms in a dot product. The goal is to compute the correct n^{th} partial sum s_n where $s_i = \sum_{i'=1}^{i} p_{i'}$. We assume, as in [135], that the product terms $\{p_i\}_{i=1}^n$ are statistically independent, zero-mean, and have the same variance σ_p^2. Thus, under ideal computation, the variance of s_n (which is equal to its second moment) should be $Var(s_n)_{\text{ideal}} = n\sigma_p^2$. However, due to swamping effects, the variance of s_n under reduced precision is $Var(s_n)_{\text{swamping}} \neq Var(s_n)_{\text{ideal}}$.

Let the product terms $\{p_i\}_{i=1}^n$ and partial sum terms $\{s_i\}_{i=1}^n$ have m_p and m_{acc} mantissa bits, respectively. Our key contribution is a formula for the *variance retention ratio* $VRR = \frac{Var(s_n)_{\text{swamping}}}{Var(s_n)_{\text{ideal}}}$. The VRR, which is always less than or equal to unity, is a function of n, m_p, and m_{acc} only, which needs no simulations to be computed. Furthermore, to preserve quality of computation under reduced precision, it is required that $VRR \to 1$. As it turns out, the VRR for a fixed precision is a curve with "knee" with respect to n, where a break point in accumulation length, beyond which a certain mantissa precision is no longer suitable, can be easily identified. Accordingly, for a given accumulation length, the mantissa precision requirements can be readily estimated.

Before proceeding, we formally define swamping. As illustrated in Figure 5.4, the bit-shift of p_i due to exponent difference may cause partial (e.g., stage 1-4) or full truncation of p_i's bits, called swamping. We define (1) "full swamping" which occurs when $|s_i| > 2^{m_{acc}} |p_{i+1}|$, and (2) "partial swamping" which occurs when $2^{m_{acc}-m_p} |p_{i+1}| < |s_i| \leq 2^{m_{acc}} |p_{i+1}|$. These two swamping types will be fully considered in our analysis.

5.4.1 Variance Retention Ratio

In the lemma below, we first present a formula for the VRR when only full swamping is considered.

Lemma 8. *The variance retention ratio of a length n accumulation using m_{acc} mantissa bits, and when only considering full swamping, is given by:*

$$VRR_{\textit{full swamping}} = \frac{\sum_{i=2}^{n-1} iq_i + n\tilde{q}_n}{kn}, \tag{5.1}$$

where $q_i = 2Q\left(\frac{2^{m_{acc}}}{\sqrt{i}}\right)\left(1 - 2Q\left(\frac{2^{m_{acc}}}{\sqrt{i-1}}\right)\right)$, $\tilde{q}_n = 1 - 2Q\left(\frac{2^{m_{acc}}}{\sqrt{n}}\right)$, $k = \sum_{i=2}^{n-1} q_i + \tilde{q}_n$ *is a normalization constant, and* Q *denotes the elementary Q-function.*

The proof is provided in Section 5.7. A preliminary check is that a very large value of m_{acc} in (5.1) causes all $\{q_i\}_{i=1}^{n-1}$ terms to vanish and $\tilde{q}_n$ to approach unity. This makes $VRR \to 1$ for high precision as expected. On the other hand, if we assume m_{acc} to be small, but let $n \to \infty$, we get $n\tilde{q}_n \to 0$ because the Q-function term will approach 1 exponentially fast as opposed to the n term which is linear. Furthermore, the terms inside the summation having a large i will vanish by the same argument, while the n term in the denominator will make the ratio decrease and we would expect $VRR \to 0$. This means that with limited precision, there is little hope to achieve a correct result when the accumulation length is very large. Also, the rapid change in VRR from 0 to 1 indicates that VRR can be used to provide a sharp decision boundary for accumulation precision.

The above result only considers full swamping and is thus incomplete. Next we augment our analysis to take into account the effects of partial swamping. The corresponding formula for the VRR is provided in the following theorem.

Theorem 4. *The variance retention ratio of a length* n *accumulation using* m_p *and* m_{acc} *mantissa bits for the input products and partial sum terms, respectively, is given by:*

$$VRR = \frac{\sum_{i=2}^{n-1}(i-\alpha)_+ q_i \mathbf{1}_{\{i>\alpha\}} + \sum_{j_r=2}^{m_p}(n-\alpha_{j_r})_+ q'_i \mathbf{1}_{\{n>\alpha_{j_r}\}} + nk_3}{kn}, \quad (5.2)$$

where $(x)_+ = \begin{cases} x \text{ if } x > 0 \\ 0 \text{ otherwise} \end{cases}$, $\mathbf{1}_A = \begin{cases} 1 \text{ if } A \text{ is true} \\ 0 \text{ otherwise} \end{cases}$,

$\alpha = \frac{2^{m_{acc}-3m_p}}{3}\sum_{j=1}^{m_p} 2^j(2^j-1)(2^{j+1}-1)$, $q_i = 2Q\left(\frac{2^{m_{acc}}}{\sqrt{i}}\right)\left(1 - 2Q\left(\frac{2^{m_{acc}}}{\sqrt{i-1}}\right)\right)$,

$\alpha_{j_r} = \frac{2^{m_{acc}-3m_p}}{3}\sum_{j=1}^{j_r-1} 2^j(2^j-1)(2^{j+1}-1)$,

$q'_{j_r} = N_{j_r-1} 2Q\left(\frac{2^{m_{acc}-m_p+j_r-1}}{\sqrt{n}}\right)\left(1 - 2Q\left(\frac{2^{m_{acc}-m_p+j_r}}{\sqrt{n}}\right)\right)$, $k = k_1 + k_2 + k_3$,

$k_1 = \sum_{i=2}^{n-1} q_i \mathbf{1}_{\{i>\alpha\}}$, $k_2 = \sum_{j_r=2}^{m_p} q'_i \mathbf{1}_{\{n>\alpha_{j_r}\}}$, *and* $k_3 = 1 - 2Q\left(\frac{2^{m_{acc}-m_p+1}}{\sqrt{n}}\right)$.

The proof is provided in Section 5.7. Observe the dependence on m_{acc}, m_p, and n. Therefore, in what follows, we shall refer to the VRR in (5.2) as $VRR(m_{acc}, m_p, n)$. Once again, we verify the extremal behavior of our formula. A very large value of m_{acc} in (5.2) causes $k_1 \approx k_2 \approx 0$ and $k_3 \approx 1$. This

makes $VRR \to 1$ for high precision as expected. In addition, assuming small m_{acc} and letting $n \to \infty$, we get $nk_3 \to 0$ because k_3 decays exponentially fast due to the Q-function term. By the same argument, $q'_{j_r} \to 0$ for all j_r and $q_i \to 0$ for all but small values of i. Thus, the numerator will be small, while the denominator will increase linearly in n causing $VRR \to 0$. Thus, once more, we establish that with limited accumulation precision, there is little hope for a correct result.

5.4.2 VRR with Chunk Based Accumulations

Next we consider an accumulation that uses chunking. In particular, assume $n = n_1 \times n_2$ so that the accumulation is broken into n_2 chunks, each of length n_1. Thus, n_2 accumulations of length n_1 are performed and the n_2 intermediate results are added to obtain s_n. This simple technique is known to greatly improve the stability of sums [134]. The VRR can be used to theoretically explain such improvements. For simplicity, we assume two-level chunking (as described above) and same mantissa precision m_{acc} for both inter-chunk and intra-chunk accumulations. Applying the above analysis, we may obtain a formula for the VRR as provided in the corollary below, which is proved in Section 5.7.

Corollary 1. *The variance retention ratio of a length $n = n_1 \times n_2$ accumulation with chunking, where n_1 is the chunk size and n_2 is the number of chunks, using m_p and m_{acc} mantissa bits for the input products and partial sum terms, respectively, is given by:*

$$\begin{aligned} VRR_{chunking} =& VRR(m_{acc}, m_p, n_1) \\ & \times VRR\left(m_{acc}, \min\left(m_{acc}, m_p + \log_2(n_1)\right), n_2\right). \end{aligned} \tag{5.3}$$

5.4.3 VRR with Sparsity

It is common to encounter sparse operands in deep learning dot products. Since addition of zero is an identity operation, the effective accumulation length is often less than as described by the network topology. Indeed, for a given accumulation, supposedly of length n, if we can estimate the non-zero

ratio (NZR) of its incoming product terms, then the effective accumulation length is $NZR \times n$.

Thus, when an accumulation is known to have sparse inputs with known NZR, a better estimate of the VRR is

$$VRR_{\text{sparsity}} = VRR(m_{acc}, m_p, NZR \times n). \tag{5.4}$$

Similarly, when considering the VRR with chunking, we may use knowledge of sparsity to obtain the effective intra-accumulation length as $NZR \times n_1$. This change is reflected both in the VRR of the intra-chunk accumulation and the input precision of the inter-chunk accumulation:

$$\begin{aligned} VRR_{\text{chunking and sparsity}} = {} & VRR(m_{acc}, m_p, NZR \times n_1) \\ & \times VRR\left(m_{acc}, \min\left(m_{acc}, m_p + \log_2(NZR \times n_1)\right), n_2\right). \end{aligned} \tag{5.5}$$

In practice, the NZR can be estimated by making several observations from baseline data. Using an estimate of the NZR makes our analysis less conservative.

5.4.4 Usage of Analysis

For a given accumulation setup, one may compute the VRR and observe how close it is to the ideal value of 1 in order to judge the suitability of the mantissa precision assignment. It turns out that when measured as a function of accumulation length n for a fixed precision, the VRR has a breakdown region. This breakdown region can very well be observed when considering what we define as the normalized exponential variance lost:

$$v(n) = e^{n(1-VRR)}. \tag{5.6}$$

In Figure 5.5 (a,b) we plot $v(n)$ for different values of m_{acc} when considering both normal accumulation and chunk-based accumulation with a chunk-size of 64. The value of m_p is set to 5 bit, corresponding to the product of two numbers in (1,5,2) floating-point format [74]. We consider m_{acc} to be suitable for a given n only if $v(n) < 50$ because in all plots the variance lost rapidly increases when $v(n) > 50$ and n increases. On the other hand, when

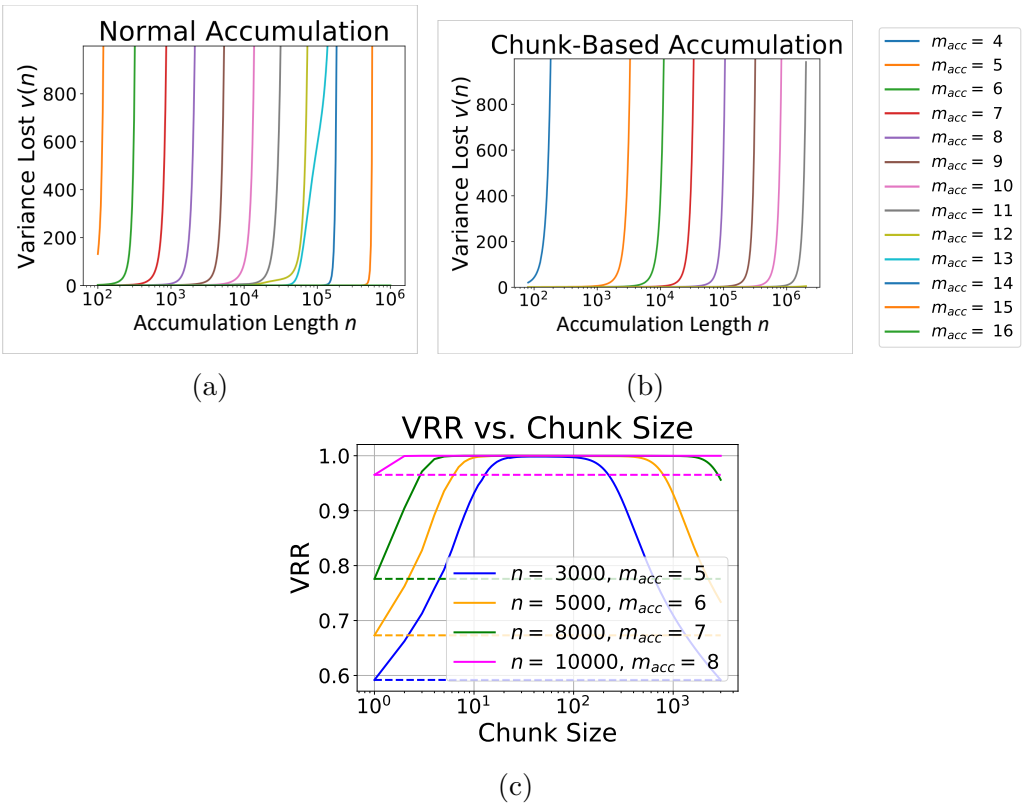


Figure 5.5: Normalized variance lost as a function of accumulation length for different values of m_{acc} for (a) a normal accumulation (no chunking) and (b) a chunk-based accumulation (chunk size of 64). The "knees" in each plot correspond to the maximum accumulation length for a given precision which indicates how the VRR is to be used to select a suitable precision. (c) VRR as a function of chunk-size for several accumulation setups. The dashed lines correspond to the value of the VRR when no chunking is used. The flat maximas indicate that the exact choice of a chunking size is not of paramount importance.

$v(n) < 50$ and n decreases, the variance lost quickly drops to zero. This choice of a cut-off value is thus chosen based purely on the accumulation length and precision.

In addition, when performing chunk-based accumulation, the chunk size is a hyperparameter that, a priori, cannot be determined trivially. Castaldo et al. [134] identified an optimal chunk size minimizing the loose upper bound on the accumulation error they derived. In practice, they did not find the accumulation error to be sensitive to the chunk-size. Neither did Wang et al. [74] who performed numerical simulations. By sweeping the

chunk size and observing the accumulation behavior on synthetic data, it was found that chunking significantly reduces accumulation error as long as the chunk size is not too small nor too large. Using our analysis, we provide a theoretical justification. Figure 5.5 (c) shows the VRR for various accumulation setups, including chunking when the chunk size is swept. For each case we see that chunking raises the VRR to a value close to unity. Furthermore, the VRR curve in that regime is "flat", meaning that a specific value of chunk size does not matter as long as it is not too small or too large. One intuition is that a moderate chunk size prevents both inter- and intra-chunk accumulations to be as large as the original accumulation. In our upcoming chunking experiments we use a chunk size of 64 as was done by [74].

5.5 Numerical Results

Using the above analysis, we predict the mantissa precisions required by the three GEMM functions for training the following networks: ResNet 32 on the CIFAR-10 dataset, ResNet 18 and AlexNet on the ImageNet dataset. Those benchmarks were chosen due to both their popularity and topologies which present large accumulation lengths, making them good candidates against which we can verify our work. We use the same configurations as [74]; in particular, we use 6 bit of exponents in the accumulations, quantize the intermediate tensors to (1,5,2) floating-point format, and keep the final layer's precision in 16 bit. The technique of loss scaling [131] is used in order to limit underflows of activation gradients. A single scaling factor of 1000 was used for all models tested.

In order to realize rounding of partial sums, we modify the CUDA code of the GEMM function (which, in principle, can be done using any framework). In particular, we add a custom rounding function where the partial sum accumulation occurs. Quantization of dot product inputs is handled similarly.

The predicted precisions for each network and layer/block are listed in Table 5.1 for the case of normal and chunk-based accumulation with a chunk size of 64. Several insights are to be noted.

- The required accumulation precision for CIFAR-10 ResNet 32 is in general lower than that of the ImageNet networks. This is simply

Table 5.1: The predicted precisions required for all accumulations of our considered networks. Each table entry is an ordered tuple of two values which correspond to the predicted mantissa precision of both normal and chunk-based accumulations, respectively. The precision requirements of FWD and BWD are typically smaller than those of GRAD. The latter needs the most precision for layers/blocks close to the input as the size of the feature maps is highest in the early stages. The benefits of chunking are nonlinear but range from 1 to 6 bit.

CIFAR-10 ResNet 32

Layer(s)	Conv 0	ResBlock 1	ResBlock 2	Resblock 3
FWD	(6,5)	(6,5)	(7,5)	(7,5)
BWD	N/A	(6,5)	(7,5)	(8,5)
GRAD	(11,8)	(11,8)	(10,6)	(9,6)

ImageNet ResNet 18

Layer(s)	Conv 0	ResBlock 1	ResBlock 2	Resblock 3	ResBlock 4
FWD	(9,6)	(7,5)	(8,5)	(8,5)	(9,6)
BWD	N/A	(8,6)	(9,6)	(9,6)	(10,6)
GRAD	(15,10)	(15,9)	(12,8)	(10,6)	(9,5)

ImageNet AlexNet

Layer	Conv 1	Conv 2	Conv 3	Conv 4	Conv 5	FC 1	FC 2
FWD	(7,5)	(9,5)	(9,5)	(8,5)	(8,5)	(9,6)	(8,5)
BWD	N/A	(8,5)	(8,5)	(10,8)	(8,5)	(8,5)	(8,5)
GRAD	(10,7)	(9,6)	(8,6)	(6,5)	(6,5)	(6,5)	(6,5)

because the network topology imposes shorter dot products.

- Though topologically the convolutional layers in the two ImageNet networks are similar, the precision requirements do vary. Specifically, the GRAD accumulation depends on the feature map dimension which is mostly dataset dependent, yet AlexNet requires less precision than ResNet 18. This is because the measured sparsity of the operands was found to be much higher for AlexNet.

- Figure 5.5 suggests that chunking decreases the precision requirements significantly. This is indeed observed in Table 5.1, where we see that the benefits of chunking reach up to 6 bit in certain accumulations, e.g., the GRAD accumulation in the first ResBlock of ImageNet ResNet 18.

Because our predicted precision assignment ensures the VRR of all GEMM accumulations to be close to unity, we expect reduced precision training to

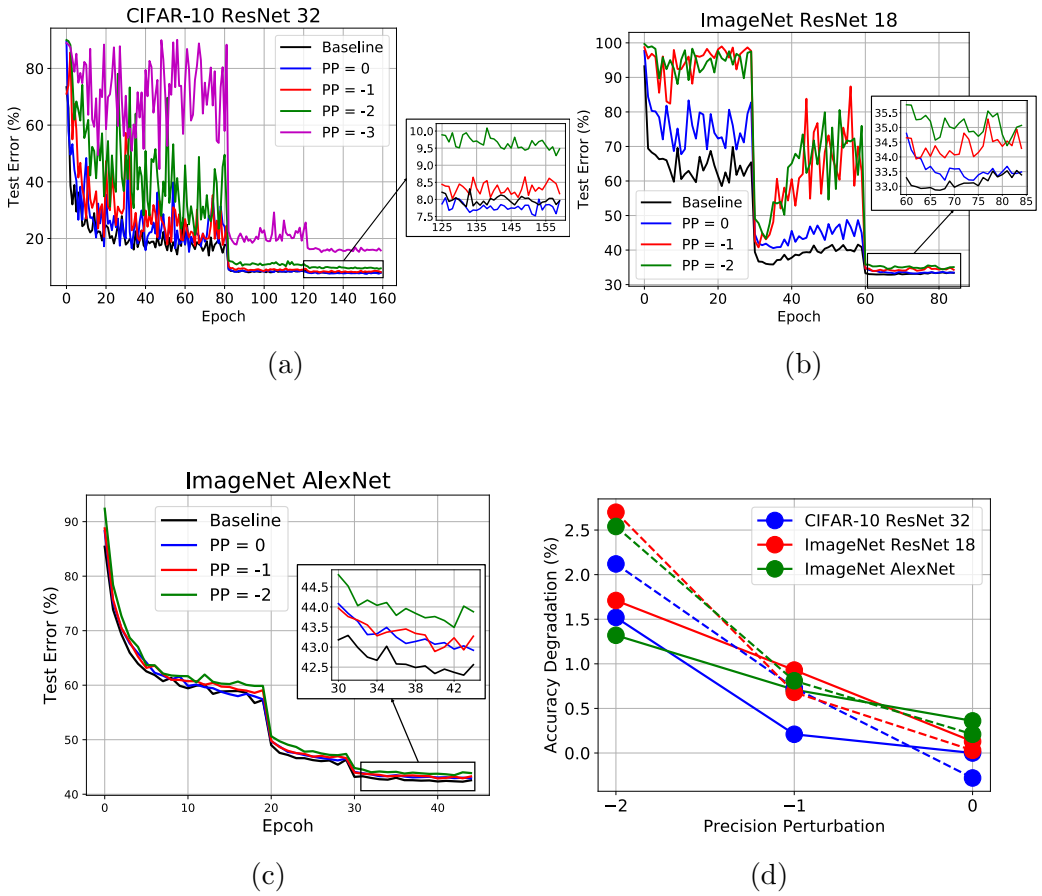


Figure 5.6: Convergence curves for (a) CIFAR-10 ResNet 32, (b) ImageNet ResNet 18, and (c) ImageNet AlexNet, and (d) final accuracy degradation with respect to the baseline as a function of precision perturbation (PP). The solid and dashed lines correspond to the no chunking and chunking case, respectively. Using our predicted precision assignment, the converged test error is close to the baseline (no more than 0.5% degradation) but increases significantly when the precision is further reduced.

converge with close fidelity to the baseline. Since our work focuses on accumulation precision, in our experiments, the baseline denotes accumulation in full precision. For a fair comparison, all upcoming results use (1,5,2) representation precision. Thus, the effects of reduced precision representation are not taken into account.

The goal of our experiments is to investigate both the validity and conservatism of our analysis. In Figure 5.6, we plot the convergence curves when training with our predicted accumulation precision for a normal accumulation. The runs corresponding to chunk-based accumulations were also performed but are omitted since the trend is similar. Furthermore, we re-

peat all experiments with precision perturbation (PP), meaning a specific reduction in precision with respect to our prediction. For instance, $PP = 0$ indicates our prediction while $PP = -1$ corresponds to a 1 bit reduction. Finally, in order to better visualize how the accumulation precision affect convergence, we plot in Figure 5.6 (d) the accuracy degradation as a function of precision perturbation for each of our three networks with both normal and chunk-based accumulations. The following are to be noted:

- When $PP = 0$, the converged accuracy always lies within 0.5% of the baseline, a strong indication of the validity of our analysis. We use a 0.5% accuracy cut-off with respect to the baseline as it corresponds to an approximate error bound for neural networks obtained by changing the random numbers seed [10, 137].

- When $PP < 0$, a noticeable accuracy degradation is observed, most notably for ImageNet ResNet 18. The converged accuracy is no longer within 0.5% of the baseline. Furthermore, a clear trend observed is that the higher the perturbation, the worse the degradation.

- ImageNet AlexNet is more robust to perturbation than the two ResNets. While $PP = -1$ causes a degradation strictly $> 0.5\%$, it is not much worse than the $PP = 0$ case. This observation aligns with that from neural net quantization that Alexnet is robust due to its overly parameterized network structure [138]. But the trend of increasing degradation remains the same.

- Figure 5.6 (d) suggests that the effects of PP are more pronounced for a chunk-based accumulation. Since the precision assignment itself is lower (Table 5.1), a specific precision perturbation corresponds to a relatively higher change. For example, decreasing 1 bit from a 6 bit assignment is more important than decreasing 1 bit from a 10 bit assignment. Further justification can be obtained by comparing Figures 5.5 (a) and 5.5 (b) where consecutive lines are less closely aligned for the chunk-based accumulation, indicating more sensitivity to precision perturbation.

Thus, overall, our predictions are adequate and close to the limits beyond which training becomes unstable. These are very encouraging signs that our analysis is both valid and tight.

5.6 Summary

We have presented an analytical method to predict the precision required for partial sum accumulation in the three GEMM functions in deep learning training. Our results prove that our method is able to accurately pinpoint the minimum precision needed for the convergence of benchmark networks to the full precision baseline. Our theoretical concepts are application agnostic, and an interesting extension would be to consider recurrent architectures such as LSTMs. In particular, training via backpropagation in time could make the GRAD accumulation very large depending on the number of past time-steps used. In such a case, our analysis is of great relevance to training precision optimization. On the practical side, this analysis is a useful tool for hardware designers implementing reduced-precision FPUs, who in the past have resorted to computationally prohibitive brute-force emulations. We believe this work addresses a critical missing link on the path to truly low-precision floating-point hardware for DNN training.

5.7 Addendum: Proofs of Theoretical Results

Preliminaries and Assumptions

The analysis of stability of sums under reduced-precision floating-point accumulation is a classically difficult problem. Indeed, statistically characterizing recursive rounding effects is often mathematically intractable. Therefore, most prior works have considered worst-case analyses and provided loose bounds on accuracy of computation as a function of precision [134]. In contrast, the results presented in this chapter were found to be tight; however, they necessitate a handful of assumptions for mathematical tractability. These assumptions are listed and discussed hereafter, and the proofs of the theoretical results presented in the chapter follow in this section.

Assumption 1: *The product terms* $\{p_i\}_{i=1}^{n}$ *are statistically independent, zero-mean, and have the same variance* σ_p^2.

This assumption, which was mentioned in Section 5.3, is a standard one in works where the issue of variance in deep learning is studied [135]. Note that as a result we have $Var(s_n)_{\text{ideal}} = \mathbf{E}_{\text{ideal}}\left[s_n^2\right]$ because $\mathbf{E}_{\text{ideal}}\left[s_n\right] = 0$.

Assumption 2: *Computation in reduced precision floating-point arithmetic is unbiased with respect to the baseline.*

This assumption is also standard in works studying quantization noise and effects [89, 115, 118]. An important implication is that $Var(s_n)_{\text{swamping}} = \mathbf{E}_{\text{swamping}}[s_n^2]$ because $\mathbf{E}_{\text{swamping}}[s_n] = \mathbf{E}_{\text{ideal}}[s_n] = 0$.

Assumption 3: *The accumulation is monotonic in the iterations leading to a full swamping event.*

This assumption means that we shall focus on a typical scenario where the partial sums $\{s_i\}_{i=1}^n$ grow in magnitude while product terms $\{p_i\}_{i=1}^n$ are of the same order. In other words, we do not consider catastrophic events where full swamping occurs unexpectedly (the probability of such event is small in any case).

Assumption 4: *We consider a partial sum s_i that experiences full swamping in reduced precision accumulation to be statistically independent from prior partial sums $s_{i'}$ for $i' < i$.*

All partial sums $\{s_i\}_{i=1}^n$ are statistically dependent as they are computed in a recursive manner. Our assumption is that should swamping noise be so significant to cause full swamping, then the recursive dependence on prior partial sums is broken.

Assumption 5: *Once a full swamping event occurs, the computation of partial sum accumulation is halted.*

It is possible, but unlikely, that the computation might recover from swamping. A partial recovery of the computation is also possible but causes negligible effects on the final result. Thus, such scenarios are neglected. Assumptions 3, 4, and 5 will be particularly useful for mathematical tractability in the proof of Lemma 8.

Assumption 6: *The bits of the mantissa representation of partial sums $\{s_i\}_{i=1}^n$ and product terms $\{p_i\}_{i=1}^n$ are equally likely to be zero or one.*

This is yet again a standard assumption in quantization noise analysis which will be particularly useful in the proof of Theorem 4.

Proof of Lemma 8

In order to compute the VRR, we first need to compute $Var(s_n)$ during swamping, i.e., $Var(s_n)_{\text{swamping}} = \mathbf{E}_{\text{swamping}}[s_n^2]$, where the equality holds by

application of Assumptions 1 and 2. To do so, we rely on the Law of Total Expectation. Indeed, assume that $\mathcal{A}$ is the set of events that describe all manners in which the accumulation s_n experiencing swamping can occur, and let $P(A)$ be the probability of event $A \in \mathcal{A}$. Hence, by the Law of Total Expectation, we have that

$$Var(s_n)_{\text{swamping}} = \sum_{A \in \mathcal{A}} \mathbf{E}\left[s_n^2 \middle| A \right] P(A). \tag{5.7}$$

It is in fact a difficult task to enumerate and describe all events in $\mathcal{A}$. Thus we consider a reduced set of events $\hat{\mathcal{A}} \subset \mathcal{A}$ which is representative enough of the manners in which the accumulation occurs, yet tractable so that it can be used as a surrogate to $\mathcal{A}$ in (5.7). We shall form the set $\hat{\mathcal{A}}$ by application of Assumption 3, 4, and 5 as we proceed.

We consider a scenario where the first occurrence of full swamping is at iteration i of the summation for $i = 2 \dots n-1$. This happens if:

$$|s_i| > 2^{m_{acc}} |p_{i+1}| \quad \text{and} \quad |s_{i'}| < 2^{m_{acc}} |p_{i'+1}| \ \text{ for } i' = 1 \dots i-1. \tag{5.8}$$

Instead of looking at the actual absolute value of an incoming product term, we replace it by its typical value of σ_p. Furthermore, we simplify the condition of no swamping prior to iteration i by only considering the accumulated sum at the previous iteration. This is due to Assumption 3 which allows us to consider a simplified scenario where the accumulation is monotonic in the iterations leading to full swamping. Hence, our simplified condition for the first occurrence of full swamping at iteration i is given by:

$$|s_{i-1}| < 2^{m_{acc}} \sigma_p < |s_i|.$$

As iteration i corresponds to a full swamping event, we may invoke Assumption 4 and treat each of the two inequalities above independently. Finally, we invoke Assumption 5: since full swamping happens at iteration i, then the result of the accumulation is $s_n = s_i$ since the computation in the following iterations is halted.

Thus, the event set $\hat{\mathcal{A}}$ we construct for our analysis consists of the mutually exclusive events $\{A_i\}_{i=2}^{n-1}$ where A_i is the event that full swamping occurs for the first time at iteration i under the above assumptions. The condition for

event i to happen is given by (5.8). By the Central Limit Theorem, which we use by virtue of the s_i being a summation of independent, identically distributed product terms, we have that $s_i \sim \mathcal{N}(0, i\sigma_p^2)$, so that:

$$P(A_i) = 2Q\left(\frac{2^{m_{acc}}}{\sqrt{i}}\right)\left(1 - 2Q\left(\frac{2^{m_{acc}}}{\sqrt{i-1}}\right)\right) = q_i. \tag{5.9}$$

Furthermore, by Assumption 5 we have:

$$\mathbf{E}\left[s_n^2 \middle| A_i\right] = \mathbf{E}\left[s_i^2\right] = i\sigma_p^2. \tag{5.10}$$

We also add to our space of events, the event A_n where no full swamping occurs over the course of the accumulation. This event happens if $|s_n| < 2^{m_{acc}}\sigma_p$ and thus has probability $P(A_n) = 1 - 2Q\left(\frac{2^{m_{acc}}}{\sqrt{n}}\right) = \tilde{q}_n$. Since this event corresponds to the ideal scenario, we have $\mathbf{E}\left[s_n^2 \middle| A_n\right] = n\sigma_p^2$.

Thus, under the above conditions we have:

$$Var(s_n)_{\text{swamping}} = \frac{1}{k}\sum_{i=2}^{n} \mathbf{E}\left[s_n^2 \middle| A_i\right] P(A_i) = \frac{\sigma_p^2}{k}\left(\sum_{i=2}^{n-1} iq_i + n\tilde{q}_n\right), \tag{5.11}$$

where $k = \sum_{i=2}^{n-1} q_i + \tilde{q}_n$ is a normalization constant needed as $\hat{\mathcal{A}}$ does not describe the full probability space.

Consequently we obtain (5.1) as formula for the VRR completing the proof for Lemma 8.

Proof of Theorem 4

First, we do not change the description of the events $\{A_i\}_{i=2}^{n-1}$ above. Notably, by Assumption 5, once full swamping occurs, i.e., $|s_i| > 2^{m_{acc}}\sigma_p$, the computation is stuck and stops. The probability of this event is $P(A_i)$ as above. However, to account for partial swamping, we alter the result of $\mathbf{E}\left[s_n^2 \middle| A_i\right]$. Indeed, partial swamping causes additional loss of variance. When the input product terms have m_p bits of mantissa, then before event A_i can occur, the computation should go through each of the m_p stages described in Figure 5.4. We again use Assumption 3 and consider a typical scenario for each of the m_p stages of partial swamping preceding a full swamping event whereby the

accumulation is monotonic and the magnitude of the incoming product term is close to its typical value σ_p. Under this assumption, stage j is expected to happen for the following number of iterations:

$$N_j = 2^{m_{acc}+1-(m_p-j)} = 2^{m_{acc}-m_p+j+1} \tag{5.12}$$

for $j = 1 \dots m_p$. At stage j, j least significant bits in the representation of the incoming product term are truncated (swamped). The variance lost because of this truncation, which we call fractional variance loss $\mathbf{E}[f_j^2]$, can be computed by assuming the truncated bits are equally likely to be 0 or 1 (Assumption 6), so that:

$$\mathbf{E}[f_j^2] = \sigma_p^2 \left[\sum_{k=0}^{2^j-1} \frac{1}{2^j} \left(2^{-m_p} k\right)^2 \right] = \sigma_p^2 2^{-2m_p} \frac{(2^j-1)(2^{j+1}-1)}{6}. \tag{5.13}$$

Hence, the total fractional variance lost before the occurrence of even A_i is $N_j \mathbf{E}[f_j^2]$. Thus, we update the value of variance conditioned on A_i as follows:

$$\begin{aligned} \mathbf{E}\left[s_n^2 \middle| A_i \right] &= \left(i\sigma_p^2 - N_j \mathbf{E}[f_j^2] \right)_+ \\ &= \sigma_p^2 \left(i - \frac{2^{m_{acc}-3m_p}}{3} \sum_{j=1}^{m_p} 2^j (2^j-1)(2^{j+1}-1) \right)_+, \end{aligned} \tag{5.14}$$

where we used the operator $(x)_+ = \begin{cases} x \text{ if } x > 0 \\ 0 \text{ otherwise} \end{cases}$ in order to guarantee that the variance is positive. Effectively, we neglect the events A_i where i is so small that the variance retained is less than the variance lost due to partial swamping. In other words, an event whereby full swamping occurs very early in the accumulation is considered to have zero probability and we replace $P(A_i)$ in (5.9) by:

$$P(A_i) = q_i \mathbf{1}_{\{i>\alpha\}}, \tag{5.15}$$

where $q_i = 2Q\left(\frac{2^{m_{acc}}}{\sqrt{i}}\right)\left(1 - 2Q\left(\frac{2^{m_{acc}}}{\sqrt{i-1}}\right)\right)$ as in (5.9), $\mathbf{1}$ is the indicator function, and $\alpha = \frac{2^{m_{acc}-3m_p}}{3} \sum_{j=1}^{m_p} 2^j(2^j-1)(2^{j+1}-1)$ as in (5.14).

In addition, some boundary conditions need to be accounted for. These

include the cases when no full swamping happens before the accumulation is complete but partial swamping does happen. We again consider a typical scenario as above and append our event set $\mathcal{A}$ with $m_p - 1$ boundary events $\left\{A'_{j_r}\right\}_{j_r=2}^{m_p}$, where the event A'_{j_r} corresponds to the case where the computation has gone through stage $j_r - 1$ of partial swamping but has not reached stage j_r yet. The condition for this event is $\sigma_p 2^{m_{acc}-m_p+j_r-1} < |s_n| < \sigma_p 2^{m_{acc}-m_p+j_r}$ and occurs typically for up to N_{j_r-1} iterations. The total fractional variance lost is:

$$\sigma_p^2 \left[\sum_{j=1}^{j_r-1} N_j \sum_{k=0}^{2^j-1} \frac{1}{2^j} \left(2^{-m_p} k\right)^2\right] = \sigma_p^2 \frac{2^{m_{acc}-3m_p}}{3} \sum_{j=1}^{j_r-1} 2^j (2^j - 1)(2^{j+1} - 1). \tag{5.16}$$

Hence,

$$\mathbf{E}\left[s_n^2 \middle| A'_{j_r}\right] = \sigma_p^2 \left(n - \alpha_{j_r}\right)_+, \tag{5.17}$$

where $\alpha_{j_r} = \frac{2^{m_{acc}-3m_p}}{3} \sum_{j=1}^{j_r-1} 2^j(2^j-1)(2^{j+1}-1)$. And the probability of event A'_{j_r} is given by:

$$P(A'_{j_r}) = q'_{j_r} \mathbf{1}_{\{n>\alpha_{j_r}\}}, \tag{5.18}$$

where $q'_{j_r} = N_{j_r-1} 2Q\left(\frac{2^{m_{acc}-m_p+j_r-1}}{\sqrt{n}}\right)\left(1 - 2Q\left(\frac{2^{m_{acc}-m_p+j_r}}{\sqrt{n}}\right)\right)$, and the multiplication by N_{j_r-1} reflects the number of iterations the event may occur for.

Finally, the event A_n is updated and corresponds to the case where neither partial nor full swamping occurs. The condition for this event is $|s_n| < 2^{m_{acc}-m_p+1}$ and has a probability $P(A_n) = 1 - 2Q\left(\frac{2^{m_{acc}-m_p+1}}{\sqrt{n}}\right)$.

Putting things together, we use the law of total expectation as in (5.7) to compute:

$$\begin{aligned} &Var(s_n)_{\text{swamping}} = \\ &\frac{\sigma_p^2}{k}\left[\sum_{i=2}^{n-1}(i-\alpha)_+ q_i \mathbf{1}_{\{i>\alpha\}} + \sum_{j_r=2}^{m_p}(n-\alpha_{j_r})_+ q'_i \mathbf{1}_{\{n>\alpha_{j_r}\}} + nk_3\right], \end{aligned} \tag{5.19}$$

where $k = k_1 + k_2 + k_3$, $k_1 = \sum_{i=2}^{n-1} P(A_i)$, $k_2 = \sum_{j_r=2}^{m_p} P(A'_{j_r})$, and $k_3 =$

$P(A_n)$. Hence, the formula for the VRR in (5.2) in the theorem follows and this concludes the proof.

Proof of Corollary 1

Applying the above analysis, we may compute the variance of the intermediate results as $\sigma_p^2 n_1 VRR(m_{acc}, m_p, n_1)$. To compute the variance of the final result, first note that the mantissa precision of the incoming terms to the inter-chunk accumulation (the results from the intra-chunk accumulation) is $\min(m_{acc}, m_p + \log_2(n_1))$. The reason is that since the intra-chunk accumulation uses m_{acc} mantissa bits, the mantissa cannot grow beyond m_{acc} due to the rounding nature of the floating-point accumulation. However, if m_{acc} is large enough and n_1 is small enough, it is most likely that the mantissa has not grown to the maximum. Assuming accumulation of terms having statistically similar absolute value as was done for the VRR analysis, then the bit growth of the mantissa is logarithmic in n_1 and starts at m_p.

Hence, the variance of the computed result s_n when chunking is used is:

$$\begin{aligned} Var(s_n)_{\text{chunking}} = \sigma_p^2 n_1 VRR(m_{acc}, m_p, n_1) \\ \times n_2 VRR\left(m_{acc}, \min\left(m_{acc}, m_p + \log_2(n_1)\right), n_2\right) \end{aligned} \tag{5.20}$$

and hence the VRR with chunking can be computed using (5.3) in Corollary 1. This completes the proof.

CHAPTER 6

CONCLUSION AND FUTURE WORK

This dissertation has addressed a hard, real problem, that of finite precision deep learning. While prior work and methods abound, none provide analytical guarantees on accuracy. Instead, current practices rely on ad-hoc techniques, largely based on trial-and-error approaches. In contrast, this dissertation addresses the issue of finite precision deep learning with theoretical guarantees. Therefore, the results presented in this dissertation can be used by hardware and software designers in order to implement *accuracy-aware* finite precision algorithms. In this final chapter, we summarize this dissertation's contributions and provide directions for future research.

6.1 Summary of Contributions

This dissertation formulates a theoretical framework for the implementation of deep learning in finite precision. For inference, we theoretically analyzed the worst-case accuracy drop in the presence of weight and activation quantization. Furthermore, we derived an optimal clipping criterion guaranteed to minimize precision of dot-product outputs. For implementations using in-memory computing, this results in a lower ADC precision requirement. We analyzed fixed-point training and presented a methodology for implementing fully quantized back-propagation with close-to-minimal per-tensor precision. Finally, we studied accumulator precision for reduced precision floating-point training using variance analysis techniques. More specifically, our contributions are summarized as follows.

In the context of DNN inference, theoretical bounds on the misclassification rate in presence of limited precision were first derived. Proper precision assignment was obtained employing these bounds, and weight-activation as well as per-layer precision trade-offs were quantified. The proposed princi-

pled precision reduction was applied to a variety of networks and datasets. Additionally, we proposed the OCC to minimize the column ADC precision in in-memory architectures. The SQNR of OCC was shown to be as accurate as the optimal LM. Using bit-slicing techniques, significant energy savings were achieved with minimal accuracy loss. An analytical methodology for proper IMC design subject to DNN accuracy constraint was derived.

In the context of DNN training, we derived a systematic methodology to obtain close-to-minimal per-layer precision requirements for guaranteed statistical similarity between fixed-point and floating-point training. The challenges of quantization noise, inter-layer and intra-layer precision trade-offs, dynamic range, and stability were jointly addressed, and the methodology was applied to several benchmarks. Furthermore, we presented the VRR of a reduced precision accumulation in the context of back-propagation dot-products. Our analysis was applied to determine mantissa precision requirements in floating-point accumulators. Experimentally, the validity and tightness of our analysis were verified across multiple deep learning benchmarks.

The work presented in this dissertation does have a few limitations. First, obtaining theoretical guarantees on a DNN's accuracy does lead to conservative designs. Indeed, minimum precision networks obtained via our analysis were found to have higher requirements (by approximately 5 bit) compared to other methods, particularly in the scope of in-training quantization [58]. One caveat is that such works have no accuracy guarantees and require substantial empirical tuning. Additionally, all accuracy guarantees that were derived in this dissertation have been provided with respect to an existing full precision baseline. Thus, we have not provided absolute guarantees on a network's accuracy in an *a priori* manner. Notwithstanding, it is important to note that such theoretical results do not even exist for full precision networks, i.e., when finite precision effects are excluded.

6.2 Future Research Prospects

The conclusions and insights derived from this dissertation work can provide a strong basis for future research directions.

Upper Bounds on the Accuracy of In-Training Quantization: Much

of the presented work on inference accuracy was related to post-training quantization. In-training quantization can lead to more aggressive results, but lacks a theoretical backing. We posit that an analysis similar to the one provided in this dissertation can be applied to in-training quantization. Specifically, we believe existential upper bounds on the converged accuracy can be derived. Indeed, using mixed-integer programming techniques, worst-case deviation between full precision and quantized dot-products can readily be obtained. By combining such classical results to a variant of our presented analysis, we believe the following question can be answered: Given a network, dataset, and pre-determined precision of activation and weights, what is the best accuracy that training with quantization can possibly achieve?

Theoretical Basis for Distributed Training: Training large models on large datasets necessitates distribution of SGD onto multiple devices [10]. When the number of such devices increases, the latency is dominated by the transmission of gradients and synchronization of all learners. As such, gradient compression techniques, such as quantization, are often applied to accelerate training. Our work on fixed-point training presented in this dissertation can be used to guide the communication strategy between learners. First, the gradient compression aspect can be studied in a manner similar to that of gradient quantization noise. Second, the asynchronous updates and impact on convergence are related to the issues of early stopping and stability discussed when analyzing the feedback loop of the back-propagation algorithm. Thus, a great potential exists to expand our work into the area of distributed learning.

Hardware-Aware Training: A recent approach for generating efficient networks, tailored to the potential deployment platform, is to embed hardware models within the training algorithm. Such methods include neural architecture search and are known to be prohibitively compute intensive [69]. Our developed analytical models derived in this dissertation can be used to significantly speed up such algorithms. Indeed, rather than invoking time-consuming trial and error routines, the impact of hardware implementation on the accuracy can be instantly predicted. Therefore, the time complexity of a search can be reduced by one whole dimension.

Analytical Guarantees on Adversarial Robustness: While DNNs have achieved impressive success when deployed with natural data, concerns have started to appear about their vulnerability. Indeed, specifically tailored

perturbation attacks were shown to implode the accuracy of state-of-the-art networks [139]. Many techniques have been proposed to robustify networks against adversarial attacks [140]. Unfortunately, prior work has been largely empirical, lacking a theoretical understanding. We postulate that our theoretical work developed in this dissertation can advance the understanding of adversarial robustness. Indeed, our work has employed general noise models for quantization as a building block. Interestingly, two of the state-of-the-art adversarial attacks, DeepFool [141] and JSMA [142], employ the concept of activation noise gains to compute adversarial samples. Thus, we believe important insights from our work on quantization can be leveraged in the context of adversarial machine learning.

Applications Beyond Deep Learning: Many applications are benefiting today from the advances of machine learning. However, some signal processing applications have known optimal algorithms that are much simpler and theoretically sound. Implementation of such algorithms in finite precision is an important and difficult problem. Indeed, the noise tolerance of signal processing applications is much lower than that of deep learning. As such, statistical error compensation techniques can be used in tandem with finite precision implementation. Such applications include radar signal processing chains where finite impulse response (FIR) filters are the main building block. Such filters are traditionally implemented using time-domain convolution which was found to be weak in tolerating quantization noise and non-idealities in IMC architectures. One option for statistical error compensation is filtering via frequency-domain convolution which has a low peak-to-average ratio and best utilization of the IMC structure.

By means of theoretical analyses, the work presented in this dissertation constitutes a first important step in the principled design and implementation of finite precision deep learning systems. The nature of the methods presented can be extended to tangential problems in the broader scope of training and inference. As mentioned above, opportunities exist in the context of in-training quantization, distributed and hardware-driven training, adversarial machine learning and applications in signal processing. It is hoped that such extensions of our work will occur in the future.

REFERENCES

[1] A. Krizhevsky et al., "ImageNet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems*, 2012, pp. 1097–1105.

[2] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2009, pp. 248–255.

[3] K. He et al., "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 770–778.

[4] H. Sak et al., "Learning acoustic frame labeling for speech recognition with recurrent neural networks," in *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2015, pp. 4280–4284.

[5] W. Chan et al., "Listen, attend and spell: A neural network for large vocabulary conversational speech recognition," in *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2016, pp. 4960–4964.

[6] W. Zhang et al., "Bridging the gap between training and inference for neural machine translation," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 2019, pp. 4334–4343.

[7] J. Gao et al., "Neural approaches to conversational ai," in *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, 2018, pp. 1371–1374.

[8] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton et al., "Mastering the game of Go without human knowledge," *Nature*, vol. 550, no. 7676, pp. 354–359, 2017.

[9] Y. Taigman et al., "Deepface: Closing the gap to human-level performance in face verification," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 1701–1708.

[10] P. Goyal, P. Dollár, R. Girshick, P. Noordhuis, L. Wesolowski, A. Kyrola, A. Tulloch, Y. Jia, and K. He, "Accurate, large minibatch SGD: training ImageNet in 1 hour," *arXiv preprint arXiv:1706.02677*, 2017.

[11] R. Sanchez-Iborra and A. F. Skarmeta, "TinyML-enabled frugal smart objects: Challenges and opportunities," *IEEE Circuits and Systems Magazine*, vol. 20, no. 3, pp. 4–18, 2020.

[12] N. Shanbhag, "Energy-efficient machine learning in silicon: A communications-inspired approach," *ICML On-device Intelligence Workshop*, 2016.

[13] S. Gupta, "Real time face recognition on an edge computing device," in *Proceedings of the 2020 9th International Conference on Software and Computer Applications*, 2020, pp. 110–113.

[14] H. Dbouk et al., "KeyRAM: A 0.34 uj/decision 18 k decisions/s recurrent attention in-memory processor for keyword spotting," in *2020 IEEE Custom Integrated Circuits Conference (CICC)*. IEEE, 2020, pp. 1–4.

[15] C. J. Baby, F. A. Khan, and J. Swathi, "Home automation using IoT and a chatbot using natural language processing," in *2017 Innovations in Power and Advanced Computing Technologies (i-PACT)*. IEEE, 2017, pp. 1–6.

[16] S. Liu, J. Tang, Z. Zhang, and J.-L. Gaudiot, "CAAD: Computer Architecture for Autonomous Driving," *arXiv preprint arXiv:1702.01894*, 2017.

[17] R. Ghaffari, J. A. Rogers, and T. R. Ray, "Recent progress, challenges, and opportunities for wearable biochemical sensors for sweat analysis," *Sensors and Actuators B: Chemical*, vol. 332, p. 129447, 2021.

[18] B. McMahan and D. Ramage, "Federated learning: Collaborative machine learning without centralized training data," 2017.

[19] W. Y. B. Lim, N. C. Luong, D. T. Hoang, Y. Jiao, Y.-C. Liang, Q. Yang, D. Niyato, and C. Miao, "Federated learning in mobile edge networks: A comprehensive survey," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 3, pp. 2031–2063, 2020.

[20] J. Lin, W.-M. Chen, Y. Lin, J. Cohn, C. Gan, and S. Han, "MCU-Net: Tiny deep learning on IoT devices," *arXiv preprint arXiv:2007.10319*, 2020.

[21] Y.-H. Chen, T. Krishna, J. S. Emer, and V. Sze, "Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks," *IEEE Journal of Solid-State Circuits*, vol. 52, no. 1, pp. 127–138, 2017.

[22] N. P. Jouppi et al., "In-datacenter performance analysis of a tensor processing unit," in *Proceedings of the 44th Annual International Symposium on Computer Architecture*. ACM, 2017, pp. 1–12.

[23] B. Fleischer, S. Shukla, M. Ziegler, J. Silberman, J. Oh, V. Srinivasan, J. Choi, S. Mueller, A. Agrawal, T. Babinsky et al., "A scalable multi-TeraOPS deep learning processor core for AI training and inference," in *2018 IEEE Symposium on VLSI Circuits*. IEEE, 2018, pp. 35–36.

[24] X. Sun, N. Wang, C.-Y. Chen, J. Ni, A. Agrawal, X. Cui, S. Venkataramani, K. El Maghraoui, V. V. Srinivasan, and K. Gopalakrishnan, "Ultra-low precision 4-bit training of deep neural networks," *Advances in Neural Information Processing Systems*, vol. 33, 2020.

[25] U. Köster, T. Webb, X. Wang, M. Nassar, A. K. Bansal, W. Constable, O. Elibol, S. Hall, L. Hornof, A. Khosrowshahi et al., "Flexpoint: An adaptive numerical format for efficient training of deep neural networks," in *Advances in Neural Information Processing Systems*, 2017, pp. 1740–1750.

[26] B. Zimmer, R. Venkatesan, Y. S. Shao, J. Clemons, M. Fojtik, N. Jiang, B. Keller, A. Klinefelter, N. Pinckney, P. Raina et al., "A 0.32–128 tops, scalable multi-chip-module-based deep neural network inference accelerator with ground-referenced signaling in 16 nm," *IEEE Journal of Solid-State Circuits*, vol. 55, no. 4, pp. 920–932, 2020.

[27] J. Zhang et al., "In-memory computation of a machine-learning classifier in a standard 6T SRAM array," *IEEE Journal of Solid-State Circuits*, vol. 52, no. 4, pp. 915–924, April 2017.

[28] M. Kang et al., "A multi-functional in-memory inference processor using a standard 6T SRAM array," *IEEE Journal of Solid-State Circuits*, vol. 53, no. 2, pp. 642–655, 2018.

[29] S. Yin et al., "XNOR-SRAM: In-memory computing SRAM macro for binary/ternary deep neural networks," *IEEE Journal of Solid-State Circuits*, vol. 55, no. 6, pp. 1733–1743, 2020.

[30] A. Biswas and A. P. Chandrakasan, "Conv-RAM: An energy-efficient SRAM with embedded convolution computation for low-power CNN-based machine learning applications," in *IEEE International Solid-State Circuits Conference (ISSCC)*, 2018, pp. 488–490.

[31] S. K. Gonugondla, M. Kang, and N. R. Shanbhag, "A variation-tolerant in-memory machine learning classifier via on-chip training," *IEEE Journal of Solid-State Circuits*, vol. 53, no. 11, pp. 3163–3173, 2018.

[32] W.-S. Khwa et al., "A 65nm 4Kb algorithm-dependent computing-in-memory SRAM unit-macro with 2.3 ns and 55.8 TOPS/W fully parallel product-sum operation for binary DNN edge processors," in *IEEE International Solid-State Circuits Conference (ISSCC)*, 2018, pp. 496–498.

[33] H. Valavi et al., "A mixed-signal binarized convolutional-neural-network accelerator integrating dense weight storage and multiplication for reduced data movement," in *2018 IEEE Symposium on VLSI Circuits*. IEEE, 2018, pp. 141–142.

[34] J. Kim et al., "Area-efficient and variation-tolerant in-memory BNN computing using 6T SRAM array," in *2019 IEEE Symposium on VLSI Circuits*. IEEE, 2019, pp. 118–119.

[35] Q. Dong et al., "A 351 TOPS/W and 372.4 GOPS compute-in-memory SRAM macro in 7nm FinFET CMOS for machine learning applications," in *IEEE International Solid-State Circuits Conference (ISSCC)*, 2020, pp. 242–243.

[36] J.-W. Su et al., "A 28nm 64Kb inference-training two-way transpose multibit 6T SRAM compute-in-memory macro for AI edge chips," in *IEEE International Solid-State Circuits Conference (ISSCC)*, 2020, pp. 240–241.

[37] X. Si et al., "A 28nm 64Kb 6T SRAM computing-in- memory macro with 8b MAC operation for AI edge chips," in *IEEE International Solid-State Circuits Conference (ISSCC)*, 2020, pp. 246–247.

[38] C.-X. Xue et al., "A 22nm 2Mb ReRAM compute-in-memory macro with 121-28TOPS/W for multibit MAC computing for tiny AI edge devices," in *IEEE International Solid-State Circuits Conference (ISSCC)*, 2020, pp. 244–245.

[39] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[40] G. Huang, Z. Liu, K. Q. Weinberger, and L. van der Maaten, "Densely connected convolutional networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, vol. 1, no. 2, 2017, p. 3.

[41] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, 2010, pp. 249–256.

[42] I. J. Goodfellow et al., "Maxout networks." *ICML (3)*, vol. 28, pp. 1319–1327, 2013.

[43] M. Lin et al., "Network in network," *arXiv preprint arXiv:1312.4400*, 2013.

[44] N. Srivastava et al., "Dropout: A simple way to prevent neural networks from overfitting." *Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[45] D. Lin et al., "Fixed point quantization of deep convolutional networks," in *Proceedings of The 33rd International Conference on Machine Learning*, 2016, pp. 2849–2858.

[46] M. Nagel, M. v. Baalen, T. Blankevoort, and M. Welling, "Data-free quantization through weight equalization and bias correction," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 1325–1334.

[47] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International Conference on Machine Learning*, 2015, pp. 448–456.

[48] Y. Lin et al., "PredictiveNet: an energy-efficient convolutional neural network via zero prediction," in *Circuits and Systems (ISCAS), 2017 IEEE International Symposium on*. IEEE, 2017.

[49] S. Jain, S. Venkataramani, V. Srinivasan, J. Choi, K. Gopalakrishnan, and L. Chang, "BiScaled-DNN: Quantizing long-tailed datastructures with two scale factors for deep neural networks," in *2019 56th ACM/IEEE Design Automation Conference (DAC)*. IEEE, 2019, pp. 1–6.

[50] Y. Bengio, N. Léonard, and A. Courville, "Estimating or propagating gradients through stochastic neurons for conditional computation," *arXiv preprint arXiv:1308.3432*, 2013.

[51] I. Hubara et al., "Binarized neural networks," in *Advances in Neural Information Processing Systems*, 2016, pp. 4107–4115.

[52] M. Rastegari et al., "XNOR-Net: ImageNet classification using binary convolutional neural networks," in *European Conference on Computer Vision*. Springer, 2016, pp. 525–542.

[53] S. Zhou, Y. Wu, Z. Ni, X. Zhou, H. Wen, and Y. Zou, "DoReFa-Net: Training low bitwidth convolutional neural networks with low bitwidth gradients," *arXiv preprint arXiv:1606.06160*, 2016.

[54] Z. Cai, X. He, J. Sun, and N. Vasconcelos, "Deep learning with low precision by half-wave Gaussian quantization," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 5918–5926.

[55] A. Zhou, A. Yao, Y. Guo, L. Xu, and Y. Chen, "Incremental network quantization: Towards lossless cnns with low-precision weights," *arXiv preprint arXiv:1702.03044*, 2017.

[56] E. Park, J. Ahn, and S. Yoo, "Weighted-entropy-based quantization for deep neural networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 5456–5464.

[57] D. Zhang, J. Yang, D. Ye, and G. Hua, "LQ-Nets: Learned quantization for highly accurate and compact deep neural networks," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 365–382.

[58] J. Choi, Z. Wang, S. Venkataramani, P. I.-J. Chuang, V. Srinivasan, and K. Gopalakrishnan, "PACT: Parameterized clipping activation for quantized neural networks," *arXiv preprint arXiv:1805.06085*, 2018.

[59] F. Tung and G. Mori, "Clip-Q: Deep network compression learning by in-parallel pruning-quantization," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 7873–7882.

[60] J. Yang, X. Shen, J. Xing, X. Tian, H. Li, B. Deng, J. Huang, and X.-S. Hua, "Quantization networks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 7308–7316.

[61] X. Sun, J. Choi, C.-Y. Chen, N. Wang, S. Venkataramani, V. Srinivasan, X. Cui, W. Zhang, and K. Gopalakrishnan, "Hybrid 8-bit floating point (HFP8) training and inference for deep neural networks," in *NeurIPS*, 2019.

[62] A. T. Elthakeb, P. Pilligundla, F. Mireshghallah, A. Yazdanbakhsh, and H. Esmaeilzadeh, "ReLeQ: A reinforcement learning approach for deep quantization of neural networks," *arXiv preprint arXiv:1811.01704*, 2018.

[63] Y. Choukroun, E. Kravchik, F. Yang, and P. Kisilev, "Low-bit quantization of neural networks for efficient inference." in *ICCV Workshops*, 2019, pp. 3009–3018.

[64] S. Gupta, S. Ullah, K. Ahuja, A. Tiwari, and A. Kumar, "ALigN: A highly accurate adaptive layerwise quantization of pre-trained neural networks," *IEEE Access*, vol. 8, pp. 118 899–118 911, 2020.

[65] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," *Technical report, University of Toronto*, 2009.

[66] S. Han, J. Pool, J. Tran, and W. Dally, "Learning both weights and connections for efficient neural network," in *Advances in Neural Information Processing Systems*, 2015, pp. 1135–1143.

[67] J. Wu, Y. Wang, Z. Wu, Z. Wang, A. Veeraraghavan, and Y. Lin, "Deep k-means: Re-training and parameter sharing with harder cluster assignments for compressing deep convolutions," in *International Conference on Machine Learning*, 2018, pp. 5359–5368.

[68] T. Elsken, J. H. Metzen, F. Hutter et al., "Neural architecture search: A survey." *J. Mach. Learn. Res.*, vol. 20, no. 55, pp. 1–21, 2019.

[69] Z. Guo, X. Zhang, H. Mu, W. Heng, Z. Liu, Y. Wei, and J. Sun, "Single path one-shot neural architecture search with uniform sampling," in *European Conference on Computer Vision*. Springer, 2020, pp. 544–560.

[70] R. Zhao, Y. Hu, J. Dotzel, C. De Sa, and Z. Zhang, "Improving neural network quantization without retraining using outlier channel splitting," in *International Conference on Machine Learning*. PMLR, 2019, pp. 7543–7552.

[71] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding," *arXiv preprint arXiv:1510.00149*, 2015.

[72] S. Gupta, A. Agrawal, K. Gopalakrishnan, and P. Narayanan, "Deep learning with limited numerical precision," in *Proceedings of The 32nd International Conference on Machine Learning*, 2015, pp. 1737–1746.

[73] S. Wu, G. Li, F. Chen, and L. Shi, "Training and inference with integers in deep neural networks," *arXiv preprint arXiv:1802.04680*, 2018.

[74] N. Wang, J. Choi, D. Brand, C.-Y. Chen, and K. Gopalakrishnan, "Training deep neural networks with 8-bit floating point numbers," in *Advances in Neural Information Processing Systems*, 2018.

[75] M. Kim and P. Smaragdis, "Bitwise neural networks," *arXiv preprint arXiv:1601.06071*, 2016.

[76] M. Courbariaux et al., "Binaryconnect: Training deep neural networks with binary weights during propagations," in *Advances in Neural Information Processing Systems*, 2015, pp. 3123–3131.

[77] K. Hwang and W. Sung, "Fixed-point feedforward deep neural network design using weights+ 1, 0, and- 1," in *Signal Processing Systems (SiPS), 2014 IEEE Workshop on*. IEEE, 2014, pp. 1–6.

[78] S. Han, J. Pool, J. Tran, and W. Dally, "Learning both weights and connections for efficient neural network," in *Advances in Neural Information Processing Systems (NIPS)*, 2015, pp. 1135–1143.

[79] X. Zhang, J. Zou, K. He, and J. Sun, "Accelerating very deep convolutional networks for classification and detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2015.

[80] C. Sakr et al., "Minimum precision requirements for the SVM-SGD learning algorithm," in *Acoustics, Speech and Signal Processing (ICASSP), 2017 IEEE International Conference on*. IEEE, 2017.

[81] M. Goel and N. Shanbhag, "Finite-precision analysis of the pipelined strength-reduced adaptive filter," *Signal Processing, IEEE Transactions on*, vol. 46, no. 6, pp. 1763–1769, 1998.

[82] C. Caraiscos and B. Liu, "A roundoff error analysis of the LMS adaptive algorithm," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 32, no. 1, pp. 34–41, 1984.

[83] K. Knauer, "Ripple-carry adder," June 13 1989, US Patent 4,839,849.

[84] C. R. Baugh and B. A. Wooley, "A two's complement parallel array multiplication algorithm," *IEEE Transactions on Computers*, vol. 100, no. 12, pp. 1045–1047, 1973.

[85] Y. Lin et al., "Variation-tolerant architectures for convolutional neural networks in the near threshold voltage regime," in *Signal Processing Systems (SiPS), 2016 IEEE International Workshop on*. IEEE, 2016, pp. 17–22.

[86] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio, "Quantized neural networks: Training neural networks with low precision weights and activations," *arXiv preprint arXiv:1609.07061*, 2016.

[87] R. E. Blahut, *Fast Algorithms for Signal Processing*. Cambridge University Press, 2010.

[88] Y. LeCun, C. Cortes, and C. J. Burges, "The MNIST database of handwritten digits," 1998.

[89] C. Sakr et al., "Analytical guarantees on numerical precision of deep neural networks," in *Proceedings of the 34th International Conference on Machine Learning*, 2017, pp. 3007–3016.

[90] L. Bottou, "Large-scale machine learning with stochastic gradient descent," in *Proceedings of COMPSTAT'2010*. Springer, 2010, pp. 177–186.

[91] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[92] M. Raghu et al., "On the expressive power of deep neural networks," in *Proceedings of the 34th International Conference on Machine Learning*, 2017, pp. 2847–2854.

[93] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in PyTorch," in *NeurIPS Workshop on Automatic Differentiation*, 2017.

[94] M. Kang et al., "An energy-efficient VLSI architecture for pattern recognition via deep embedding of computation in SRAM," in *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2014, pp. 8326–8330.

[95] N. Shanbhag and Kothers, "Compute memory," US Patent 9,697,877, Issued July 4th 2017.

[96] M. Kang et al., "Deep in-memory architectures for machine learning - accuracy vs. efficiency trade-offs," *IEEE Transactions on Circuits and Systems - Part I*, vol. 67, no. 5, pp. 1627–1639, January 2020.

[97] S. K. Gonugondla et al., "Fundamental limits on energy-delay-accuracy of in-memory architectures in inference applications," *arXiv preprint arXiv:2012.13645*, 2020.

[98] B. Murmann, "Mixed-signal computing for deep neural network inference," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, pp. 1–11, 2020.

[99] M. Kang et al., "A multi-functional in-memory inference processor using a standard 6T SRAM array," *IEEE Journal of Solid-State Circuits*, vol. 53, no. 2, pp. 642–655, 2018.

[100] A. Rekhi et al., "Analog/mixed-signal hardware error modeling for deep learning inference," in *Proceedings of the 56th Annual Design Automation Conference 2019*, 2019, pp. 1–6.

[101] S. Lloyd, "Least squares quantization in PCM," *IEEE Transactions on Information Theory*, vol. 28, no. 2, pp. 129–137, 1982.

[102] B. Murmann, "ADC performance survey 1997-2019," 2019. [Online]. Available: https://web.stanford.edu/ murmann/adcsurvey.html

[103] B. Murmann, "A/D converter trends: Power dissipation, scaling and digitally assisted architectures," in *2008 IEEE Custom Integrated Circuits Conference*. IEEE, 2008, pp. 105–112.

[104] B. Widrow and I. Kollár, "Quantization noise," *Cambridge University Press*, vol. 2, p. 5, 2008.

[105] K. Parhi, *VLSI Digital Signal Processing Systems: Design and Implementation*. John Wiley & Sons, 2007.

[106] P. Westerink et al., "Scalar quantization error analysis for image subband coding using QMFs," *IEEE Transactions on Signal Processing*, vol. 40, no. 2, pp. 421–428, 1992.

[107] K. R. Varshney and L. R. Varshney, "Quantization of prior probabilities for hypothesis testing," *IEEE Transactions on Signal Processing*, vol. 56, no. 10, pp. 4553–4562, 2008.

[108] J. B. Evans et al., "Analysis and implementation of variable step size adaptive algorithms," *IEEE Transactions on Signal Processing*, vol. 41, no. 8, pp. 2517–2535, 1993.

[109] S. K. Gonugondla et al., "Fundamental limits on the precision of in-memory architectures," in *Proceedings of the 39th International Conference on Computer-Aided Design*, 2020, pp. 1–9.

[110] O. Johnson, *Information Theory and the Central Limit Theorem*. World Scientific, 2004.

[111] R. Narasimha et al., "Ber-optimal analog-to-digital converters for communication links," *IEEE Transactions on Signal Processing*, vol. 60, no. 7, pp. 3683–3691, 2012.

[112] J. Mo and R. W. Heath, "Capacity analysis of one-bit quantized MIMO systems with transmitter channel state information," *IEEE Transactions on Signal Processing*, vol. 63, no. 20, pp. 5498–5512, 2015.

[113] S. Zhang and N. R. Shanbhag, "Embedded algorithmic noise-tolerance for signal processing and machine learning systems via data path decomposition," *IEEE Transactions on Signal Processing*, vol. 64, no. 13, pp. 3338–3350, 2016.

[114] H. Jia et al., "A microprocessor implemented in 65nm CMOS with configurable and bit-scalable accelerator for programmable in-memory computing," *arXiv preprint arXiv:1811.04047*, 2018.

[115] C. Sakr and N. Shanbhag, "An analytical method to determine minimum per-layer precision of deep neural networks," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018, pp. 1090–1094.

[116] L. Liu et al., "Capacity-achieving MIMO-NOMA: Iterative LMMSE detection," *IEEE Transactions on Signal Processing*, vol. 67, no. 7, pp. 1758–1773, 2019.

[117] T.-H. Chan et al., "A convex analysis framework for blind separation of non-negative sources," *IEEE Transactions on Signal Processing*, vol. 56, no. 10, pp. 5120–5134, 2008.

[118] C. Sakr and N. R. Shanbhag, "Per-tensor fixed-point quantization of the back-propagation algorithm," in *7th International Conference on Learning Representations, ICLR 2019*, 2019.

[119] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[120] T. J. Ypma, "Historical development of the Newton–Raphson method," *SIAM Review*, vol. 37, no. 4, pp. 531–551, 1995.

[121] W. Wen, C. Xu, F. Yan, C. Wu, Y. Wang, Y. Chen, and H. Li, "Terngrad: Ternary gradients to reduce communication in distributed deep learning," in *Advances in Neural Information Processing Systems*, 2017, pp. 1508–1518.

[122] D. Alistarh, D. Grubic, J. Li, R. Tomioka, and M. Vojnovic, "QSGD: Communication-efficient SGD via gradient quantization and encoding," in *Advances in Neural Information Processing Systems*, 2017, pp. 1707–1718.

[123] R. Pascanu, T. Mikolov, and Y. Bengio, "On the difficulty of training recurrent neural networks," in *International Conference on Machine Learning*, 2013, pp. 1310–1318.

[124] C. Sakr et al., "Accumulation bit-width scaling for ultra-low precision training of deep networks," in *7th International Conference on Learning Representations, ICLR 2019*, 2019.

[125] C. Dwork, V. Feldman, M. Hardt, T. Pitassi, O. Reingold, and A. Roth, "The reusable holdout: Preserving validity in adaptive data analysis," *Science*, vol. 349, no. 6248, pp. 636–638, 2015.

[126] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng, "Reading digits in natural images with unsupervised feature learning," in *NIPS Workshop on Deep Learning and Unsupervised Feature Learning*, vol. 2011, no. 2, 2011, p. 5.

[127] S. Zagoruyko and N. Komodakis, "Wide residual networks," *arXiv preprint arXiv:1605.07146*, 2016.

[128] D. Balduzzi, M. Frean, L. Leary, J. P. Lewis, K. W.-D. Ma, and B. McWilliams, "The shattered gradients problem: If resnets are the answer, then what is the question?" in *Proceedings of the 34th International Conference on Machine Learning*, 2017, pp. 342–350.

[129] A. G. Anderson and C. P. Berg, "The high-dimensional geometry of binary neural networks," *arXiv preprint arXiv:1705.07199*, 2017.

[130] I. S. Tyurin, "Refinement of the upper bounds of the constants in Lyapunov's theorem," *Russian Mathematical Surveys*, vol. 65, no. 3, pp. 586–588, 2010.

[131] P. Micikevicius, S. Narang, J. Alben, G. Diamos, E. Elsen, D. Garcia, B. Ginsburg, M. Houston, O. Kuchaev, G. Venkatesh et al., "Mixed precision training," *arXiv preprint arXiv:1710.03740*, 2017.

[132] T. G. Robertazzi and S. C. Schwartz, "Best "ordering" for floating-point addition," *ACM Transactions on Mathematical Software (TOMS)*, vol. 14, no. 1, pp. 101–110, 1988.

[133] N. J. Higham, "The accuracy of floating point summation," *SIAM Journal on Scientific Computing*, vol. 14, no. 4, pp. 783–799, 1993.

[134] A. M. Castaldo, R. C. Whaley, and A. T. Chronopoulos, "Reducing floating point error in dot product using the superblock family of algorithms," *SIAM Journal on Scientific Computing*, vol. 31, no. 2, pp. 1156–1174, 2008.

[135] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 1026–1034.

[136] J. L. McKinstry, S. K. Esser, R. Appuswamy, D. Bablani, J. V. Arthur, I. B. Yildiz, and D. S. Modha, "Discovering low-precision networks close to full-precision networks for efficient embedded inference," *arXiv preprint arXiv:1809.04191*, 2018.

[137] X. Gastaldi, "Shake-shake regularization," *arXiv preprint arXiv:1705.07485*, 2017.

[138] C. Zhu, S. Han, H. Mao, and W. J. Dally, "Trained ternary quantization," *CoRR*, vol. abs/1612.01064, 2016. [Online]. Available: http://arxiv.org/abs/1612.01064

[139] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," *arXiv preprint arXiv:1312.6199*, 2013.

[140] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," *arXiv preprint arXiv:1706.06083*, 2017.

[141] S.-M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard, "Deepfool: A simple and accurate method to fool deep neural networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 2574–2582.

[142] P. McDaniel, N. Papernot, and Z. B. Celik, "Machine learning in adversarial settings," *IEEE Security & Privacy*, vol. 14, no. 3, pp. 68–72, 2016.