UNIVERSITY OF CALIFORNIA SAN DIEGO

Algorithmic Techniques towards Efficient Quantization of Deep Neural Networks

A dissertation submitted in partial satisfaction of the
requirements for the degree
Doctor of Philosophy

in

Electrical Engineering (Applied Physics)

by

Ahmed Taha Elthakeb Naguib Youssef

Committee in charge:

Professor Hadi Esmaeilzadeh, Chair
Professor Shadi Dayeh, Co-Chair
Professor Young-Han Kim
Professor Truong Nguyen
Professor Steven Swanson

2020

The dissertation of Ahmed Taha Elthakeb Naguib Youssef is approved, and it is acceptable in quality and form for publication on microfilm and electronically:

_____

_____

_____

_____
Co-Chair

_____
Chair

University of California San Diego

2020

DEDICATION

To my beloved family.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

Foremost, *Praise be to Allah, Lord of the Worlds*. This degree has been accomplished thanks to many persons.

First, I would like to thank my advisor, Professor Hadi Esmaeilzadeh, for the continuous support of my Ph.D study and research, for his patience, and immense knowledge. Hadi has been an exceptional advisor and I have been fortunate to join his research lab, the Alternative Computing Technologies (ACT), and receive his guidance during my Ph.D. I am most grateful to his trust and belief in me at times which granted me the freedom and confidence to pursue new research ideas on my own along with his generous support. I also owe it to him for significantly improving my academic writing skills.

At the same time, I would like to express my sincere gratitude to my co-advisor, Professor Shadi Dayeh, for his unprecedented support. Shadi was the reason I jointed UC San Diego, and the first person I met in US. I jointed Shadi's lab, Integrated Electronics and Biointerfaces Laboratory (IEBL), in my first year. Shadi offered me invaluable advice and diligently guided me through challenging problems. I am forever grateful to him.

Besides my advisors, I would like to thank the rest of my thesis committee, Professor Young-Han Kim, Professor Truong Nguyen, Professor Steven Swanson, and Professor Charles Deledalle for their encouragement, insightful comments, and stimulating questions.

It also has been an honor to work with many great collaborators during my time at UC San Diego. In particular, I would like to thank Professor Charles Deledalle for his insightful comments and feedback about extending SinReQ project to WaveQ. I would like to thank Professor Tarek Elgindi for his insightful discussions and efforts in developing theoretical results to support the ideas in SinReQ project. I would like also to thank Professor Alex Cloninger for all his great efforts in developing theoretical framework for DCQ project which was a major reason for completing the study and getting accepted at ICML 2020.

I am grateful to my internship mentors for offering me amazing opportunities in their groups to work on exciting projects. At Apple, I had the privilege to be mentored by and closely work with Mihir Sabnis and Ahran Dunsmoor in the VLSI Timing / CAD team, where we built interesting tool from scratch. At ARM, I had the privilege to be mentored by and closely work with Naveen Suda and Danny Loe in the Machine Learning Technology Group, where I learned so much about the practical aspects of my research.

I would like to thank my friends and lab mates in the ACT Lab, Prannoy Pilligundla, Fatemehsadat (Niloofar), Soroush Ghodrati, Joon Kyung Kim, Byung Hoon Ahn, Sean Kinzer, Jongse Park, and Behnam Khaleghi. I want to extend special thanks to Prannoy for the stimulating discussions, the sleepless nights we were working together before deadlines, all his efforts, and for all the fun we have had. Also my friends at UCSD: Yun Goo Ro, Namseok Park, Atsunori Tanaka, Sang Heon Lee, Woojin Choi, Mehran Ganji, Renjie Chen, Ren Liu, Sriram Venkatesh, and Iftikhar Ahmad Niaz. Everyone of them helped me one way or another, and I thank them for all the fun we have had in the last few years which made it a lot easier for me.

Last but not least, I would like to extend my gratitude to the electrical department staff members: Teresa Chiu, and Mary Duarte, they both have been always extraordinarily caring and helping from my day one in UCSD, I really appreciate all their efforts.

Finally, I would like to express my gratitude to my mother for her endless support and keeping my morals high in moments where desperation seemed to be the only choice.

Chapter 2, in part, contains a re-organized reprint of the material as it appears in *the Journal of IEEE, MICRO*. Ahmed T. Elthakeb, Prannoy Pilligundla, A. Yazdanbakhsh, FatemehSadat Mireshghallah, H. Esmaeilzadeh, 2020. The dissertation author was the primary investigator and author of this paper.

Chapter 3, in part, contains a re-organized reprint of the material as it appears in *International Conference on Machine Learning*. Ahmed T. Elthakeb, Prannoy Pilligundla, Fatemeh Mireshghallah, Alexander Cloninger, Hadi Esmaeilzadeh, 2020. The dissertation author was the

primary investigator and author of this paper.

Chapter 4, in part, has been submitted for publication of the material as it appears in *International Conference on Learning Representations*. Ahmed T. Elthakeb, Prannoy Pilligundla, Fatemeh Mireshghallah, Tarek Elgindi, Charles-Alban Deledalle, Hadi Esmaeilzadeh, 2021; and in part, contains a re-organized reprint of the material as it appears in *ICML Workshop on Understanding and Improving Generalization in Deep Learning*. Ahmed T. Elthakeb, Prannoy Pilligundla, Fatemeh Mireshghallah, Hadi Esmaeilzadeh, 2019. The dissertation author was the primary investigator and author of both papers.

Chapter 5, in part, contains a re-organized reprint of the material as it appears in *MLArch-Sys Workshop, ISCA*. Ahmed T. Elthakeb, Hadi Esmaeilzadeh, 2020. The dissertation author was the primary investigator and author of this paper.

VITA

2012	B. S. in Electrical Engineering, Cairo University, Egypt
2015	M. S. in Electrical and Computer Engineering, The American University in Cairo, Egypt
2020	Ph. D. in Electrical Engineering (Applied Physics), University of California San Diego

PUBLICATIONS

Ahmed T. Elthakeb, Prannoy Pilligundla, Fatemeh Mireshghallah, Alexander Cloninger, Hadi Esmaeilzadeh, "Divide and Conquer:Leveraging Intermediate Feature Representations for Quantized Training of Neural Networks", *Proceedings of the 37th International Conference on Machine Learning (ICML)*, 2020.

Ahmed T. Elthakeb, Prannoy Pilligundla, Fatemeh Mireshghallah, Tarek Elgindi, Charles-Alban Deledalle, Hadi Esmaeilzadeh, "WaveQ: Gradient-Based Deep Quantization of Neural Networks through Sinusoidal Adaptive Regularization", *ICLR*, 2021 (under review).

Ahmed T. Elthakeb, Prannoy Pilligundla, A. Yazdanbakhsh, FatemehSadat Mireshghallah, H. Esmaeilzadeh, "ReLeQ: A Reinforcement Learning Approach for Automatic Deep Quantization of Neural Networks", *Journal IEEE Micro*, 2020.

Ahmed T. Elthakeb, Hadi Esmaeilzadeh, "ΣΔ-BNN: Sigma-Delta Approach for Deep Neural Networks Binarization", *MLArchSys Workshop, ISCA*, 2020.

Fatemehsadat Mireshghallah, Mohammadkazem Taram, Ali Jalali, **Ahmed T. Elthakeb**, Dean Tullsen, Hadi Esmaeilzadeh, "A Principled Approach To Learning Stochastic Representations For Privacy In Deep Neural Inference", *Participatory Approaches to Machine Learning Workshop, ICML*, 2020.

Ahmed T. Elthakeb, Prannoy Pilligundla, Hadi Esmaeilzadeh, "SinReQ: Generalized Sinusoidal Regularization for Automatic LowBitwidth Deep Quantized Training", *Understanding and Improving Generalization in Deep Learning Workshop, ICML*, 2019.

Mehran Ganji, **Ahmed T. Elthakeb**, Atsunori Tanaka, Vikash Gilja, Eric Halgren and Shadi A. Dayeh, "Scaling Effects on the Electrochemical Performance of PEDOT, Au, and Pt for Electrocorticography Recording", *Journal Advanced Functional Materials, 83*, 2017.

Ren Liu, Renjie Chen, **Ahmed T. Elthakeb**, Sang Heon Lee, Sandy Hinckley, Massoud L Khraiche, John Scott, Deborah Pre, Yoontae Hwang, Atsunori Tanaka, Yun Goo Ro, Albert K Matsushita, Xing Dai, Cesare Soci, Steven Biesmans, Anthony James, John Nogan, Katherine L Jungjohann, Douglas V Pete, Denise B Webb, Yimin Zou, Anne G Bang, Shadi A Dayeh, "High Density Individually Addressable Nanowire Arrays Record Intracellular Activity from Primary Rodent and Human Stem Cell Derived Neurons", *Nano Letters, 17, 2757-2764*, 2017.

Ilke Uguz, Mehran Ganji, Adel Hama, Atsunori Tanaka, Sahika Inal, **Ahmed T. Elthakeb**, Roisin M Owens, Pascale P Quilichini, Antoine Ghestem, Christophe Bernard, Shadi A Dayeh, George G Malliaras, "Autoclave Sterilization of PEDOT: PSS Electrophysiology Devices', *Journal Advanced Healthcare Materials, vol.5, no.24, pp.3094-3098*, 2016.

ABSTRACT OF THE DISSERTATION

Algorithmic Techniques towards Efficient Quantization of Deep Neural Networks

by

Ahmed Taha Elthakeb Naguib Youssef

Doctor of Philosophy in Electrical Engineering (Applied Physics)

University of California San Diego, 2020

Professor Hadi Esmaeilzadeh, Chair
Professor Shadi Dayeh, Co-Chair

With numerous breakthroughs over the past several years, deep learning (DL) techniques have transformed the world of artificial intelligence (AI). The abilities that were once considered unique and humane, are now characteristics of powerful machines. State-of-the art performance across various perceptual tasks from computer vision, speech recognition, game playing, and others, have been demonstrated. Now that we know it works, current research is more directed towards exploring: (a) **TinyAI**: how to make it more efficient (deployable in resource-constrained devices), through developing new optimization algorithms and tooling; (b) **AutoAI**: how to reduce human effort and speedup the development cycle of AI systems through automation. (c)

InterpretableAI: understand why it works, through detailed theoretical studies; (d) **AppliedAI**: how to combine all these efforts to move from "Narrow AI" (resolving specific task) into "General AI" (human semi-equivalent).

This dissertation primarily takes on the exploration of the first and second directions (autoAI & tinyAI). In particular, we make progress towards developing algorithms for more efficient and automated AI systems with particular focus on quantization methods.

(i) Discovering optimal quantization bitwidths. *Research question: What is the optimal bitwidth per layer for optimal quantization of a deep neural network?* Proposal: we developed a systematic approach to automate the process of discovering the optimal bitwidth for each layer of a deep neural network while complying to the constraint of maintaining the accuracy through an end-to-end deep Reinforcement Learning framework (ReLeQ).

(ii) Quantization-aware training. *Research question: Can we train a DNN in such a way that makes them inherently robust to quantization?* Proposal: we developed a novel quantization-friendly regularization technique based on sinusoidal function, called WaveQ. WaveQ exploits the periodicity, differentiability, and the local convexity profile in sinusoidal functions to automatically propel weights towards values that are inherently closer to quantization levels. Moreover, leveraging the fact that sinusoidal period is a continuous valued parameter, we utilized it as an ideal optimization objective and a proxy to minimize the actual quantization bitwidth, which avoids the issues of gradient-based optimization for discrete valued parameters.

(iii) Improved and accelerated finetuning methods. *Research question: Can we finetune a quantized DNN in an efficient way to better improve its final accuracy?* Proposal: we developed a novel finetuning algorithm for quantized DNNs. The proposed approach utilizes knowledge distillation through teacher-student paradigm in a novel setting that exploits the feature extraction capability of DNNs for higher-accuracy quantization. This divide and conquer strategy makes the training of each student section possible in isolation while all these independently trained sections are later stitched together to form the equivalent fully quantized network.

Chapter 1

Introduction

Deep Neural Networks (DNNs), in the last decade, have made waves across a variety of domains from game playing, voice assistants, self-driving cars, all the way to medical diagnostics [2, 13, 36, 38, 56, 57]. These strikes have been primarily enabled by continuous and consistent progress and innovations in many directions including computer vision, object detection, speech recognition, and natural language processing [2]. All these try to solve different aspects of perception. DNNs compute efficiency (and AI in general), however, have become a major constraint in unlocking further applications and capabilities, as these models require rather massive amounts of computation even for a single inquiry.

1.1 Challenge: AI and Compute

Recent analysis [openAI] has shown that the amount of compute used in the largest AI training runs has been increasing exponentially with a 3.4-month doubling time (by comparison, Moore's Law had a 2-year doubling period), Figure 1.1. Since 2012, this metric has grown by more than 300,000x (a 2-year doubling period would yield only a 7x increase). Improvements in compute have been a key component of AI progress, so as long as this trend continues, it's imperative to prepare for the implications of systems far outside today's capabilities.

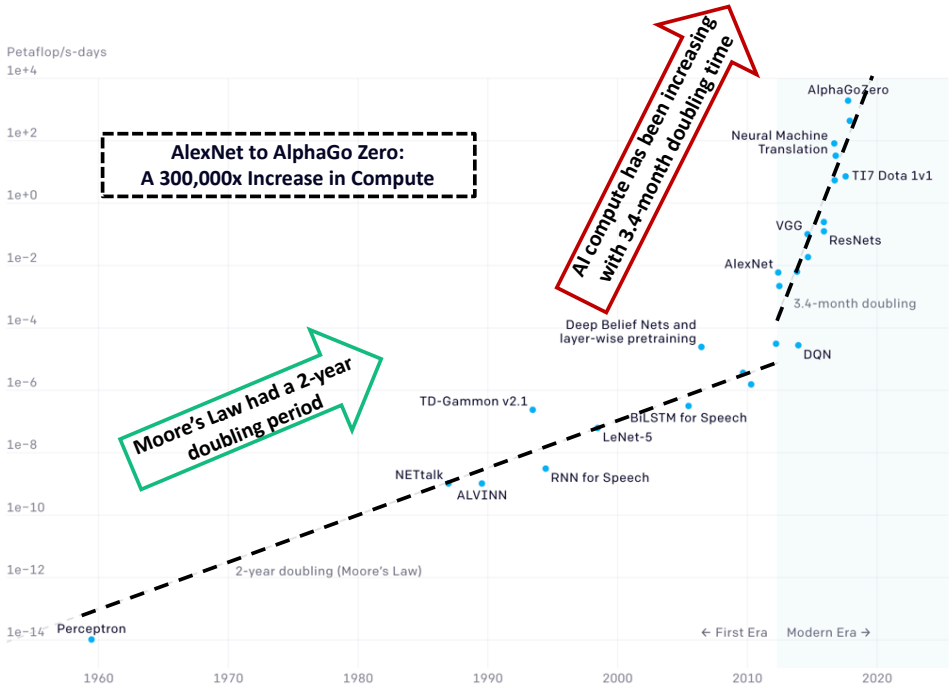

Figure 1.1: AI Compute Progression [source: OpenAI]

1.2 Solution: Algorithmic Innovations

Algorithmic innovation provide a path forward as a key factor for driving the advance of AI which might outpace gains from hardware efficiency. Algorithmic efficiency can be improved through several model optimization techniques, Figure 1.2.

One approach to reduce the intensity of the DNN computation is to reduce the complexity of each operation. To this end, quantization of neural networks reduces the bitwidth of the operations as well as the data footprint [43, 47, 83]. Albeit alluring, quantization can lead to significant accuracy loss if not employed with diligence. Years of research and development has yielded current levels of accuracy, which is the driving force behind the wide applicability of DNNs nowadays. To prudently preserve this valuable feature of DNNs, accuracy, while benefiting from quantization the following two fundamental problems need to be addressed. (1) learning techniques need to be developed that can train or tune quantized neural networks given a level of quantization for each layer. (2) Algorithms need to be designed that can discover the appropriate

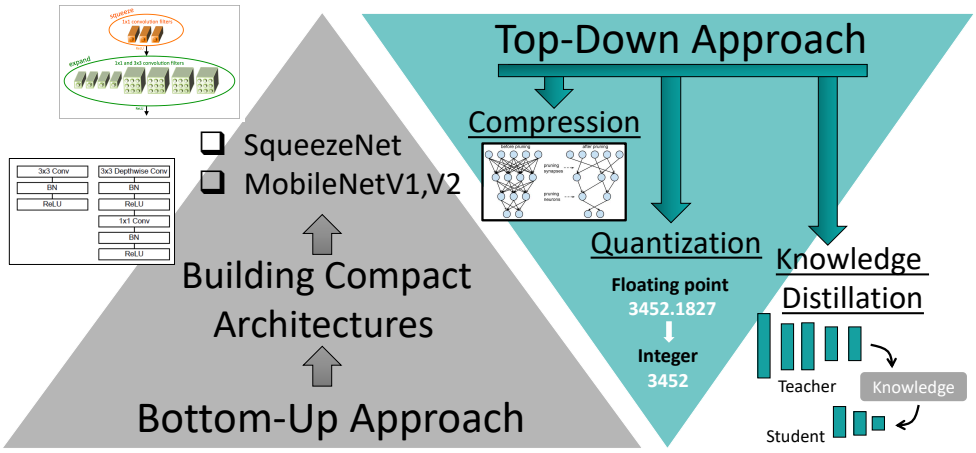

Figure 1.2: Neural Networks Optimization Approaches

level of quantization for each layer while considering the accuracy. This work takes on the second challenge as there are inspiring efforts that have developed techniques for quantized training.

1.3 Quantization of Neural Networks

Typically, neural networks are trained from scratch in full precision. Quantization, however, imposes a hard constraint on the parameters to assume discrete values. This hard constraint, in turn, introduces discontinuities which makes the objective function non-differentiable. As such, with the presence of quantization, backpropagation becomes infeasible since the gradient is zero almost everywhere. Figure 1.3 depicts an end-to-end quantized networks development pipeline showing different quantization opportunities. Depending on where quantization is applied (considered), these opportunities can be divided into three categories.

First, *quantization-aware training*, in this case quantization can be taken into consideration during training from scratch, where gradients are still in full precision. An example of this is quantization-aware regularization (or custom loss functions). This introduces a secondary objective (in addition to the primary task objective) to minimize quantization error. The resultant weights tend to get clustered around underlying quantization levels.

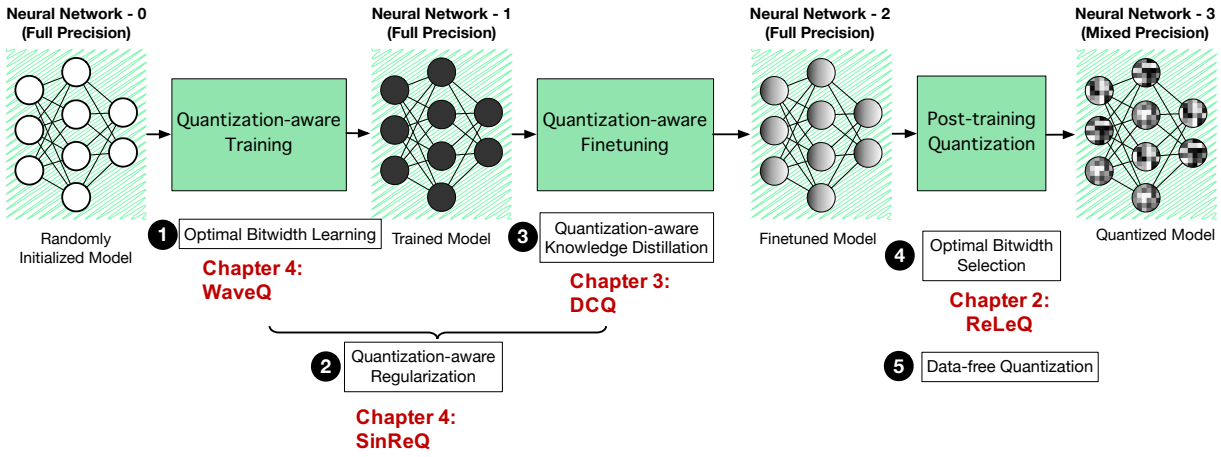

Figure 1.3: Thesis overview (Quantization opportunities)

Second, *quantization-aware finetuning*, in this case the assumption is there is a pre-trained model, then we apply quantization then finetuning to recover accuracy loss.

1.4 Thesis Outline and Contributions

This section provides an overview of this thesis. We propose multiple algorithmic techniques towards efficient deep quantization (below 8-bits) of neural networks (Figure 5.1).

Chapter 2: A Reinforcement Learning Approach for Deep Quantization of Neural Networks. Recent research affirms that carefully selecting the quantization levels for each layer can preserve the accuracy while pushing the bitwidth below eight bits. However, without arduous manual effort, this deep quantization can lead to significant accuracy loss, leaving it in a position of questionable utility. As such, deep quantization opens a large hyper-parameter space (bitwidth of the layers), the exploration of which is a major challenge. We propose a systematic approach to tackle this problem, by automating the process of discovering the quantization levels through an end-to-end deep reinforcement learning framework (ReLeQ). We adapt policy optimization methods to the problem of quantization, and focus on finding the best design decisions in choosing the state and action spaces, network architecture and training framework, as well as the tuning

of various hyperparamters. We show how ReLeQ can balance speed and quality, and provide an asymmetric general solution for quantization of a large variety of deep networks (AlexNet, CIFAR-10, LeNet, MobileNet-V1, ResNet-20, SVHN, and VGG-11) that virtually preserves the accuracy ($\leq$ 0.3% loss) while minimizing the computation and storage cost. With these DNNs, ReLeQ enables conventional hardware to achieve 2.2$\times$ speedup over 8-bit execution. Similarly, a custom DNN accelerator achieves 2.0$\times$ speedup and energy reduction compared to 8-bit runs. These encouraging results mark ReLeQ as the initial step towards automating the deep quantization of neural networks.

Chapter 3: Divide and Conquer for Quantization (quantization-aware knowledge distillation). The deep layers of modern neural networks extract a rather rich set of features as an input propagates through the network. This project sets out to harvest these rich intermediate representations for quantization with minimal accuracy loss while significantly reducing the memory footprint and compute intensity of the DNN. This work utilizes knowledge distillation through teacher-student paradigm (Hinton et al., 2015) in a novel setting that exploits the feature extraction capability of DNNs for higher-accuracy quantization. As such, our algorithm logically divides a pre-trained full-precision DNN to multiple sections, each of which exposes intermediate features to train a team of students independently in the quantized domain. This divide and conquer strategy, in fact, makes the training of each student section possible in isolation while all these independently trained sections are later stitched together to form the equivalent fully quantized network. Our algorithm is a sectional approach towards knowledge distillation and is not treating the intermediate representation as a hint for pre-training before one knowledge distillation pass over the entire network (Romero et al., 2015). Experiments on various DNNs (AlexNet, LeNet, MobileNet, ResNet-18, ResNet-20, SVHN and VGG-11) show that, this approach—called DCQ (Divide and Conquer Quantization)—on average, improves the performance of a state-of-the-art quantized.

Chapter 4: Gradient-Based Deep Quantization of Neural Networks through Sinusoidal

Adaptive Regularization. We propose a novel sinusoidal regularization, called WaveQ, for deep quantized training. Leveraging the sinusoidal properties, we seek to learn multiple quantization parameterization in conjunction during gradient-based training process. Specifically, we learn (i) a per-layer quantization bitwidth along with (ii) a scale factor through learning the period of the sinusoidal function. At the same time, we exploit the periodicity, differentiability, and the local convexity profile in sinusoidal functions to automatically propel (iii) network weights towards values quantized at levels that are jointly determined. We show how WaveQ balance compute efficiency and accuracy, and provide a heterogeneous bitwidth assignment for quantization of a large variety of deep networks (AlexNet, CIFAR-10, MobileNet, ResNet-18, ResNet-20, SVHN, and VGG-11) that virtually preserves the accuracy. Furthermore, we carry out experimentation using fixed homogenous bitwidths with 3- to 5-bit assignment and show the versatility of WaveQ in enhancing quantized training algorithms (DoReFa and WRPN) with about 4.8% accuracy improvements on average, and then outperforming multiple state-of-the-art techniques.

Chapter 5: Food for Thought on Neural Networks Optimization. Lastly, in this chapter we shed the light on few more ideas, some of them are ongoing projects, others are just thoughts.

Chapter 2

Reinforcement Learning for Deep Quantization of DNNs

As Deep Neural Networks (DNNs) make their ways into different domains and application, their compute efficiency is becoming a first-order constraint. *Deep Quantization (below eight bits)* can significantly reduce DNN computation and storage by decreasing the bitwidth of network encodings. Recent research affirms that carefully selecting the quantization levels for each layer can preserve the accuracy while pushing the bitwidth below eight bits. However, without arduous manual effort, this deep quantization can lead to significant accuracy loss, leaving it in a position of questionable utility. As such, deep quantization opens a large hyper-parameter space (bitwidth of the layers), the exploration of which is a major challenge. We propose a systematic approach to tackle this problem, by automating the process of discovering the quantization levels through an end-to-end deep reinforcement learning framework (ReLeQ). ReLeQ defines a new performance-driven point in the emerging area of *Automated Machine Learning (AutoML)*, which aims to automatically discover the best hyperparameters for the DNN model under deployment. This framework utilizes the sample efficiency of Proximal Policy Optimization (PPO) to explore the exponentially large space of possible assignment of the quantization-levels to the layers. We show

how ReLeQ can balance speed and quality, and provide a heterogeneous bitwidth assignment for quantization of a large variety of deep networks (AlexNet, CIFAR-10, LeNet, MobileNet-V1, ResNet-20, SVHN, and VGG-11) that virtually preserves the accuracy ($\leq 0.3\%$ loss) while minimizing the computation and storage cost. With these DNNs, ReLeQ enables conventional hardware to achieve $2.2\times$ speedup over 8-bit execution. Similarly, a custom DNN accelerator achieves $2.0\times$ speedup and energy reduction compared to 8-bit runs. These encouraging results mark ReLeQ as the initial step towards automating the deep quantization of neural networks.

2.1 Introduction

Deep Neural Networks (DNNs) have made waves across a variety of domains, from image recognition [52] and synthesis, object detection [77], natural language processing [19], medical imaging, self-driving cars, video surveillance, and personal assistance [36, 56]. DNN compute efficiency has become a major constraint in unlocking further applications and capabilities, as these models require rather massive amounts of computation even for a single inquiry. One approach to reduce the intensity of the DNN computation is to reduce the complexity of each operation. To this end, quantization of neural networks provides a path forward as it reduces the bitwidth of the operations as well as the data footprint [43, 47, 83]. Albeit alluring, quantization can lead to significant accuracy loss if not employed with diligence. Years of research and development has yielded current levels of accuracy, which is the driving force behind the wide applicability of DNNs nowadays. To prudently preserve this valuable feature of DNNs, accuracy, while benefiting from quantization the following two fundamental problems need to be addressed. (1) learning techniques need to be developed that can train or tune quantized neural networks given a level of quantization for each layer. (2) Algorithms need to be designed that can discover the appropriate level of quantization for each layer while considering the accuracy. This work takes on the second challenge as there are inspiring efforts that have developed techniques for

quantized training [65, 101, 102].

This work builds on the algorithmic insight that the bitwidth of operations in DNNs can be reduced *below eight bits* without compromising their classification accuracy. However, this possibility is manually laborious [62, 64, 93] as to preserve accuracy, the bitwidth varies across individual layers and different DNNs [58, 65, 101, 102]. Each layer has a different role and unique properties in terms of weight distribution. Thus, intuitively, different layers display different sensitivity towards quantization. Over-quantizing a more sensitive layer can result in stringent restrictions on subsequent layers to compensate and maintain accuracy. Nonetheless, considering layer-wise quantization opens a rather exponentially large hyper-parameter space, specially when quantization below eight bits is considered. For example, ResNet-20 exposes a hyper-parameter space of size $8^l = 8^{20} > 10^{18}$, where $l = 20$ is the number of layers and 8 is the possible quantization levels. This exponentially large hyper-parameter space grows with the number of the layers making it impractical to exhaustively assess and determine the quantization level for each layer.

To that end, this work sets out to propose a toolset in the emerging area of Automated Machine Learning (AutoML) [45].

This emerging area of AutoML aims to remove humans from the loop of designing and applying machine learning techniques, from the selection of DNN architecture [21], to determining algorithm-specific hyper-parameter settings [28]. Despite these effort, automatic assignment of bits to the DNN layers for quantization is still an open challenge. As such, we develop an end-to-end framework, dubbed ReLeQ, that exploits the sample efficiency of the Proximal Policy Optimization [82] to explore the quantization hyper-parameter space. The RL agent starts from a full-precision previously trained model and learns the sensitivity of final classification accuracy with respect to the quantization level of each layer, determining its bitwidth while keeping classification accuracy virtually intact. Observing that the quantization bitwidth for a given layer affects the accuracy of subsequent layers, our framework implements

an LSTM-based RL framework which enables selecting quantization levels with the context of previous layers' bitwidths. Rigorous evaluations with a variety of networks (AlexNet, CIFAR, LeNet, SVHN, VGG-11, ResNet-20, and MobileNet) shows that ReLeQ can effectively find heterogenous deep quantization levels that virtually preserve the accuracy ($\leq$0.3% loss) while minimizing the computation and storage cost. The results (Table 2.2) show that there is a high variance in quantization levels across the layers of these networks. For instance, ReLeQ finds quantization levels that average to 6.43 bits for MobileNet, and to 2.81 bits for ResNet-20. With the seven benchmark DNNs, ReLeQ enables conventional hardware [14] to achieve 2.2$\times$ speedup over 8-bit execution. Similarly, a custom DNN accelerator [48] achieves 2.0$\times$ speedup and 2.7$\times$ energy reduction compared to 8-bit runs. These results suggest that ReLeQ takes an effective first step towards automating the deep quantization of neural networks.

2.2 RL for Deep Quantization of DNNs

2.2.1 Need for Heterogeneity

Deep neural networks, by construction, and the underlying training algorithms cast different properties on different layers as they learn different levels of features representations. First, it is widely known that neural networks are heavily overparameterized [5]; thus, different layers exhibit different levels of redundancy. Second, for a given initialization and upon training, each layer exhibits a distribution of weights (typically bell-shaped) each of which has a different dynamic range leading to different degrees of robustness to quantization error, hence, different/heterogenous precision requirements. Third, our experiments (see Figure 2.6), where the design space of deep quantization is enumerated for small and moderate size networks, empirically shows that indeed Pareto optimal frontier is mostly composed of heterogenous bitwidths assignment. Furthermore, recent works empirically studied the layer-wise functional structure of overparameterized deep models and provided evidence for the heterogeneous characteristic of

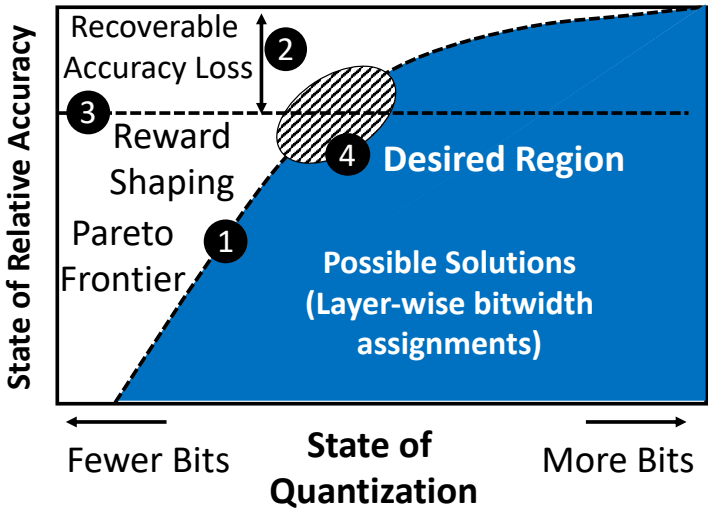

Figure 2.1: Sketch of the multi-objective optimization problem of layer-wise quantization of a neural network showing the underlying search space and the different design components.

layers. Recent experimental work [98] also shows that layers can be categorized as either "ambient" or "critical" towards post-training re-initialization and re-randomization. Another work [29] showed that a heterogeneously quantized versions of modern networks with the right mix of different bitwidths can match the accuracy of homogeneous versions with lower effective bitwidth on average. All aforementioned points poses a requirement for methods to efficiently discover heterogenous bitwidths assignment for neural networks. However, exploiting this possibility is manually laborious [62, 64, 93] as to preserve accuracy, the bitwidth varies across individual layers and different DNNs [58, 65, 101, 102]. Next subsection describes how to formulate this problem as a multi-objective optimization solved through Reinforcement Learning.

2.2.2 Multi-Objective Optimization

Figure 2.1 shows a sketch of the multi-objective optimization problem of layer-wise quantization of a neural network showing the underlying search space and the different design components. Given a particular network architecture, different patterns of layer-wise quantization bitwidths form a network specific design space (possible solutions). Pareto frontier (❶) defines

the optimum patterns of layer-wise quantization bitwidths (Pareto optimal solutions). However, not all Pareto optimal solutions are equally of interest as the accuracy objective has a priority over the quantization objective. In the context of neural network quantization, preferred/desired region on the Pareto frontier for a given neural network is motivated by how recoverable the accuracy is for a given state of quantization (i.e., a particular pattern of layer-wise bitwidths) of the network. Practically, the amount of recoverable accuracy loss (upon quantization) is determined by many factors: (1) the quantized training algorithm; (2) the amount of finetuning epochs; (3) the particular set of used hyperparameters; (4) the amenability of the network to recover accuracy, which is determined by the network architecture and how much overparameterization it exhibits. As such, a constraint (❷) is resulted below which even solutions on Pareto frontier are not interesting as they yield either unrecoverable or unaffordable accuracy loss depending on the application in hand. We formulate this as a Reinforcement Learning (RL) problem where, by tuning a parametric reward function (❸), the RL agent can be guided towards solutions around the desirable region (❹) that strike a particular balance between the "State of Relative Accuracy" (y-axis) and the "State of Quantization" (x-axis). As such, ReLeQ is an automated method for efficient exploration of large hyper-parameter space (heterogeneous bitwidths assignments) that is orthogonal to the underlying quantized training technique, quantization method, network architecture, and the particular hardware platform. Changing one or more of these components could yield different search spaces, hence, different results.

2.2.3 Method Overview

ReLeQ trains a reinforcement learning agent that determines the level of deep quantization (below 8 bits) for each layer of the network. ReLeQ agent explores the search space of the quantization levels (bitwidths), layer by layer. To account for the interplay between the layers with respect to quantization and accuracy, the state space designed for ReLeQ comprises of both static information about the layers and dynamic information regarding the network state during

the RL process (Section 2.2.4). In order to consider the effects of previous layers' quantization levels, the agent steps sequentially through the layers and chooses a bitwidth from a predefined set, e.g., $\{2,3,4,5,6,7,8\}$, one layer at a time (Section 2.2.5). The agent, consequently, receives a reward signal that is proportional to its accuracy after quantization and its benefits in terms of computation and memory cost. The underlying optimization problem is multi-objective (higher accuracy, lower compute, and reduced memory); however, preserving the accuracy is the primary concern. To this end, we shape the reward asymmetrically to incentivize accuracy over the quantization benefits (Section 2.2.6). With this formulation of the RL problem, ReLeQ employs the state-of-the-art Proximal Policy Optimization (PPO) [82] to train its policy and value networks. This section details the components and the research path we have examined to design them.

2.2.4 State Space Embedding to Consider Interplay between Layers

Table 2.1: Layer and network parameters for state embedding.

	Layer Specific	**Network Specific**
Static	Layer index Layer Dimensions Weight Statistics (standard deviation)	N/A
Dynamic	Quantization Level (Bitwidth)	State of Quantization State of Accuracy

Interplay between layers. The final accuracy of a DNN is the result of interplay between its layers and reducing the bitwidth of one layer can impact how much another layer can be quantized. Moreover, the sensitivity of accuracy varies across layers. We design the state space and the actions to consider these sensitivities and interplay by including the knowledge about the bitwidth of previous layers, the index of the layer-under-quantization, layer size, and statistics

(e.g., standard deviation) about the distribution of the weights. However, this information is incomplete without knowing the accuracy of the network given a set of quantization levels and state of quantization for the entire network. Table 2.1 shows the parameters used to embed the state space of ReLeQ agent, which are categorized across two different axes. (1) "Layer-Specific" parameters which are unique to the layer vs. "Network-Specific" parameters that characterize the entire network as the agent steps forward during training process. (2) "Static" parameters that do not change during the training process vs. "Dynamic" parameters that change during training depending on the actions taken by the agent while it explores the search space.

State of quantization and relative accuracy. The "Network-Specific" parameters reflect some indication of the state of quantization and relative accuracy. State of Quantization is a metric to evaluate the benefit of quantization for the network and it is calculated using the compute cost and memory cost of each layer. For a neural network with L layers, we define compute cost of layer l as the number of *M*ultiply-*Acc*umulate (*MAcc*) operations (n_l^{MAcc}), where ($l = 0, ..., L$). Additionally, since ReLeQ only quantizes weights, we define memory cost of layer l as the number of weights (n_l^w) scaled by the ratio of *Memory Access Energy* ($E_{MemoryAccess}$) to *MAcc Computation Energy* (E_{MAcc}), which is estimated to be around 120× [30]. It is intuitive to consider that the sum of memory and compute costs linearly scale with the number of bits for each layer (n_l^{bits}). The n_{max}^{bits} term is the maximum bitwidth among the predefined set of bitwidths that's available for the RL agent to pick from. Lastly, the State of Quantization ($State_{Quantization}$) is the normalized sum over all layers (L) that accounts for the total memory and compute cost of the entire network.

$$State_{Quantization} = \frac{\sum_{l=0}^{L}[(n_l^w \times \frac{E_{MemoryAccess}}{E_{MAcc}} + n_l^{MAcc}) \times n_l^{bits}]}{\sum_{l=0}^{L}[n_l^w \times \frac{E_{MemoryAccess}}{E_{MAcc}} + n_l^{MAcc}] \times n_{max}^{bits}} \tag{2.1}$$

Besides the potential benefits, captured by $State_{Quantization}$, ReLeQ considers the State of Relative Accuracy to gauge the effects of quantization on the classification performance. To that end, the State of Relative Accuracy ($State_{Accuracy}$) is defined as the ratio of the current accuracy (Acc_{Curr})

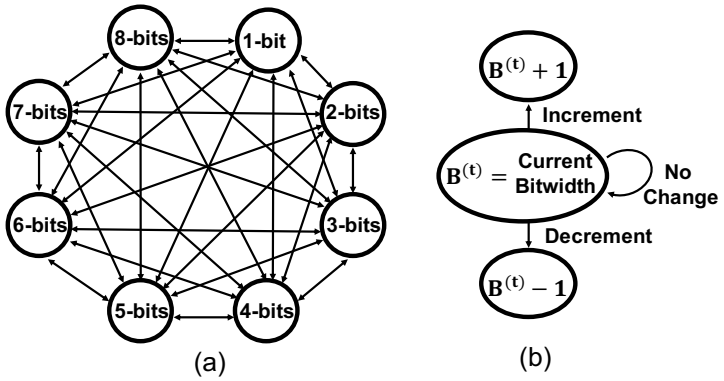

Figure 2.2: (a) Flexible action space (used in ReLeQ). (b) Alternative action space with restricted movement.

with the current bitwidths for all layers during RL training, to accuracy of the network when it runs with full precision (Acc_{FullP}). $State_{Accuracy}$ represents the degradation of accuracy as the result of quantization. The closer this term is to 1.0, the lower the accuracy loss and the more desirable/feasible the quantization.

$$State_{Accuracy} = \frac{Acc_{Curr}}{Acc_{FullP}} \tag{2.2}$$

Given these embedding of the observations from the environment, the ReLeQ agent can take actions, described next.

2.2.5 Flexible Actions Space

Intuitively, as calculations propagate through the layers, the effects of quantization will accumulate. As such, the ReLeQ agent steps through each layer sequentially and chooses from the bitwidth of a layer from a discrete set of quantization levels which are provided as possible choices. Figure 2.2(a) shows the representation of action space in which the set of bitwidths is $\{1,2,3,4,5,6,7,8\}$. As depicted, the agent can flexibly choose to change the quantization level of a given layer from any bitwidth to any other bitwidth. The set of possibilities can be changed as desired. Nonetheless, the action space depicted in Figure 2.2(a) is the possibilities considered for deep quantization in this work. As illustrated in Figure 2.2(b), an alternative

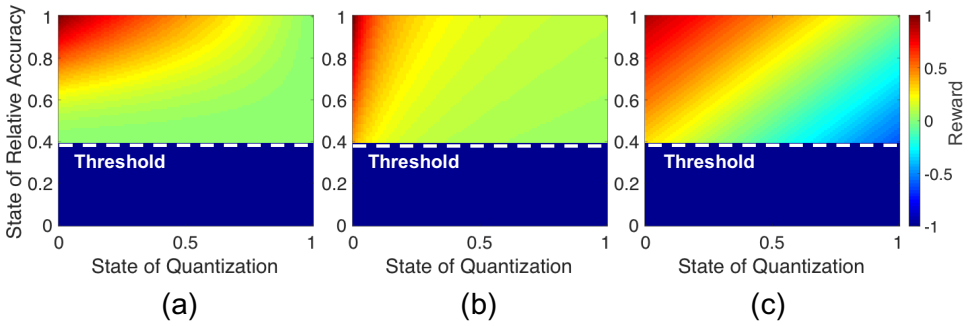

Figure 2.3: Reward shaping with three different formulations as functions of the optimization objectives: state of relative accuracy and state of quantization. (a) Proposed formulation, (b) direct division, and (c) direct subtraction. The color palette shows the intensity of the reward.

that we experimented with was to only allow the ReLeQ agent to increment/decrement/keep the current bitwidth of the layer ($B^{(t)}$). The experimentation showed that the convergence is much longer than the aforementioned flexible action space, which is used, as it encourages more exploration.

2.2.6 Asymmetric Reward Formulation for Accuracy

While the state space embedding focused on interplay between the layers and the action space provided flexibility, reward formulation for ReLeQ aims to preserve accuracy and minimize bitwidth of the layers simultaneously. This requirement creates an asymmetry between the accuracy and bitwidth reduction, which is a core objective of ReLeQ. The following Reward Shaping formulation provides the asymmetry and puts more emphasis on maintaining the accuracy as illustrated with different color intensities in Figure 2.3(a). This reward uses the same terms of State of Quantization ($State_{Quantization}$) and State of Relative Accuracy ($State_{Acc}$) from Section 2.2.4. One of the reasons that we chose this formulation is that it produces a smooth reward gradient as the agent approaches the optimum quantization combination. In addition, the varying 2-dimensional gradient speeds up the agent's convergence time. In the reward formulation, $a = 0.2$ and $b = 0.4$ can also be tuned and $th = 0.4$ is threshold for relative accuracy below which the

Reward Shaping:

$reward = 1.0 - (State_{Quantization})^a$

if $(State_{Acc} < th)$ **then**

  $reward = -1.0$

else

  $Acc_{discount} = max(State_{Acc}, th)^{(b/max(State_{Acc}, th))}$

  $reward = reward \times Acc_{discount}$

end if

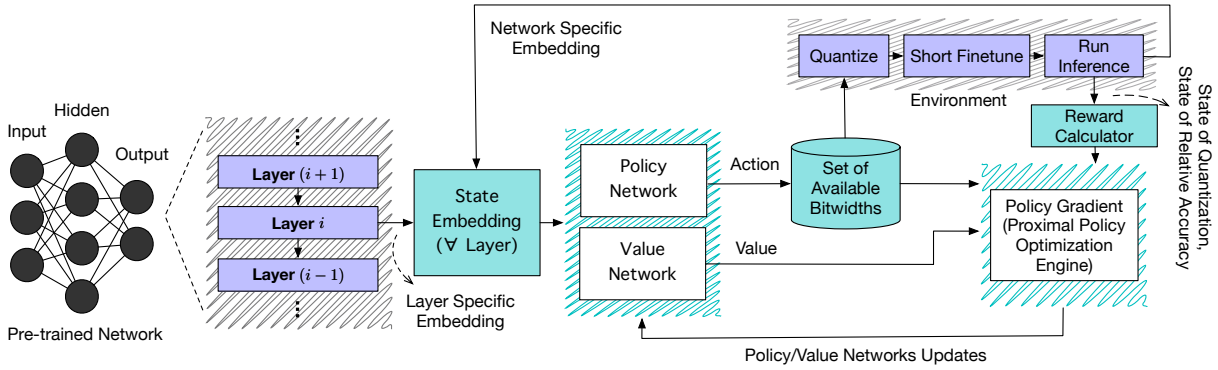

Figure 2.4: Overview of ReLeQ, which starts from a pre-trained network and delivers its corresponding deeply quantized network.

accuracy loss may not be recoverable and those quantization levels are completely unacceptable. The use of threshold also accelerates learning as it prevents unnecessary or undesirable exploration in the search space by penalizing the agent when it explores undesired low-accuracy states. While Figure 2.3(a) shows the aforementioned formulation, Figures 2.3(b) and (c) depict two other alternatives. Figure 2.3(b) is based on $State_{Acc}/State_{Quantization}$ while Figure 2.3(c) is based on $State_{Acc} - State_{Quantization}$. Section 2.5.6 provides detailed experimental results with these three reward formulations. The threshold also enables the agent to explore more relevant regions within the design space. In summary, the formulation for Figure 2.3(a) offers faster convergence.

2.2.7 Policy and Value Networks

While state, action and reward are three essential components of any RL problem, Policy and Value complete the puzzle and encode learning in terms of a RL context. While there are both

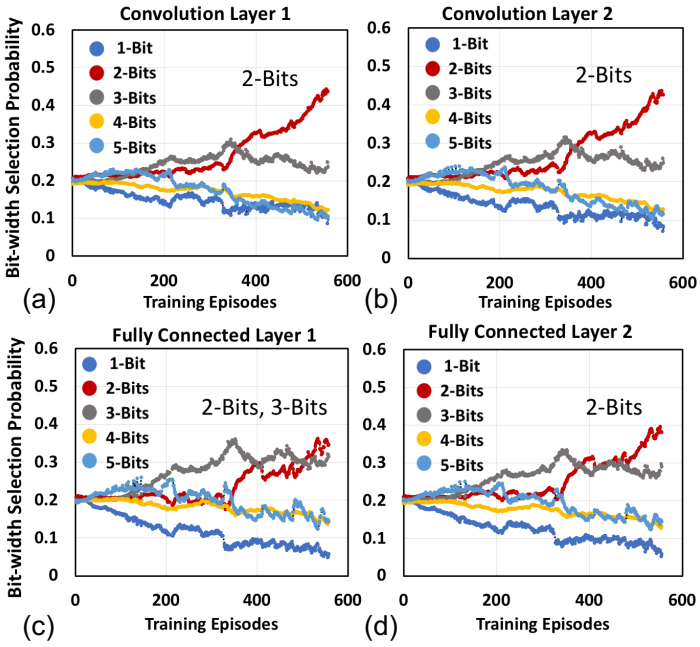

Figure 2.5: Action (Bitwidths selection) probability evolution over training episodes for LeNet.

policy-based and value-based learning techniques, ReLeQ uses a state-of-the-art policy gradient based approach, Proximal Policy Optimization (PPO) [82]. PPO is an actor-critic style algorithm so ReLeQ agent consists of both Policy and Value networks. Because all the layers in a network are linked with each other and a decision at the beginning of the network impacts other layers, we use a LSTM based neural network architecture.

2.2.8 Network Architecture of Policy and Value Networks

Both Policy and Value are functions of state, so the state space defined in Section 2.2.4 is encoded as a vector and fed as input to a Long short-term memory (LSTM) layer and this acts as the first hidden layer for both Policy and Value networks. Apart from the LSTM, policy network has two fully connected hidden layers of 128 neurons each and the number of neurons in the final output layer is equal to the number of available bitwidths the agent can choose from. Whereas the Value network has two fully connected hidden layers of 128 and 64 neurons each. Based on our

evaluations, LSTM enables the ReLeQ agent to converge almost $\times 1.33$ faster in comparison to a network with only fully connected layers.

While this section focused on describing the components of ReLeQ in isolation, the next section puts them together and shows how ReLeQ automatically quantizes a pre-trained DNN.

2.3 Putting it All Together: ReLeQ in Action

As discussed in Section 2.2, state, action and reward enable the ReLeQ agent to maneuver the search space with an objective of quantizing the neural network with minimal loss in accuracy. ReLeQ starts with a pre-trained model of full precision weights and proposes quantization levels of weights for all layers in a DNN. Figure 5.1 depicts the entire workflow for ReLeQ and this section gives an overview of how everything fits together.

2.3.1 Interacting with the Environment.

ReLeQ agent steps through all layers one by one, determining the quantization level for the layer at each step. For every step, the state embedding for the current layer comprising of different elements, described in Section 2.2.4, is fed as an input to the Policy and Value networks of the ReLeQ agent and the output is the probability distribution over the different possible bitwidths and value of the state respectively. ReLeQ agent then takes a stochastic action based on this probablity distribution and chooses a quantization level for the current layer. Weights for this particular layer are quantized to the predicted bitwidth and with accuracy preservation being a primary component of ReLeQ's reward function, retraining of a quantized neural network is required in order to properly evaluate the effectiveness of deep quantization. Such retraining is a time-intensive process and it undermines the search process efficiency. To get around this issue, we reward the agent with an estimated validation accuracy after retraining for a shortened amount of epochs. Dynamic network specific parameters, listed in Table 2.1, are updated based on the

validation accuracy and current quantization levels of the entire network before stepping on to the next layer. In this context, we define an epsiode as a single pass through the entire neural network and the end of every episode, we use Proximal Policy Optimization [82] to update the Policy and Value networks of the ReLeQ agent. After the learning process is complete and the agent has converged to a quantization level for each layer of the network, for example 2 bits for second layer, 3 bits for fourth layer and so on, we perform a long retraining step using the quantized bitwidths predicted by the agent and then obtain the final accuracy for the quantized version of the network.

2.3.2 Learning the Policy

Policy in terms of neural network quantization is to learn to choose the optimal bitwidth for each layer in the network. Since ReLeQ uses a Policy Gradient based approach, the objective is to optimize the policy directly, it's possible to visualize how policy for each layer evolves with respect to time (i.e., the number of episodes). Figure 2.5 shows the evolution of ReLeQ agent's bitwidth selection probabilities for all layers of LeNet over time (number of episodes), which reveals how the agent's policy changes with respect to selecting a bitwidth per layer. As indicated on the graph, the end results suggest the following quantization patterns, $2,2,2,2$ or $2,2,3,2$ bits. For the first two convolution layers (Convolution Layer 1, Convolution Layer 2), the agent ends up assigning the highest probability for two bits and its confidence increases with increasing number of training episodes. For the third layer (Fully Connected Layer 1), the probabilities of two bits and three bits are very close. Lastly, for the fourth layer (Fully Connected Layer 2), the agent again tends to select two bits, however, with relatively smaller confidence compared to layers one and two. With these observations, we can infer that bitwidth probability profiles are not uniform across all layers and that the agent distinguishes between the layers, understands the sensitivity of the objective function to the different layers and accordingly chooses the bitwidths. Looking at the agent's selection for the third layer (Fully Connected Layer 1) and recalling the

Table 2.2: Benchmark DNNs and their deep quantization with ReLeQ.

Network	Dataset	Quantization Bitwidths	Average Bitwidth	Accuracy Loss (%)
AlexNet	ImageNet	{8, 4, 4, 4, 4, 4, 4, 8}	5	**0.08**
SimpleNet	CIFAR10	{5, 5, 5, 5, 5}	5	**0.30**
LeNet	MNIST	{2, 2, 3, 2}	2.25	**0.00**
MobileNet	ImageNet	{8,5,6,6,4,4,7,8,4, 6,8,5,5,8,6,7,7,7, 6,8,6,8,8,6,7,5,5,7,8,8}	6.43	**0.26**
ResNet-20	CIFAR10	{8, 2, 2, 3, 2, 2, 2, 3, 2, 3, 3, 3, 2, 2, 2, 3, 2, 2, 2, 2, 2, 8}	2.81	**0.12**
10-Layers	SVHN	{8, 4, 4, 4, 4, 4, 4, 4, 4, 8}	4.80	**0.00**
VGG-11	CIFAR10	{8, 5, 8, 5, 6, 6, 6, 6, 8}	6.44	**0.17**
VGG-16	CIFAR10	{8, 8, 8, 6, 8, 6, 8, 6, 8, 6, 8, 6, 8, 8}	7.25	**0.10**

initial problem formulation of quantizing all layers while preserving the initial full precision accuracy, it is logical that the probabilities for two and three bits are very close. Going further down to two bits was beneficial in terms of quantization while staying at three bits was better for maintaining good accuracy which implies that third layer precision affects accuracy the most for this specific network architecture. This points out the importance of tailoring the reward function and the role it plays in controlling optimization tradeoffs.

2.4 Experimental Setup

2.4.1 Benchmarks

To assess the effectiveness of ReLeQ across a variety of DNNs, we use the following seven diverse networks that have been used in different real-world vision tasks: AlexNet, CIFAR-10

(Simplenet), LeNet, MobileNet (Version 1), ResNet-20, SVHN and VGG-11. Of these seven networks, AlexNet and MobileNet were evaluated on the ImageNet (ILSVRC'12) dataset, ResNet-20, VGG-11 and SimpleNet (5 layers) on CIFAR-10, SVHN (10 layers) on SVHN and LeNet on the MNIST dataset.

2.4.2 Quantization Technique

As described in earlier sections, ReLeQ is an off-the-shelf automated framework that works on top of any existing quantization technique to yield efficient heterogeneous bitwidths assignments. Here, we use the technique proposed in WRPN [65] where weights are first scaled and clipped to the $(-1.0, 1.0)$ range and quantized as per the following equation. The parameter k is the bitwidth used for quantization out of which $k-1$ bits are used for quantization and one bit is used for sign.

$$w_q = \frac{round((2^{k-1}-1)w_f)}{2^{k-1}-1} \tag{2.3}$$

Additionally, different quantization styles (e.g., mid-tread vs. mid-rise) yield different quantization levels. In mid-tread, zero is considered as a quantization level, while in mid-rise, quantization levels are shifted by half a step such that zero is not included as a quantization level. Here, we use mid-tread style following WRPN.

2.4.3 Granularity of Quantization

Quantization can come at different granularities: per-network, per-layer, per-channel/group, or per-parameter [50]. However, as the granularity becomes finer, the search space goes exponentially larger. Here, we consider per-layer granularity which strikes a balance between adapting for specific network requirements and practical implementations as supported in a wide range of hardware platforms such as CPUs, FPGAs, and dedicated accelerators. Nonetheless, similar principles of automated optimization can be extended for other granularities as needed.

2.4.4 Deep Quantization with Conventional Hardware

ReLeQ's solution can be deployed on conventional hardware, such as general purpose CPUs to provide benefits and improvements. To manifest this, we have evaluated ReLeQ using TVM [14] on an Intel Core i7-4790 CPU. We use TVM since its compiler supports deeply quantized operations with bit-serial vector operations on conventional hardware. We compare our solution in terms of the inference execution time (since the TVM framework does not offer energy measurements) against 8-bit quantized network. The results can be seen in Figure 2.8 and will be further elaborated in the next section.

2.4.5 Deep Quantization with Custom Hardware Accelerators

To further demonstrate the energy and performance benefits of the solution found by ReLeQ, we evaluate it on Stripes [48], a custom accelerator designed for DNNs, which exploits bit-serial computation to support flexible bitwidths for DNN operations. Stripes does not support or benefit from deep quantization of activations and it only leverages the quantization of weights. We compare our solution in terms of energy consumed and inference execution time against the 8-bit quantized network.

2.4.6 Comparison with Prior Work

We also compare against prior work [94], which proposes an iterative optimization procedure (dubbed ADMM) through which they find quantization bitwidths only for AlexNet and LeNet. Using Stripes [48] and TVM [14], we show that ReLeQ's solution provides higher performance and energy benefits compared to ADMM [94].

Table 2.3: Hyperparameters of PPO used in ReLeQ.

Hyperparameter	Value
Adam Step Size	1×10^{-4}
Generalized Advantage Estimation Parameter	0.99
Number of Epochs	3
Clipping Parameter	0.1

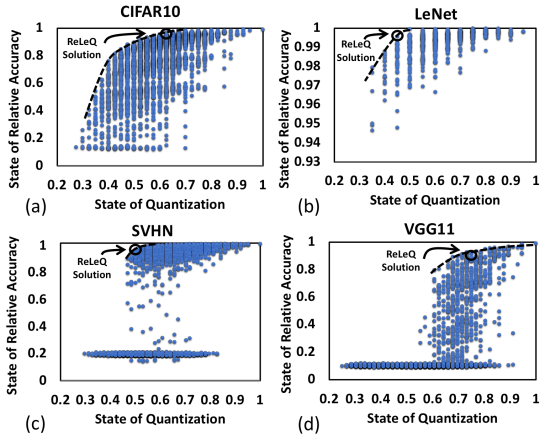

Figure 2.6: Quantization space and its Pareto frontier for (a) CIFAR-10, (b) LeNet, (c) SVHN, and (d) VGG-11.

2.4.7 Implementation and Hyper-parameters of the Proximal Policy Optimization (PPO)

As discussed, ReLeQ uses PPO [82] as its RL engine, which we implemented in python where its policy and value networks use TensorFlow's Adam Optimizer with an initial learning rate of 10^{-4}. The setting of the other hyper-parameters of PPO is listed in Table 2.3.

2.5 Experimental Results

2.5.1 Quantization Levels with ReLeQ

Table 2.2 provides a summary of the evaluated networks, datasets and shows the results with respect to layer-wise quantization levels (bitwidths) achieved by ReLeQ. Regarding the layer-wise quantization bitwidths, at the onset of the agent's exploration, all layers are initialized to 8-bits. As the agent learns the optimal policy, each layer converges with a high probability to a particular quantization bitwidth. As shown in the "Quantization Bitwidths" column of Table 2.2, ReLeQ quantization policies show a spectrum of varying bitwidth assignments to the layers. The bitwidth for MobileNet varies from 4 bits to 8 bits with an irregular pattern, which averages to 6.43. ResNet-20 achieves mostly 2 and 3 bits, again with an irregular interleaving that averages to 2.81. In many cases, there is significant heterogeneity and irregularity in the bitwidths and a uniform assignment of the bits is not always the desired choice to preserve accuracy. These results demonstrate that ReLeQ automatically distinguishes different layers and their varying importance with respect to accuracy while choosing their respective bitwidths. As shown in the "Accuracy Loss" column of Table 2.2, the deeply quantized networks with ReLeQ have less than 0.30% loss in classification accuracy. To assess the quality of these bitwidths assignments, we conduct a Pareto analysis on the DNNs for which we could populate the search space.

2.5.2 Validation: Pareto Analysis

Figure 2.6 depicts the solutions space for four benchmarks (CIFAR10, LeNet, SVHN, and VGG11). Each point on these charts is a unique combination of bitwidths that are assigned to the layers of the network. The boundary of the solutions denotes the Pareto frontier and is highlighted by a dashed line. The solution found by ReLeQ is marked out using an arrow and lays on the desired section of the Pareto frontier where the accuracy loss can be recovered through fine-tuning, which demonstrates the quality of the obtained solutions. It is worth noting that as

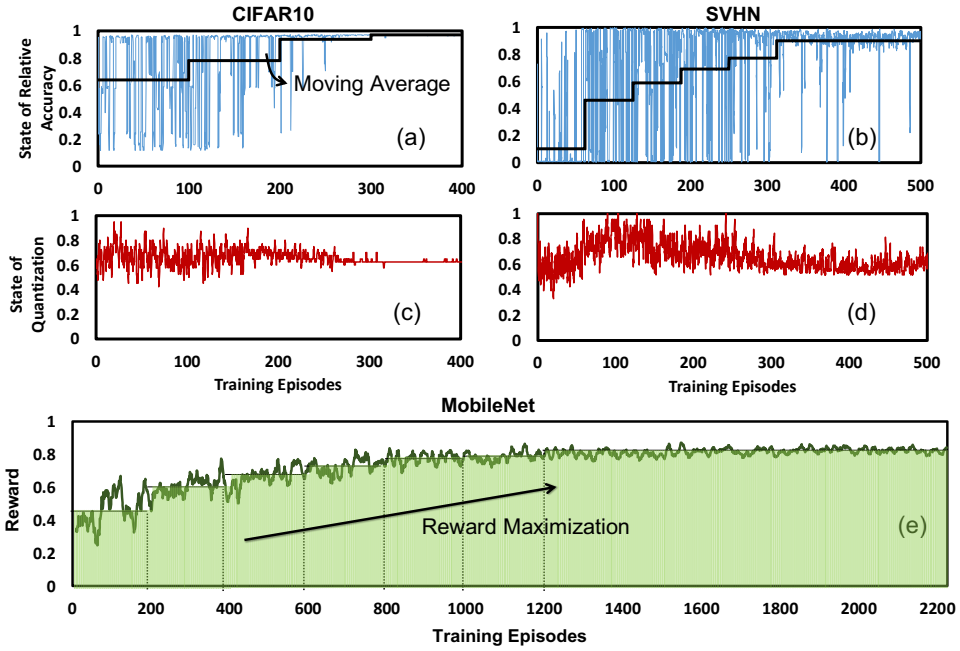

Figure 2.7: The evolution of reward and its basic elements: State of Relative Accuracy for (a) CIFAR-10, (b) SVHN. State of Quantization for (c) CIFAR-10, (d) SVHN, as the agent learns through the episodes. The last plot (e) shows an alternative view by depicting the evolution of reward for MobileNet. The trends are similar for the other networks.

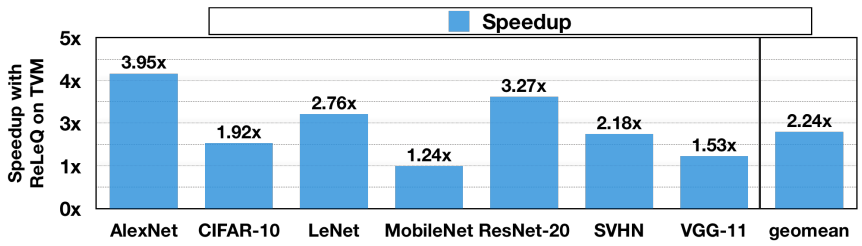

Figure 2.8: Speedup with ReLeQ for conventional hardware using TVM over the baseline run using 8 bits.

a result of the moderate size of the four networks presented in this subsection, it was possible to enumerate the design space, obtain Pareto frontier and assess ReLeQ quantization policy for each of the four networks. However, it is infeasible to do so for state-of-the-art deep networks (e.g., MobileNet and AlexNet) which further stresses the importance of automation and efficacy of ReLeQ.

2.5.3 Learning and Convergence Analysis

We further study the desired behavior of ReLeQ in the context of convergence. An appropriate evidence for the correctness of a formulated reinforcement learning problem is the ability of the agent to consistently yield improved solutions. The expectation is that the agent learns the correct underlying policy over the episodes and gradually transitions from the exploration to the exploitation phase. Figures 2.7(a) and (b) first show the State of Relative Accuracy for CIFAR10 and SVHN, respectively. We overlay the moving average of State of Relative Accuracy as episodes evolve, which is denoted by a black line in Figures 2.7(a) and (b). Similarly, Figures 2.7(c) and (d) depict the evolution of State of Quantization. As another indicative parameter of learning, Figure 2.7(e) plots the evolution of the reward, which combines the two States of Accuracy and Quantization (Section 2.2.6). As all the graphs show, the agent consistently yields solutions that increasingly preserve the accuracy (maximize rewards), while seeking to minimize the number of bits assigned to each layer (minimizing the state of quantization) and eventually converges to a rather stable solution. The trends are similar for the other networks.

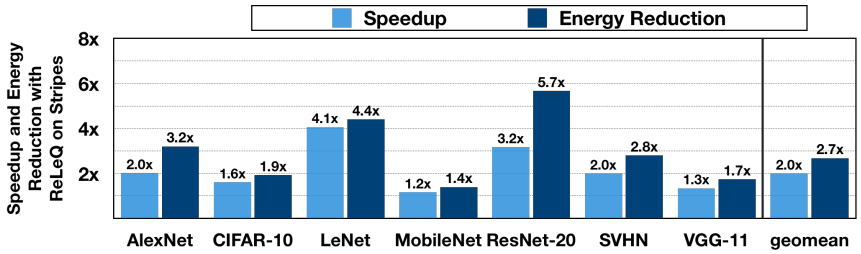

Figure 2.9: Energy reduction and speedup with ReLeQ for Stripes over the baseline execution when the accelerator is running 8-bit DNNs.

Table 2.4: Speedup and energy reduction with ReLeQ over ADMM [94].

Network	Dataset	Technique	Bitwidth	ReLeQ speedup on TVM	ReLeQ speedup on Stripes	Energy Improvement of ReLeQ on Stripes
AlexNet	ImageNet	**ReLeQ**	{8,4,4,4,4,4,4,8}	1.20*X*	1.22*X*	1.25*X*
		ADMM	{8,5,5,5,5,3,3,8}			
LeNet	MNIST	**ReLeQ**	{2,2,3,2}	1.42*X*	1.86*X*	1.87*X*
		ADMM	{5,3,2,3}			

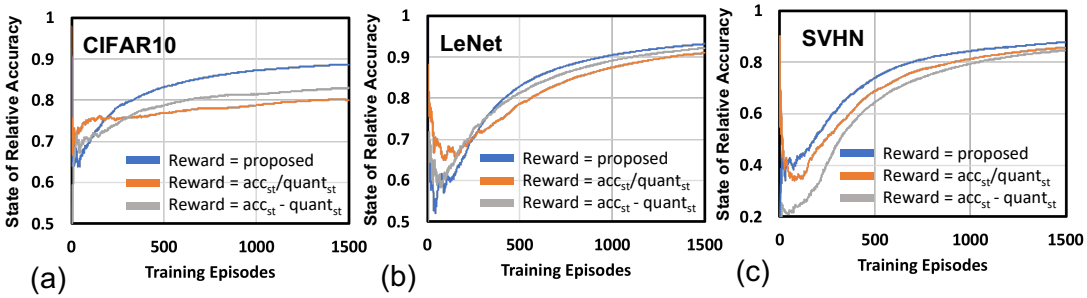

Figure 2.10: Three different reward functions and their impact on the state of relative accuracy over the training episodes for three different networks. (a) CIFAR-10, (b) LeNet, and (c) SVHN.

2.5.4 Execution Time and Energy Benefits with ReLeQ

Figure 2.8 shows the speedup for each benchmark network on conventional hardware using TVM compiler. The baseline is the 8-bit runtime for inference. ReLeQ's solution offers, on average, 2.2× speedup over the baseline as the result of merely quantizing the weights that reduces the amount of computation and data transfer during inference. Figure 2.9 shows the speedup and energy reduction benefits of ReLeQ's solution on Stripes custom accelerator. The baseline here is the time and energy consumption of 8-bit inference execution on the same accelerator.

ReLeQ's solutions yield, on average, 2.0× speedup and an additional 2.7× energy reduction. MobileNet achieves 1.2× speedup which is coupled with a similar degree of energy reduction. On the other end of the spectrum, ResNet-20 and LeNet achieve 3.0× and 4.0× benefits, respectively. As shown in Table 2.2, MobileNet needs to be quantized to higher bitwidths to maintain accuracy, compared with other networks and that is why the benefits are smaller.

2.5.5 Speedup and Energy Reduction over ADMM

As mentioned in Section 2.4, we compare ReLeQ's solution in terms of speedup and energy reduction against ADMM [94], another procedure for finding quantization bitwidths. As shown in Table 2.4, ReLeQ's solution provides 1.25× energy reduction and 1.22× average speedup over ADMM with Stripes for AlexNet and the benefits are higher for LeNet. The benefits are similar for the conventional hardware using TVM as shown in Table 2.4. ADMM does not report other networks.

2.5.6 Sensitivity Analysis: Influence of Reward Function

The design of reward function is a crucial component of reinforcement learning as indicated in Section 2.2.6. There are many possible reward functions one could define for a

particular application setting. However, different designs could lead to either different policies or different convergence behaviors. In this work, we incorporate reward engineering by proposing a special parametric reward formulation. To evaluate the effectiveness of the proposed reward formulation, we have compared three different reward formulations in Figure 2.10: (a) proposed in Section 2.2.6, (b) $R = State_{Accuracy}/State_{Quantization}$, (c) $R = State_{Accuracy} - State_{Quantization}$. As the blue line in all the charts shows, the proposed reward formulation consistently achieves higher State of Relative Accuracy during the training episodes. That is, our proposed reward formulation enables ReLeQ finds better solutions in shorter time.

2.5.7 Tuning: PPO Objective Clipping Parameter

One of the unique features about PPO algorithm is its novel objective function with clipped probability ratios, which forms a lower-bound estimate of the change in policy. Such modification controls the variance of the new policy from the old one, hence, improves the stability of the learning process. PPO uses a Clipped Surrogate Objective function, which uses the minimum of two probability ratios, one non-clipped and one clipped in a range between $[1-\varepsilon, 1+\varepsilon]$, where ε is a hyper-parameter that helps to define this clipping range. Table 2.5 provides a summary of tuning epsilon (commonly in the range of 0.1, 0.2, 0.3). Based on our performed experiments, $\varepsilon = 0.1$ often reports the highest average reward across different benchmarks.

2.6 Related Work

ReLeQ is the initial step in utilizing reinforcement learning to automatically find the level of quantization for the layers of DNNs such that their classification accuracy is preserved. As such, it relates to the techniques that given a level of quantization, train a neural network or develop binarized DNNs. Furthermore, the line of research that utilizes RL for hyperparameter discovery and tuning inspires ReLeQ. Nonetheless, ReLeQ, uniquely and exclusively, offers an

Table 2.5: Sensitivity of reward to different clipping parameters.

PPO Clipping Parameter	Average Normalized Reward (Performance)		
	LeNet on MNIST	SimpleNet on CIFAR-10	8-Layers on SVHN
$\varepsilon = 0.1$	**0.209**	0.407	**0.499**
$\varepsilon = 0.2$	0.165	**0.411**	0.477
$\varepsilon = 0.3$	0.160	0.399	0.455

RL-based approach to determine the levels of quantization.

Automated machine learning (AutoML) methods. Because of the increased deployment of deep learning models into various domains and applications, AutoML has recently gained a substantial interest from both academia [28, 61] and industry as internal tools [32], or open services [6, 60]. Hyperparameter optimization (HPO) is a major subfield of AutoML. One notable AutoML system is Auto-sklearn [28] that uses Bayesian optimization method to find the best instantiation of classifiers in scikit-learn [71]. Another major application of AutoML is neural architecture search (NAS) which also a has significant overlap with hyperparameter optimization [21]. Recently, Google introduced Cloud AutoML service [60] that is suite of machine learning products that enables developers with limited machine learning expertise to train high-quality models specific to their business needs. It relies on Google's state-of-the-art transfer learning and neural architecture search technology. Even though most of these frameworks include a wide range of supervised learning methods, a little include modern neural networks and their optimization.

Reinforcement learning for automatic tuning. RL based methods have attracted much attention within NAS after obtaining the competitive performance on the CIFAR-10 dataset employing RL as the search strategy despite the massive amount of the used computational resources [104]. Different RL approaches differ in how they represent the agent's policy. Zoph

and Le [104] use a recurrent neural network (RNN) trained by policy gradient, in particular, REINFORCE, to sequentially sample a string that in turn encodes a neural architecture. Baker et. al. [8] use Q-learning to train a policy which sequentially chooses a layer's type and corresponding hyperparameters.

Recently, a wide variety of methods have been proposed in quick succession to reduce the computational costs and achieve further performance improvements [72, 96].

Aside from NAS applications, i.e. engineering neural architecture from scratch, [37] employ RL to prune existing architectures where a policy gradient method is used to automatically find the compression ratio for different layers of a network. Here, we employ RL in the context of quantization to choose an appropriate quantization bitwidth for each layer of a network.

Training algorithms for quantized neural networks. There have been several techniques [65, 101, 102] that train a neural network in a quantized domain after the bitwidth of the layers is determined manually. DoReFa-Net [101] trains quantized convolutional neural networks with parameter gradients which are stochastically quantized to low bitwidth numbers before they are propagated to the convolution layers. [65] introduces a scheme to train networks from scratch using reduced-precision activations by decreasing the precision of both activations and weights and increasing the number of filter maps in a layer. [102] performs the training phase of the network in full precision, but for inference uses ternary weight assignments. For this assignment, the weights are quantized using two scaling factors which are learned during training phase. PACT [15] introduces a quantization scheme for activations, where the variable α is the clipping level and is determined through a gradient descent based method.

ReLeQ is an orthogonal technique with a different objective: automatically finding the level of quantization that preserves accuracy and can potentially use any of these training algorithms.

Ternary and binary neural networks. Extensive work, [43, 58, 75] focuses on binarized neural networks, which impose accuracy loss but reduce the bitwidth to lowest possible level. In

BinaryNet [42], an extreme case, a method is proposed for training binarized neural networks which reduce memory size, accesses and computation intensity at the cost of accuracy. XNOR-Net [75] leverages binary operations (such as XNOR) to approximate convolution in binarized neural networks. Another work [58] introduces ternary-weight networks, in which the weights are quantized to -1, 0, +1 values by minimizing the Euclidian distance between full-precision weights and their ternary assigned values. However, most of these methods rely on handcrafted optimization techniques and ad-hoc manipulation of the underlying network architecture that are not easily extendable for new networks. For example, multiplying the outputs with a scale factor to recover the dynamic range (i.e., the weights effectively become -w and w, where w is the average of the absolute values of the weights in the filter), keeping the first and last layers at 32-bit floating point precision, and performing normalization before convolution to reduce the dynamic range of the activations. Moreover, these methods [58, 75] are customized for a single bitwidth, binary only or ternary only in the case of [75] or [58], respectively, which imposes a blunt constraint on inherently different layers with different requirements resulting in sub-optimal quantization solutions. ReLeQ aims to utilize the levels between binary and 8 bits to avoid loss of accuracy while offering automation.

Techniques for selecting quantization levels. Recent work ADMM [94] runs a binary search to minimize the total square quantization error in order to decide the quantization levels for the layers. Then, they use an iterative optimization technique for fine-tuning. NVIDIA also released an automatic mixed precision (AMP) [68] which employs mixed precision during training by automatically selecting between two floating point (FP) representations (FP16 or FP32). There is a concurrent work HAQ [91] which also uses RL in the context of quantization. The following highlights some of the differences. ReLeQ uses a unique reward formulation and shaping that enables simultaneously optimizing for two objectives (accuracy and reduced computation with lower-bitwidth) within a unified RL process. In contrast, HAQ utilizes accuracy in the reward formulation and then adjusts the RL solution through an approach that sequentially

decreases the layer bitwidths to stay within a predefined resource budget. This approach also makes HAQ focused more towards a specific hardware platform whereas we are after a strategy than can generalize. Additionally, we also provide a systemic study of different design decisions, and have significant performance gain across diverse well known benchmarks. The initial version of our work [26], predates HAQ, and it is the first to use RL for quantization. Later HAQ was published in CVPR [90], and we published initial version of ReLeQ in NeurIPS ML for Systems Workshop [25].

2.7 Conclusion

Quantization of neural networks offers significant promise in reducing their compute and storage cost. However, the utility of quantization hinges upon automating its process while preserving accuracy. This work sets out to define the automated discovery of quantization levels for the layers while complying to the constraint of maintaining the accuracy. As such, this work offered the RL framework that was able to effectively navigate the huge search space of quantization and automatically quantize a variety of networks leading to significant performance and energy benefits. The results suggest that a diligent design of our RL framework, which considers multiple concurrent objectives can automatically yield high-accuracy, yet deeply quantized, networks.

Acknowledgment. Chapter 2, in part, contains a re-organized reprint of the material as it appears in *the Journal of IEEE, MICRO*. Ahmed T. Elthakeb, Prannoy Pilligundla, A. Yazdanbakhsh, FatemehSadat Mireshghallah, H. Esmaeilzadeh, 2020. The dissertation author was the primary investigator and author of this paper.

Chapter 3

Divide and Conquer: Leveraging Intermediate Feature Representations for Quantized Training of Neural Networks

The deep layers of modern neural networks extract a rather rich set of features as an input propagates through the network, this work sets out to harvest these rich intermediate representations for quantization with minimal accuracy loss while significantly reducing the memory footprint and compute intensity of the DNN. This work utilizes knowledge distillation through teacher-student paradigm [39] in a novel setting that exploits the feature extraction capability of DNNs for higher-accuracy quantization. As such, our algorithm logically divides a pretrained full-precision DNN to multiple sections, each of which exposes intermediate features to train a team of students independently in the quantized domain. This divide and conquer strategy, makes the training of each student section possible in isolation, which offers additional speedup through enabling parallelization, while all these independently trained sections are later stitched together to form the equivalent fully quantized network.

Experiments on various DNNs (AlexNet, LeNet, MobileNet, ResNet-18, ResNet-20,

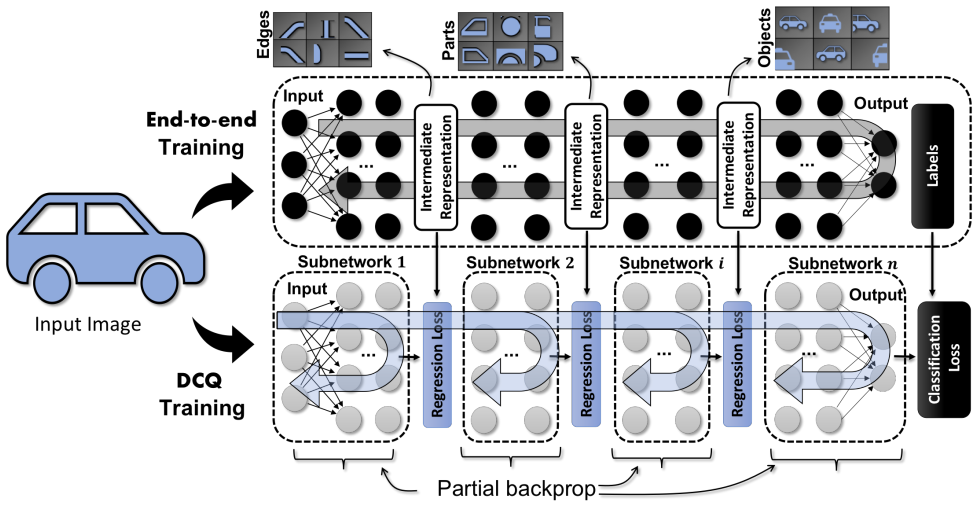

Figure 3.1: Overview of Divide and Conquer Quantization.

SVHN and VGG-11) show that, this approach—called DCQ (**D**ivide and **C**onquer **Q**uantization)—on average, improves the performance of a state-of-the-art quantized training technique, DoReFa-Net [101] by 21.6% and 9.3% for binary and ternary quantization, respectively. Additionally, we show that incorporating DCQ to existing quantized training methods leads to improved accuracies as compared to previously reported by multiple state-of-the-art quantized training methods.

3.1 Introduction

Today deep learning, with its superior performance, dominates a wide range of real life inference tasks including image recognition, voice assistants, and natural language processing [36, 53, 55, 56]. However, the shear complexity of deep learning models and the associated heavy compute and memory requirement appears as a major challenge as the demand for such services rapidly scale. Quantization, which can reduce the complexity of each operation as well as the overall storage requirements of the DNN, has proven to be a promising path forward. Nevertheless, quantization requires carefully tailored training and recovery algorithms [20, 35, 44, 100, 101] to even partially overcome its losses in accuracy. In this work, we set out to devise an algorithm

that enables quantization with much less accuracy degradation. The key insight is that the intermediate layers of a deep network already extract a very rich set of features and these intermediate representations can be used to train/teach a quantized network more effectively. To that end, we define a new approach towards knowledge distillation through teacher-student paradigm [12, 39] focusing on teaching the knowledge of intermediate features to a corresponding quantized student. Knowledge distillation [39] is a generic approach to reduce a large model down to a simpler or smaller distilled model. At a high level, a softened version of the final output is used to train a small model (student) to mimic the behavior of the original large model (teacher). FITNETS [78] extends this idea and takes hints from an intermediate layer of the teacher to pretrain the first few layers of the student and then apply knowledge distillation to the entire student network. We, on the other hand, tap into the multiple intermediate layers and apply knowledge distillation through sectioning. The sectioning enables DCQ to train each section of the students independently in isolation to deliver a quantized counterpart for the teacher, which enables parallelization. As such, DCQ offers additional speedup through parallelization on the algorithmic level and independent of improvements in hardware accelerators. In fact, the hints as proposed in FITNETS are complementary and can potentially be used in our sectional knowledge distillation. The proposed algorithm, DCQ, employs a divide and conquer approach that divides a pretrained full-precision network into multiple sections, each of which exposes a set of intermediate features. As Figure 5.1 illustrates, DCQ allocates a student section to each teacher counterpart and independently trains them using the intermediate feature representations. DCQ calculates the loss of each student section by comparing it with the output activations of the corresponding teacher section of the full precision network. Loss is optimized through a sectional multi-backpropagation scheme using conventional gradient-based training as shown in Figure 5.1. These trained student sections are then sewed back together to form the corresponding quantized DNN.

We validate our method through experiments on a variety of DNNs including AlexNet,

LeNet, MobileNet, ResNet-18, ResNet-20, SVHN and VGG-11 with binary and ternary weights. Results show that DCQ, on average, improves the performance of a state-of-the-art quantized training technique, DoReFa-Net [101] by 21.6% and 9.3% for binary and ternary quantization, respectively, which further helps in closing the accuracy gap between state-of-the-art quantized training techniques and the full-precision runs. Additionally, we show that our approach, DCQ, can improve performance of existing knowledge-distillation based approaches [65] and multiple state-of-the-art quantized training methods. These encouraging results suggest that leveraging the inherent feature extraction ability of DNNs for knowledge distillation can lead to significant improvement in their efficiency, reducing their bitwidth in this particular case.

The contributions of this proposal can be summarized as follows.

- **Extending knowledge distillation.** DCQ enables leveraging *arbitrary number of intermediate layers* relying on the inherent hierarchical learning characteristic of deep neural networks in contrast to only the output layer or hint layer. As such, distillation learning and hint learning fall as special cases of the proposed *divide and conquer strategy*.

- **Enabling parallelization towards training quantized networks.** DCQ applies knowledge distillation through sectioning. As such it trains each section of the students independently in isolation to deliver a quantized counterpart for the teacher, which can occur in parallel.

- **Complementary to other methods.** DCQ is a complementary method as it acts as an auxiliary approach to boost performance of existing training techniques by applying whatever the underlying training technique but in a stage-wise fashion with defining a regression loss per stage.

- **Theoretical analysis.** We provide a theoretical analysis/guarantee of the error upper bound across the network through a chaining argument.

3.2 DCQ: Divide and Conquer for Quantization

Overview. We take inspiration from knowledge distillation and apply it to the context of quantization by proposing a novel technique dubbed DCQ. The main intuition behind DCQ is that a deeply quantized network can achieve accuracies similar to full precision networks if intermediate layers of the quantized network can retain the intermediate feature representations that was learnt by the full precision network. To this end, DCQ splits the quantized network and full precision network into multiple small sections and trains each section individually by means of partial backpropagation so that every section of the quantized network learns and represents similar features as the corresponding section in the full precision network. In other words, DCQ divides the original classification problem into multiple regression problems by matching the intermediate feature (activation) maps. The following points summarizes the practical significance and contribution of DCQ.

Weight and activation quantization. The proposed technique is orthogonal to the quantity of interest for quantization, as it's basically applying whatever the underlying/used training technique but in a stage-wise fashion with defining a new regression loss per stage. In fact, the regression loss is defined to match the respective activation maps for each stage. As such, DCQ can be equally applied for weight and/or activation quantization alike. Section 3.3.2 presents results for both weight and activation quantization.

Integration to other methods. The proposed technique is a complementary method as it acts as an auxiliary approach to boost performance of existing training techniques by applying whatever the underlying/used training technique but in a stage-wise fashion with defining a new regression loss per stage.

Knowledge distillation utilization. DCQ extends the concept of knowledge distillation to its limits by leveraging multiple intermediate layers as opposed to limiting it to the output layer only as in [63], [39] or the output layer and hint layer as in [78].

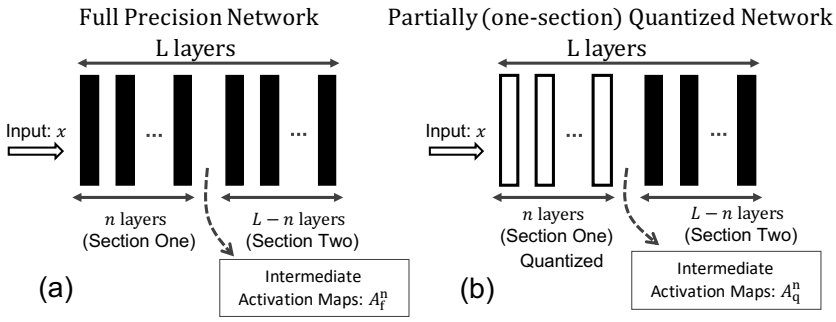

Figure 3.2: DCQ two stage split example

Other performance benefits. DCQ enables per-network training "parallelization" by enabling training different sections/stages in isolation (stage-wise fashion). Moreover, it applies the standard back propagation in a simpler settings (small subnetworks) which enables both faster convergence time and higher accuracy than existing conventional fine-tuning methods in the quantized domain.

This section describes different steps and rationale of our technique in more detail.

3.2.1 Matching Activations for Intermediate Layers

Figure 3.2 (a) shows a sketch representing a full precision network of L layers, whereas Figure 3.2 (b) is a deeply quantized version of the same network where first n layers are quantized and the remaining $L-n$ layers are at full precision. When we pass the same input image x to both these networks, if the output activations of layer n for full precision network, i.e., A_f^n, are equivalent to the output activations of layer n for the semi-quantized network, A_q^n, then both the networks classify the input to a same class because rest of the $L-n$ layers are same for both the networks and their input activations are same as well. Therefore, if both these networks shown in Figure 3.2 (a) and (b), have similar output activations for all the input images, then the network with first n layers quantized has learnt to represent similar features as the first n layers of the original network and it will have the same classification accuracy as the full precision network. We can extend this argument further and say that if we quantize the remaining $L-n$ layers of the

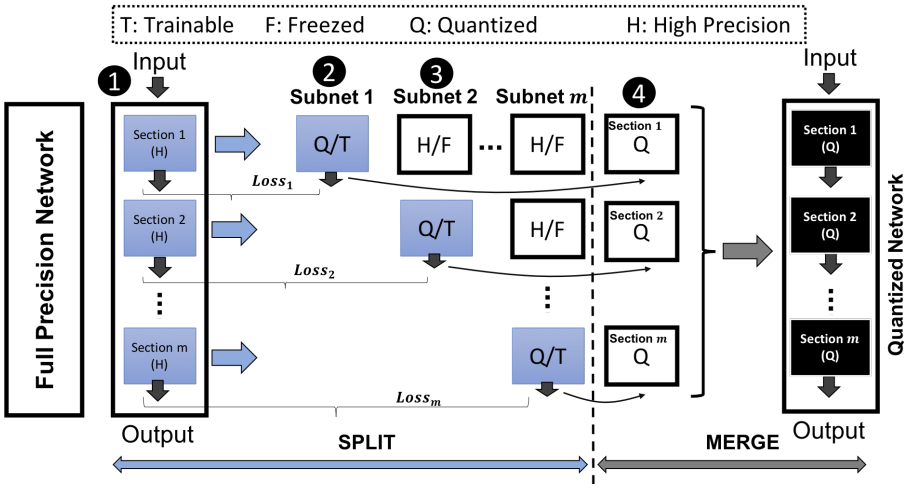

Figure 3.3: Divide and Conquer approach overview showing SPLIT phase; dividing the teacher full precision network into smaller subnetworks, and MERGE; by combining the training results of each subnetwork to form a fully quantized network

network in Figure 3.2 (b) while keeping it's output same as the corresponding $L-n$ layers of the full precision network, then we now have a deeply quantized network with the same accuracy as the full precision network. This is the underlying principle for our proposed quantization technique DCQ. In the above example, the network was split into two sections of n and $L-n$ layers, instead DCQ splits the original network into multiple sections and trains those sections individually to output same activations as the corresponding section in the full precision network. Following subsections explain the DCQ methodology in more detail.

3.2.2 Splitting, Training and Merging

Splitting the full precision network. As described in Section 3.2.1, DCQ splits the original network into multiple sections and trains them in isolation and in parallel. Figure 3.3 shows an overview of the entire process. As shown in the figure, after splitting the full precision network into m sub sections, DCQ quantizes and trains these subsections independently. After training, DCQ puts them all together again to get the deeply quantized version of the entire original network. As discussed in Section 3.2.1, because each of these sections is trained to

capture the same features as the full precision network, although these sections are trained independently, they can be put together at the end to give similar accuracy as the full precision network.

If the original network has L layers then m decides how many layers will be part of each section (sections need not be equal in terms of number of layers). In this work, we used a configuration of two layers per every section and then decided m according to the total number of layers in the network. Although, for networks like ResNet which have logical splits in terms of basic blocks, we split the network in a way that each section corresponds to a basic block. We leave the task of deciding the optimal number of sections (splits) and how many layers per section for a given network to future work. However, we provide some empirical analysis to this regard in Section 3.3.5.

Training the sub-networks. As Figure 3.3 illustrates, ❶ we create m sections in order to train each of the m sub-networks. For each section i, the sub-network i (or subnet i for short) consists of all the sections preceding it. Subnet 1 column in Figure 3.3 ❷ shows a subnet for section 1. To train this section, the output activations of the quantized version of section 1 are compared with the output activations of the full precision version of section 1 and the loss is calculated accordingly. Section 3.2.3 gives more details on how the loss is calculated for each subnet. Similarly, Subnet 2 column ❸ shows the subnet for section 2 and it comprises of both section 1 and section 2. Output activations of section 2 are used to calculate the loss in this case. Since section 2 is being trained in this subnet, weights for section 1 are frozen(not trainable) in this subnet and backpropagation based on the loss only affects section 2. Similarly there are subnets for sections 3 up to last section m and the last subnet m is basically similar to the full precision network except that the section m is quantized and all the other sections from 1 to $m-1$ are frozen.

Merging the sections. ❹ After training all the sections, since each of these sections has been trained independently to learn the same features as the corresponding section of the full

precision network but with quantized weights, they can be put together to form a fully trained quantized network. In every subnet, freezing all the sections except the one being trained is the key in enabling merging of all the individual sections at the end.

3.2.3 Loss Function for Training Sub Networks

All machine learning algorithms rely on minimizing a loss function to achieve a certain objective. The parameters of the network are trained by back-propagating the derivative of the loss with respect to the parameters throughout the network, and updating the parameters via stochastic gradient descent. Broadly speaking, according to the particular task, loss functions can be categorized into two types: Classification Loss and Regression Loss. Fundamentally, classification is about predicting a label (discrete valued output) and regression is about predicting a quantity (continuous valued output). Since DCQ aims to capture the intermediate features learnt by the full precision network, loss needs to be calculated based on the output activations of intermediate layers unlike the traditional loss which is calculated using the output of the final classification layer and the targets. As such, and in the context of this work focusing on classification tasks, DCQ divides the original classification problem into multiple *regression* problems by matching the intermediate feature (activation) maps. In this study, we have examined three of the most commonly used regression loss formulations. Namely:
(1) Mean Square Error (MSE): $\mathcal{L} = \frac{1}{n}\sum_{i=1}^{n}(y^{(i)} - \hat{y}^{(i)})^2$, (2) Mean Absolute Error (MAE): $\mathcal{L} = \frac{1}{n}\sum_{i=1}^{n}|y^{(i)} - \hat{y}^{(i)}|$, and (3) Huber Loss:

$$\mathcal{L} = \frac{1}{n}\sum_{i=1}^{n}\begin{cases} \frac{1}{2}(y^{(i)} - \hat{y}^{(i)})^2 & , \ |y^{(i)} - \hat{y}^{(i)}| \leq \delta \\ \delta(y^{(i)} - \hat{y}^{(i)}) - \frac{1}{2}\delta & , \ otherwise \end{cases}$$

where y is the target value, and $\hat{y}$ is the predicted value, and the summation is across all samples. For Huber loss, δ (delta) is a hyperparameter which can be tuned. Huber loss approaches MAE

when $\delta \sim 0$ and MSE when $\delta \sim \infty$ (large numbers). Section 3.3.5 provides experimental results for each of the above loss formulations.

3.2.4 Overall Algorithm

Algorithm 1 outlines the step by step procedure for DCQ putting together all the steps described in Sections 3.2.2 and 3.2.3. Since each iteration of the loop, shown in the algorithm, is independent, all the sections can potentially be trained in parallel leading to an overall reduction in training time.

Algorithm 1: Divide and Conquer for Quantization: Training Procedure

Input: Pretrained Full Precision Neural Network (N^{FP})
Output: Quantized Neural Network (N^{Q})

1: Split N^{FP} into **m** sections: $\{N_1, N_2, ..., N_m\}$; ▷ **`SPLIT phase`**
2: Each section N_i has a set of layers: $\{l_1, l_2, ..., l_n\}$
3: **for** N_i in $\{N_1, N_2, ..., N_m\}$ **do**
4: Create a subnet SN_i for section N_i containing all the sections from N_1 to N_i
5: Quantize all the layers in section N_i with the desired bitwidth to get N_i^q
6: Set all layers of section N_i as trainable, freeze all other remaining layers in the subnetwork SN_i
7: Calculate $LOSS_i$ using the output activations of section N_i of the full precision network and the subnetwork SN_i
8: Minimize $\{LOSS_i\}$ to train N_i^q to represent similar features as N_i
9: **end for**
10: $N^{Q} \leftarrow \text{merge}\{N_1^q, N_2^q, ..., N_m^q\}$; ▷ **`MERGE phase`**

3.3 Experimental Results

3.3.1 Experimental Setup

In this section, we evaluate the efficacy of our proposed approach on various DNNs (AlexNet, LeNet, MobileNet, ResNet-18, ResNet-20, SVHN and VGG-11) and different datasets: CIFAR10, ImageNet, MNIST, and SVHN. We compare our approach to conventional end-to-end

Table 3.1: Summary of results comparing DCQ (our appraoch) to DoReFa-Net for different networks considering binary and ternary weight quantization.

Weight Quantization	Benchmark	LeNet on MNIST	ResNet-20 on CIFAR10	SVHN-10 on SVHN	VGG-11 on CIFAR10
	Partitioning	2 Stages	4 Stages	2 Stages	3 Stages
	Method	Top-1 Accuracy (%)			
FP (W32)	Baseline	99.86	92.60	96.47	94.13
Binary (W1)	DoReFa	75.25	73.38	81.45	72.78
	DoReFa + DCQ	**99.28**	**90.52**	**93.21**	**87.48**
	Improvement	**31.9%** ↑	**23.3%** ↑	**14.4%** ↑	**20.2%** ↑
Ternary (W2)	DoReFa	90.91	85.24	89.56	81.98
	DoReFa + DCQ	**99.76**	**92.40**	**95.32**	**93.96**
	Improvement	**9.7%** ↑	**8.3%** ↑	**6.4%** ↑	**14.6%** ↑

training approach. We consider DoReFa-Net [101] as our baseline but also show comparision with BWN [76] in Section 3.3.2, and Apprentice [63] , in addition to state-of-the-art quantized training methods: PACT [15], LQ-Net [99], DSQ [33] in Section 3.3.3.

For all the experiments, we use an open source framework for quantization, Distiller [103]. While reporting accuracies in their paper, DoReFa-Net doesn't quantize first and last layers of the network whereas in our case, we quantize all the layers including the first and last layers. Because of this difference in quantization and using built-in implementation of Distiller, the accuracies we report might not exactly match the accuracies reported in their paper.

3.3.2 Binarization and Ternarization using DCQ

Table3.1 shows summary of results comparing plain DoReFa to DoReFa + DCQ for different networks considering binary {-1,1} and ternary {-1, 0, 1} weight quantization for various networks: LeNet, ResNet-20, SVHN and VGG-11. As seen, integrating DCQ into DoReFa outperforms the conventional approach and achieves a consistent improvements across the different networks with average 22.45% for binarization and 9.7% for ternarization.

Delving into the results, the reported improvements can be attributed to the following

Table 3.2: Summary of results comparing our approach (DCQ) to state-of-the-art quantized training methods.

Bitwidth	Benchmark	AlexNet		ResNet-18		MobileNet-V2	
	Partitioning	3 Stages		3 Stages		3 Stages	
	Method	Top-1	Top-5	Top-1	Top-5	Top-1	Top-5
W32/A32	Full Precision	57.1	80.2	70.1	89.5	71.8	90.3
W4/A4	PACT	55.7	-	69.2	89.0	61.4	83.7
	LQ-Nets	-	-	69.3	88.8	-	-
	DSQ	-	-	69.6	-	64.8	-
	DoReFa	55.0	76.3	68.9	88.1	64.6	85.1
	DoReFa + DCQ	**56.2**	**79.2**	**69.9**	**89.2**	**66.2**	**87.3**
	Improvement	**0.89%** ↑		**0.43%** ↑		**2.47%** ↑	
W3/A3	PACT	55.6	-	68.1	88.2	-	-
	LQ-Nets	-	-	68.2	87.9	-	-
	DSQ	-	-	68.7	-	-	-
	DoReFa	54.1	75.1	67.9	87.5	60.1	83.0
	DoReFa + DCQ	**55.8**	**77.2**	**69.2**	**89.9**	**62.2**	**88.7**
	Improvement	**0.36%** ↑		**0.72%** ↑		**3.49%** ↑	

reasons. First, deep multi-hidden-layer neural networks are much more difficult to tackle as compared to shallower ones. Furthermore, end-to-end backpropagation can be inefficient [46]. Thus, adopting such divide and conquer approach yields simpler subproblems that are easier to optimize. Second, matching intermediate learning objectives also guides the optimization as compared to following a single global objective that indirectly specifies learning objectives to the intermediate layers.

Comparison with BWN. BWN [76] proposes approximate convolutions using binary operations for a set of networks. We show comparison on LeNet as it is the only common benchmark between both the works. As Table 3.1 shows, our technique achieves an accuracy of 99.3%, which is close to the accuracy of 99.2% reported by BWN. However, BWN involves restructuring the original network architecture whereas our implementation does not introduce

Table 3.3: Comparing DCQ to a knowledge distillation based quantization method, Apprentice.

Method	ResNet-20 on CIFAR10	ResNet-18 on ImageNet
	Top-1 Accuracy (%)	
	W2/A32	W2/A32
Apprentice	92.00	66.60
DCQ	**93.40**	**67.90**

any changes to the architecture.

3.3.3 Comparison with Quantized Training Methods

Here, we provide comparison to multiple state-of-the-art quantized training methods considering both weights and activation quantization. Table 3.2 summarizes the results of comparing to PACT, LQ-Net, DSQ, and DoReFa (the baseline) for several networks (AlexNet, ResNet-18, MobileNet). As seen, DCQ outperforms these previously reported accuracies and achieves on average improvements of 0.98%, and 0.96% for W4/A4 and W3/A3, respectively.

We also provide a comparison against knowledge distillation-based method Apprentice [63], a recent work which also combines knowledge distillation with quantization. Table 3.3 shows that our technique outperforms Apprentice for both ResNet-20 on CIFAR10, and ResNet-18 on ImageNet considering ternary weights quantization. The reported improvement can be attributed to the fact that DCQ combines the conventional knowledge distillation approach, as in [63], in addition to its unique intermediate learning approach by regressing the quantized network intermediate feature maps to the corresponding full precision ones in a stage wise fashion. Moreover, the network architecture of the student network in [63] is typically different from that of the teacher network as opposed to DCQ where same network architecture is utilized for the student network but with quantized weights. From one side, this saves a huge amount of effort designing a student network architecture which might incur significant hyperparameter tuning. On the other side, it enables a direct finetuning instead of a complete training from scratch as a

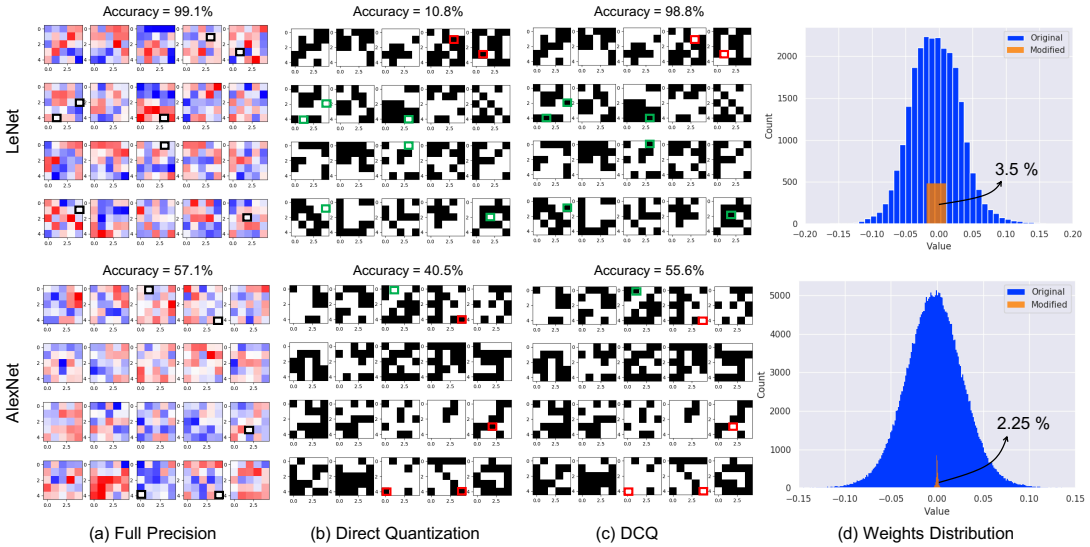

Figure 3.4: Visualization of a subset of weight kernels of the second convolutional layer of LeNet (top row), and AlexNet (bottom row), highlighting the differences between different versions of binary weight kernels: (a) Full precision weight kernels, (b) binary weight kernels upon direct binarization from full precision, (c) binary weight kernels obtained using our method DCQ, and (d) weights histogram of the convolutional layer highlighting the altered binary weights after training (using DCQ) relative to the original distribution

result of preserving the original network architecture.

3.3.4 Analysis: DCQ vs Conventional Binary Kernels

This section provides an analysis of our obtained binary weight kernels and sheds light on some interesting observations. We start by posing the following questions: how are trained binary weight kernels different from just direct binarization from the original full precision weight kernels? and whether different training algorithms can yield qualitatively different binary weight kernels?

Figure 3.4 shows a visualization of a subset of weight kernels from the second convolutional layer of LeNet and AlexNet. (a) is the original full precision kernels, (b) direct binarization of full precision kernels, and (c) binarization after training (applying DCQ). In the figure, weights that are different between the trained binary kernel and the directly binarized

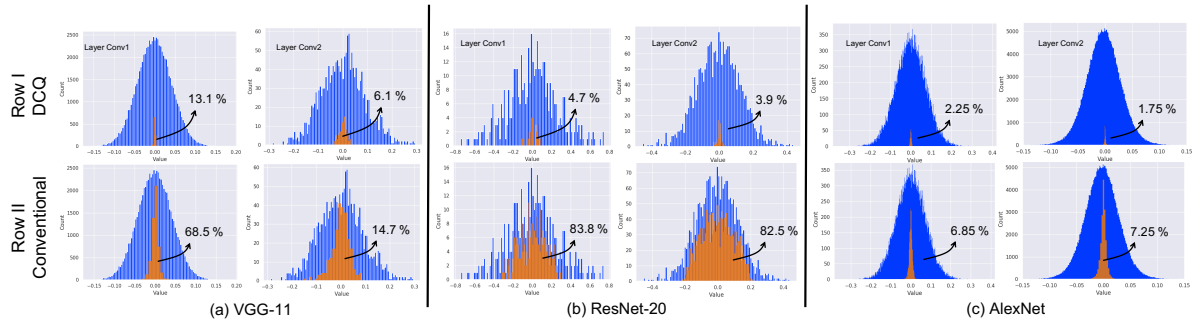

Figure 3.5: Weights histograms of the first two convolutional layers of three different DNNs: (a) VGG11, (b) ResNet-20, and (c) AlexNet, highlighting the altered portion of the trained binary weights (depicted percentages indicate the exact portion in orange) relative to the directly binarized weights. Original total weights histograms are shown in blue. Row I shows the results using our method (DCQ), and Row II shows for the conventional end-to-end taining method.

kernel are highlighted with square rectangles across the three visualizations. Spatially contrasting those highlighted altered weights on the full precision kernels, it can be noticed that they mostly share a common feature that is being low in magnitude (shown as white squares in (a)). From statistical point of view, Figure 3.4 (d) shows the original full precision weights histogram (in blue) and overlaying the portion of the altered weights (in light orange). We can observe the following. First, during training, only very small percentage of the weights are actually altered relative to the total number of weights. Specifically, in this example, it is around 3.5% and 2.25% for LeNet and AlexNet respectively, of the total weights got impacted by training. Moreover, despite the marginal difference between the binary kernels, they experience dramatic accuracy difference: 10.8% vs 98.1% for kernels in (b) and (c) respectively, for LeNet, and 40.5% and 55.6% for AlexNet.

Now, to check whether this is a general trend and whether different training algorithms has an impact on this, we extend our statistical analysis to more networks. Figure 3.5 shows weight histograms of the first two convolutional layers of AlexNet, ResNet-20, and VGG-11. As seen in the figure, first, for Figure 3.5 Row I (DCQ), the altered portion of binary weights during training is consistently small in both number and magnitude across different layers and different networks. Second, contrasting that behavior using DCQ vs using the conventional end-to-end

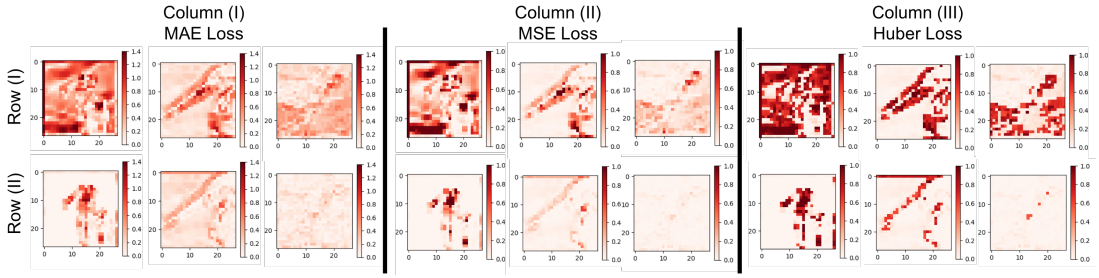

Figure 3.6: Loss visualization of intermediate feature maps samples. Row(I): before DCQ training, Row(II): after DCQ training. Columns show results for different loss formulations. Col(I) MAE, Col(II) MSE, and Col(III) Huber loss. The results are for the second convolution layer in AlexNet with binary quantization.

quantized training, as shown in Figure 3.5 Row II (Conventinoal), we see that binary weight kernels clearly encounter much more variations during the conventional end-to-end training as compared to our approach, DCQ.

Comparing the two training algorithms, DCQ *yields minimal changes in the right place* to the binary weights as the entire technique is based on matching the intermediate features represented by weight kernels. Which, consequently, leads to faster convergence behavior and higher solution quality at the same time. Moreover, this opens up the possibility of *magnitude-constrained weight training* where only weights below a certain magnitude are set to be trainable which can potentially improve the optimization process further.

3.3.5 Exploratory Studies

Impact of different loss formulations for intermediate learning. As mentioned in section 3.2.3, we have examined three of the most commonly used loss formulations. Namely: (1) Mean Square Error (MSE); (2) Mean Absolute Error (MAE); (3) Huber Loss. Figure 3.6 shows different samples of feature maps losses (for the second convolution layer of AlexNet wtih binary weights). Row(I) shows different samples of feature map losses before DCQ training. Row(II) shows the losses for the same samples after DCQ training (matching feature maps). Different columns show different loss formulations. Col(I): MSE Loss; Col(II): MAE Loss; and Col(III):

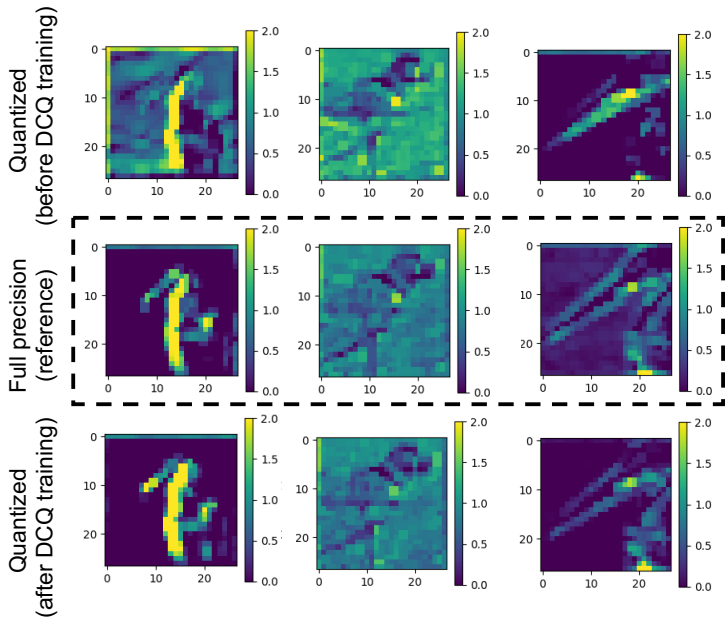

Figure 3.7: Feature maps before and after DCQ training compared to full precision maps. The results are for the second convolution layer in AlexNet with binary quantization.

Huber Loss. As it can be seen, the feature map losses (the amount of redness) significantly decreases after DCQ training as a result of regressing the quantized model intermediate feature maps to the full precision counterparts. We can also notice that the behavior is consistent across different regression losses. Nevertheless, based on our experimentation, among the considered formulations, MSE seems to be the most effective during the intermediate learning process. The trends are similar for the other networks. Figure 3.7 compares visualizations of different samples of actual feature maps before and after DCQ training with respect to the full precision ones demonstrating the effectiveness of the proposed approach. Lastly, divide and conquer is a very basic and universal engineering principle that is commonly and widely applied across a variety of fields. Here, we propose a procedure that extends such effective principle to quantized training of neural networks.

Impact of the number of splitting points. As number of splitting points increases, the large optimization problem gets divided into smaller subproblems. Thus, on one side, it becomes easier to solve each subproblem separately. On the other side, however, the complexity overhead

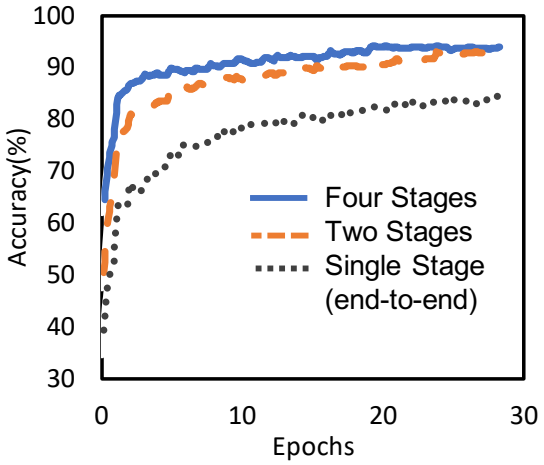

Figure 3.8: Impact of different splitting on the convergence behavior for VGG-11 (ternary quantization).

increases as well. We leave the optimal choice of how many stages a network should be divided and how many layers per stage to future work. Here, we provide one experimental example to give some intuition about the impact of different splitting points. Figure 3.8 shows the convergence behavior for different splittings of VGG-11: four-stage and two-stage splitting as compared to single stage (conventional knowledge distillation). As seen in the figure, not only the convergence is faster as number of stages increases but also it eventually converges to a higher final accuracy as compared to lesser number of stages or no splitting at all.

3.3.6 Memory Analysis

Compared to DoReFa, DCQ only needs an extra set of weights (divided across the nodes) which is same as the other conventional knowledge distillation approaches. However, DCQ does not impose any extra memory requirements on the activations. Analysis follows. DoReFa maintains weights in full-precision (FP) and quantizes them during inference, so, for a network N, total memory taken by DoReFa is all the FP weights (W_{fp}) of N. DCQ sections the network N, to subnets: $S_1, \ldots, S_i, \ldots, S_m$, and maps them to parallel nodes: $C_1, \ldots, C_i, \ldots, C_m$. Since DCQ has a FP version of the entire network, C_1 is also responsible to run the inference in the FP mode.

Each C_i node only keeps a subset of the FP weights ($W_{fp,i}$) corresponding to its subnet S_i and only trains that subnet. C_1, which runs the whole network in FP, sends each subnet $S_i's$ inputs and outputs to the corresponding nodes ($C_i s$). As such, all the $C_i s$ can operate in parallel since they use knowledge distillation and only need to have their respective $W_{fp,i}$. Memory usage in C_1 node $= W_{fp} + W_{fp,1}$. Memory usage in all other C_i nodes $= W_{fp,i}$. Overall memory usage in the parallel system $= W_{fp} + sum(W_{fp,i}) = 2W_{fp}$ (same as conventional knowledge distillation techniques)

3.4 Theoretical Analysis

One issue that arises as a result of the strategy of splitting into sections and training each section separately is accumulation of error residuals through sections which may impact the overall performance of the proposed technique. Here, we theoretically derive an upper bound on the total accumulated error across the resulting subnetworks after splitting using a chaining argument and utilizing Lipschitz continuity. A rigorous analysis bounding the Lipschitz constant of a deep network can be found in [88] for arbitrary networks and [105] for particular convolutional networks.

We provide a worst case upper bound on the error, but it is also possible to establish probabilistic bounds on the error under the assumption that the quantization error on the weights is uniformly random. In particular, one can directly apply the bounds from [80] to attain probabilistic bounds on the classification error even for our layer-wise quantization framework.

3.4.1 Upper Bounding Network-wide Error

Let's consider a feed-forward full precision network with the following function formulation.

$$f_{fp}(x) = (\phi^{(m)} \circ \phi^{(m-1)} \circ ... \circ \phi^{(1)})(x)$$

where $\phi^{(i)}$ is a given layer of the network. Also, for a given layer, let the quantized layer be denoted $\phi_q^{(i)}$. If we quantize every layer, we will refer to the fully quantized network f_q.

Assume the application of our quantization scheme leads to an error in the output of size $\|\phi^{(i)}(x) - \phi_q^{(i)}(x)\| < \delta$. This comes from the quantization error guarantee of the used technique. Unless otherwise stated, $\|\cdot\|$ refers to the 2-norm. Further, assume that $\phi^{(i)}$ has Lipschitz constant L_i. Every one layer network is always a Lipschitz function, where L_i is always bounded by the norm of the weights matrix (see Appendix 3.4.3 for a full description). Under this model, we can use a simple triangle inequality to get $\|\phi_q^{(i)}(x) - \phi_q^{(i)}(y)\| < L_i\|x-y\| + 2\delta$. Using this fact, and chaining it together across multiple layers, we are able to bound the pointwise error between the full precision network and the quantized network.

Theorem 1. *Let f_{fp} be an m layer network, and each layer has Lipschitz constant L_i. Assume that quantizing each layer leads to a maximum pointwise error of δ_i, and results in a quantized m layer network f_q. Then for a point $x \in X$, f_q satisfies*

$$\|f_q(x) - f_{fp}(x)\| \leq 3\Delta_{m,L},$$

where $\Delta_{m,L} = \delta_m + \sum_{i=1}^{m-1} \left(\prod_{j=i+1}^{m} L_j \right) \delta_i$.

The proof can be found in the Appendix 3.4.4.

As the Lipschitz constant of the network is the product of its individual layers' Lipschitz constants, L can grow exponentially if $L_i \geq 1$. This is the common case for normal network training [18], and thus the perturbation will be amplified for such a network. Therefore, to keep the Lipschitz constant of the whole network small, we need to keep the Lipschitz constant of each layer $L_i < 1$. This is often done using regularization or weight clipping [9, 18, 34, 85] to suppress network's accumulation of error. We call a network with $L_i < 1, \forall i = 1, ..., L$ a non-expansive network. Experimentally, Lipschitz constant of each layer is found empirically by taking $\max_{x,y} \|\phi_i(x) - \phi_i(y)\| / \|\phi_{i-1}(x) - \phi_{i-1}(y)\|$.

3.4.2 Lipschitz Constants in Classification Networks

The Lipschitz constant is traditionally defined for regression problems where f can take arbitrary values on $\mathbb{R}$, but it also has implications for classification networks. For a classification network, the input is labeled data (x_i, y_i) for y_i coming from one of K classes. Then the last regression layer output $f(x)$ is a function $f : X \to \mathbb{R}^K$. This either directly predicts the probability of classification, or is fed into a softmax layer to normalize the probabilities. We will work with the $f(x)$ regression layer (prior to the softmax if there is one) for the subsequent theory, and use the notation that a network classifies x_i as class k if and only if $f(x_i)_k > f(x_i)_j$ for all $j \neq k$. This still applies even if a softmax layer is added, as the softmax does not alter the relative order of its inputs.

A common problem for classification networks is to determine how much one can perturb the data point x_i and maintain the correct classification.

Definition 1. *The* output margin *of a data point* (x_i, y_i) *is*

$$r_i := \frac{1}{2}\left(f(x_i)_k - \max_{j \neq k} f(x_i)_j\right)_+$$

for $y_i = k$, *and* $(x)_+ = \max(x, 0)$.

This is half the minimum amount one must change the network output to change the classification of x_i from class k to some other class. This leads to the following theorem.

Theorem 2. *Let* f_{fp} *and* f_q *be the full precision and quantized m layer networks as in Section 3.4.1. Let* $L = \prod_{i=1}^{m} L_i$ *be the Lipschitz constant of* f_{fp}. *Let* (x_i, y_i) *be a data point where* f_{fp} *correctly classifies* x_i *with output margin* $r_i > 0$. *Then for any perturbation* η *such that*

$$\|\eta\| < \frac{r_i - 5\Delta_{m,L}}{L},$$

f_q *will also classify* $x_i + \eta$ *correctly.*

The proof can be found in the Appendix 3.4.4.

This leads to a final method for bounding the probability of misclassification across all data points for DCQ. The proof can be seen as a byproduct of Theorem 2 where we count the number of points for which it's possible to perturb x_i with a nonzero η and maintain the correct classification.

Theorem 3. *Let e_{fp} be the classification error probability of a full precision network f_{fp}, and e_q the classification error probability of the* DCQ *quantized network f_q. Then we can bound the quantized classification error probability by*

$$e_q \leq e_{fp} + (1 - e_{fp})\mathbb{E}_{x_i \in X}\left[\mathbf{1}_{r_i \leq 5\Delta_{m,L}} \middle| \widehat{y}_{i,fp} = y_i\right],$$

where r_i is the output margin of x_i for f_{fp}, and $\widehat{y}_{i,fp}$ is the estimated class of x_i using f_{fp}.

The proof can be found in the Appendix 3.4.4. We note that r_i can be easily checked for a given full precision network by examining the last regression layer across all points in the data set.

3.4.3 Lipschitz Constants

The Lipschitz constant describes: when input changes, how much does the output change correspondingly. For a function $f : X \to Y$, if it satisfies

$$\|f(x_1) - f(x_2)\|_Y \leq L\|x_1 - x_2\|_X, \ \ \forall\, x_1, x_2 \in X$$

for $L \geq 0$, and norms $\|\cdot\|_X$ and $\|\cdot\|_Y$ on their respective spaces, then we call h Lipschitz continuous and L is the known as the Lipschitz constant of h.

For a one layer network, full precision network f_{fp} has Lipschitz constant L, which

satisfies

$$L \leq C_\sigma \|W_{fp}\| \text{ for } C_\sigma = \frac{d\sigma}{dx}.$$

This bound is immediate from the fact that $\nabla f_{fp}(x) = \sigma'(W_{fp}x) \cdot \begin{bmatrix} W_{\cdot,1} & \dots & W_{\cdot,d} \end{bmatrix}$, and $L \leq \max_x \|\nabla f_{fp}(x)\|$.

3.4.4 Proofs and Additional Lemmas

Lemma 1. *Let f_{fp} be an m layer network, and each layer has Lipschitz constant L_i. Assume that quantizing each layer leads to a maximum pointwise error of δ_i, and results in a quantized m layer network f_q. Then for any two points $x, y \in X$, f_q satisfies*

$$\|f_q(x) - f_q(y)\| < \left(\prod_{j=1}^{m} L_j\right) \|x-y\| + 2\Delta_{m,L},$$

where $\Delta_{m,L} = \delta_m + \sum_{i=1}^{m-1} \left(\prod_{j=i+1}^{m} L_j\right) \delta_i$.

Proof of Lemma 1. Let $\phi_q^{(i)}$ be the quantized i^{th} layer of the network. From Section 3.4.3, we know that

$$\|\phi_q^{(i)}(x) - \phi_q^{(i)}(y)\| < L_i\|x-y\| + 2\delta_i.$$

Similarly, we know that feeding in the previous layer's quantized output yields

$$\begin{aligned} \|\phi_q^{(2)} \circ \phi_q^{(1)}(x) - \phi_q^{(2)} \circ \phi_q^{(1)}(y)\| &\leq L_2\|\phi_q^{(1)}(x) - \phi_q^{(1)}(y)\| + 2\delta_2 \\ &\leq L_2 L_1 \|x-y\| + 2L_2\delta_1 + 2\delta_2. \end{aligned}$$

By chaining together the i layers inductively up to m, we complete the desired inequality.

□

Proof of Theorem 1. We know that $\|\phi_q^{(1)}(x) - \phi^{(1)}(x)\| < \delta_1$. This means $\phi^{(2)}$ receives different

inputs depending on whether $\phi^{(1)}$ was quantized or not, and thus requires the Lipschitz bound. Thus

$$\begin{aligned}\|\phi_q^{(2)}((\phi_q^{(1)}(x))-\phi^{(2)}(\phi^{(1)}(x))\| &\leq \|\phi_q^{(2)}(\phi_q^{(1)}(x))-\phi_q^{(2)}(\phi^{(1)}(x))\|+\|\phi_q^{(2)}(\phi^{(1)}(x))-\phi^{(2)}(\phi^{(1)}(x))\| \\ &\leq \left(L_2\|\phi_q^{(1)}(x)-\phi^{(1)}(x)\|+2\delta_2\right)+\delta_2 \\ &\leq 2L_2\delta_1+3\delta_2,\end{aligned}$$

where the second ineuqlity comes from Lemma 1. Chaining the argument for the i^{th} layer inductively up to m, we arrive at the desired inequality. □

Proof of Theorem 2. From the guarantee of Lemma 1, we know

$$\|f_q(x+\eta)-f_q(x)\|\leq L\|(x+\eta)-x\|+2\Delta_{m,L}.$$

If we consider a full precision network f_{fp} that classifies x_i correctly with output margin $r_i > 0$, then we must simply apply a triangle inequality to attain

$$\begin{aligned}\|f_q(x_i+\eta)-f_{fp}(x_i)\| &\leq \|f_q(x_i+\eta)-f_q(x_i)\|+\|f_q(x_i)-f_{fp}(x_i)\| \\ &\leq L\|(x_i+\eta)-x_i\|+2\Delta_{m,L}+3\Delta_{m,L}.\end{aligned}$$

Thus for η such that $\|\eta\|<\frac{r_i-5\Delta_{m,L}}{L}$, we will attain $\|f_q(x_i+\eta)-f_{fp}(x_i)\|<r_i$.

Since we also have that $\|z\|_\infty\leq\|z\|_2$ for any $z\in\mathbb{R}^K$, this means that $\|f_q(x_i+\eta)-f_{fp}(x_i)\|_\infty<r_i$. If f_{fp} classifies x_i as class k, this means that

$$f_{fp}(x_i)_k-f_{fp}(x_i)_j\geq 2r_i,\ \forall j\neq k.$$

By the triangle inequality, we get

$$\begin{aligned} f_q(x_i+\eta)_k - f_q(x_i+\eta)_j &= f_q(x_i+\eta)_k - f_q(x_i+\eta)_j \pm f_{fp}(x_i)_k \pm f_{fp}(x_i)_j \\ &= (f_q(x_i+\eta)_k - f_{fp}(x_i)_k) - (f_q(x_i+\eta)_j - f_{fp}(x_i)_j) \\ &\qquad + (f_{fp}(x_i)_k - f_{fp}(x_i)_j) \\ &> -r_i - r_i + 2r_i \\ &\geq 0. \end{aligned}$$

Since this difference is strictly greater than 0, f_q classifies $x+\eta$ correctly. □

Proof of Theorem 3. Let $\widehat{y}_{i,fp}$ be the estimated class of x_i using f_{fp} and $\widehat{y}_{i,q}$ be the estimated class of x_i using f_q. We use basic probabilistic bounds (where the probability is a uniform distribution over the dataset) to arrive at

$$\begin{aligned} e_q &= \Pr(\widehat{y}_{i,q} \neq y_i) \\ &= \Pr(\widehat{y}_{i,q} \neq y_i \text{ and } \widehat{y}_{i,fp} \neq y_i) + \Pr(\widehat{y}_{i,q} \neq y_i \text{ and } \widehat{y}_{i,fp} = y_i) \\ &\leq \Pr(\widehat{y}_{i,fp} \neq y_i) + \Pr(\widehat{y}_{i,fp} = y_i \text{ and } \widehat{y}_{i,q} \neq \widehat{y}_{i,fp}) \\ &\leq e_{fp} + \Pr(\widehat{y}_{i,fp} = y_i) \Pr(\widehat{y}_{i,q} \neq \widehat{y}_{i,fp} | \widehat{y}_{i,fp} = y_i) \\ &\leq e_{fp} + (1-e_{fp}) \Pr(\widehat{y}_{i,q} \neq \widehat{y}_{i,fp} | \widehat{y}_{i,fp} = y_i) \\ &= e_{fp} + (1-e_{fp})(1 - \Pr(\widehat{y}_{i,q} = \widehat{y}_{i,fp} | \widehat{y}_{i,fp} = y_i)) \end{aligned}$$

All that remains is lower bounding the final conditional probability of matching. However, this can be done using Theorem 2. We know that $\widehat{y}_{i,q} = \widehat{y}_{i,fp}$ so long as $\|f_q(x_i) + f_{fp}(x_i)\|_\infty < r_i$. From Theorem 2, a sufficient condition for this is for $r_i - 5\Delta_{m,L} > 0$, as this implies one can construct a neighborhood of positive radius $\|\eta\| < \frac{r_i - 5\Delta_{m,L}}{L}$ such that $\|f_q(x_i+\eta) + f_{fp}(x_i)\|_\infty < r_i$.

In particular, this implies $\|f_q(x_i) + f_{fp}(x_i)\|_\infty < r_i$ by choosing $\eta = 0$. This gives us

$$\begin{aligned}
\Pr(\widehat{y}_{i,q} = \widehat{y}_{i,fp} | \widehat{y}_{i,fp} = y_i) &= \Pr(\|f_q(x_i) + f_{fp}(x_i)\|_\infty < r_i | \widehat{y}_{i,fp} = y_i) \\
&\geq \Pr(\exists \delta \geq 0, \forall \|\eta\| < \delta, \|f_q(x_i + \eta) + f_{fp}(x_i)\|_\infty < r_i | \widehat{y}_{i,fp} = y_i) \\
&\geq \Pr\left(\frac{r_i - 5\Delta_{m,L}}{L} > 0 \middle| \widehat{y}_{i,fp} = y_i \right) \\
&= \mathbb{E}_{x_i \in X} \left[\mathbf{1}_{r_i > 5\Delta_{m,L}} \middle| \widehat{y}_{i,fp} = y_i \right].
\end{aligned}$$

Combining these terms, we arrive at

$$\begin{aligned}
e_q &\leq e_{fp} + (1 - e_{fp}) \left(1 - \mathbb{E}_{x_i \in X} \left[\mathbf{1}_{r_i > 5\Delta_{m,L}} \middle| \widehat{y}_{i,fp} = y_i \right] \right) \\
&= e_{fp} + (1 - e_{fp}) \mathbb{E}_{x_i \in X} \left[\mathbf{1}_{r_i \leq 5\Delta_{m,L}} \middle| \widehat{y}_{i,fp} = y_i \right].
\end{aligned}$$

□

3.5 Related Work

Knowledge distillation. Knowledge distillation [39] is proposed to attain a smaller or shallower neural network (student) from one or an ensemble of bigger deep networks (teacher). The student network is trained on a softened version of the *final* output of teacher(s) [12]. FITNETS [78] extends knowledge distillation by extracting a hint from the teacher to train even a deeper but thinner student. The hint is an intermediate feature representation of the teacher, that is used as a regularizer to pretrain the first few layers of the deep and thin student network. After the pretraining phase, the full knowledge distillation is used to finish the training of the student. FITNETS [78] does not explore hints from more than one intermediate layer of the teacher. Furthermore, FITNETS applies the knowledge distillation pass over the entire student network at once. FITNETS are a complementary approach to our sectional knowledge distillation and similar

hints can be utilized for each section. Nonetheless, the following discusses the differences. In contrast to this technique, DCQ (1) partitions the neural network to multiple independent sections and (2) applies knowledge distillation to each section in isolation and trains them independently, (3) not utilizing the intermediate representations as hint for pretraining. (4) After the sections are trained through knowledge distillation, they are put together instead of applying another phase of training as done in FITNETS [78]. (5) Moreover, DCQ, exclusively, applies various regression losses in matching the quantized network intermediate feature maps to the corresponding full precision ones in a stage wise fashion. (6) Last but not least, the objective differ as the knowledge distillation and FITNETS aim to compress the network while DCQ quantizes it preserving the teacher's original network architecture.

Other work [95] proposes an information metric, in terms of inter-layer flow (the inner product of feature maps), using which a teacher DNN can transfer the distilled knowledge to other student DNNs.

Knowledge distillation is also used for training a lower bitwidth student network from a full-precision teacher [63, 73, 89]. However, these works do not partition the network as DCQ does and also do not utilize teacher's intermediate layers.

Other quantization techniques. Several techniques have been proposed for quantizing DNNs: algorithmic-wise [24,25,65,101,102], and hardware-wise [31,81]. DoReFa-Net [101] uses straight through estimator [10] for quantization and extends it for any arbitrary k bit quantization. DoReFa-Net also proposes a method to train a CNNs with low bitwidth weights and activations, low bitwidth parameter gradients using deterministic quantization of weights, activations and stochastic quantization of activations. TTQ [102] proposes a method to reduce the weights to ternary values by adding scaling coefficients to each layer. These scaling coefficients are learnt during training and during deployment, weights are directly quantized to ternary bitwidths and these scaling coefficients are used to scale the weights during inference. PACT [15] proposes a technique for quantizing activations using an activation clipping parameter which is optimized

during training. There have also been a lot of efforts [43, 58, 75] to binarize neural networks at the cost of some accuracy loss.

However, these inspiring efforts do not introduce sectioning nor they leverage knowledge distillation in the context of either quantization or binarizing the neural networks.

3.6 Conclusion

Quantization offers a promising path forward to reduce the compute complexity and memory footprint of deep neural networks. This work sets out to tackle the main challenge in quantization, recovering as much accuracy as possible. To that end, we developed a sectional multi-backpropagation algorithm that leverages multiple instances of knowledge distillation and intermediate feature representations to teach a quantized student through divide and conquer. This algorithm, DCQ, achieves significantly higher accuracy compared to the state-of-the-art quantization methods by exploring a new sectional approach towards knowledge distillation.

Acknowledgment. Chapter 3, in part, contains a re-organized reprint of the material as it appears in *International Conference on Machine Learning*. Ahmed T. Elthakeb, Prannoy Pilligundla, Fatemeh Mireshghallah, Alexander Cloninger, Hadi Esmaeilzadeh, 2020. The dissertation author was the primary investigator and author of this paper.

Chapter 4

Gradient-Based Deep Quantization of Neural Networks through Sinusoidal Adaptive Regularization

Deep quantization of neural networks below eight bits can lead to superlinear benefits in storage and compute efficiency. However, homogeneously quantizing all the layers to the same level does not account for the distinction of the layers and their individual properties. Heterogeneous assignment of bitwidths to the layers is attractive but opens an exponentially large non-contiguous hyperparameter space (Available Bitwidths$^{\#\,\text{Layers}}$). Thus, finding the bitwidth while also quantizing the network to those levels becomes a major challenge. This work addresses this challenge through a sinusoidal regularization mechanism, dubbed WaveQ. Adding our parametrized sinusoidal regularizer enables WaveQ to not only find the quantized weights, but also learn the bitwidth of the layers by making the period of the sinusoidal regularizer a trainable parameter. In addition, the sinusoidal regularizer itself is designed to align its minima on the quantization levels. With these two innovations, during training, stochastic gradient descent uses the form of the sinusoidal regularizer and its minima to push the weights to the quantization levels

while it is also learning the period which will determine the bitwidth of each layer separately. As such WaveQ is a gradient-based mechanism that jointly learns the quantized weights as well as the heterogeneous bitwidths. We show that WaveQ balances compute efficiency and accuracy, and provides a heterogeneous bitwidth assignment for quantization of a large variety of deep networks (AlexNet, CIFAR-10, MobileNet, ResNet-18, ResNet-20, SVHN, and VGG-11) that virtually preserves the accuracy. WaveQ is versatile and can also be used with predetermined bitwidths by fixing the period of the sinusoidal regularizer. In this case, WaveQ, on average, improves the accuracy of quantized training algorithms (DoReFa and WRPN) by $\sim 4.8\%$, and outperforms multiple state-of-the-art techniques. Finally, WaveQ is applicable to quantizing transformers and yields significant benefits.

4.1 Introduction

Quantization, in general, and deep quantization (below eight bits) [51], in particular, aims to not only reduce the compute requirements of DNNs but also reduce their memory footprint [43, 47, 65, 83, 101]. Nevertheless, without specialized training algorithms, quantization can diminish the accuracy. As such, the practical utility of quantization hinges upon addressing two fundamental challenges: (1) discovering the appropriate bitwidth of quantization for each layer while considering the accuracy; and (2) learning weights in the quantized domain for a given set of bitwidths.

This proposal formulates both of these challenges as a *gradient-based* joint optimization problem by introducing an additional novel sinusoidal regularization term in the training loss, called WaveQ. The following two main insights drive this work. (1) Sinusoidal functions (sin^2) have inherent periodic minima and by adjusting the period, the minima can be positioned on quantization levels corresponding to a bitwidth at per-layer granularity. (2) As such, sinusoidal period becomes a direct and continuous representation of the bitwidth. Therefore, WaveQ incorporates this continuous variable (i.e., period) as a differentiable part of the training loss in

the form of a regularizer. Hence, WaveQ is a differentiable regularization mechanism, it piggy backs on the stochastic gradient descent that trains the neural network to also learn the bitwidth (the period). Simultaneously this parametric sinusoidal regularizer pushes the weights to the quantization levels (sin^2 minima).

By adding our parametric sinusoidal regularizer to the original training objective function, our method automatically yields the bitwidths for each layer along with nearly quantized weights for those bitwidths. In fact, the original optimization procedure itself is harnessed for this purpose, which is enabled by the differentiability of the sinusoidal regularization term. As such, quantized training algorithms [65, 101] that still use some form of backpropagation [79] can effectively utilize the proposed mechanism by modifying their loss. Moreover, the proposed technique is flexible as it enables heterogeneous quantization across the layers. The WaveQ regularization can also be applied for training a model from scratch, or for fine-tuning a pretrained model.

In contrast to the prior inspiring works [27, 86], WaveQ is the only technique that casts finding the bitwidths and the corresponding quantized weights as a simultaneous gradient-based optimization through sinusoidal regularization during the training process. We also prove a theoretical result to provide an insight on why the proposed approach leads to solutions preserving the original accuracy during quantization. We evaluate WaveQ using different bitwidth assignments across different DNNs (AlexNet, CIFAR-10, MobileNet, ResNet-18, ResNet-20, SVHN, and VGG-11). To show the versatility of WaveQ, it is used with two different quantized training algorithms, DoReFa [101] and WRPN [65]. Over all the bitwidth assignments, the proposed regularization, on average, improves the top-1 accuracy of DoReFa by 4.8%. The reduction in the bitwidth, on average, leads to 77.5% reduction in the energy consumed during the execution of these networks. Finally, we apply WaveQ to Transformer DNNs citepby augmenting their loss with WaveQ parametric sinusiodal regularization. In this case, the conventional stochastic gradient descent plus WaveQ regularization is used to quantize the big Transformer model from [70] for machine translation on the IWSLT14 German-English dataset [1]. For 5, 6, and 7-bit quantization,

training with WaveQ yields 0.46, 0.14, 0.04 improved BiLingual Evaluation Understudy (BLEU) score, respectively. As a point of reference, the original big Transformer model from [70] improves the BLEU by only 0.1 over the state-of-the-art.

4.2 Joint Learning of Layer Bitwidths and Quantized Parameters

Our proposed method WaveQ exploits weight regularization in order to automatically quantize a neural network while training. To that end, Section 4.2.1 describes the role of regularization in neural networks and then Section 4.2.2 explains WaveQ in more details.

4.2.1 Preliminaries

Soft constraints through regularization and the loss landscape of neural networks. Neural networks' loss landscapes are known to be highly non-convex and it has been empirically verified that loss surfaces for large networks have many local minima that essentially provide equivalent test errors [17, 59]. This opens up the possibility of adding soft constrains as extra custom objectives during the training process, in addition to the original objective (i.e., to minimize the accuracy loss). The added constraint could be with the purpose of increasing generalization or imposing some preference on the weights values.

Regularization in action. Regularization effectively constrains weight parameters by adding a corresponding term (regularizer) to the objective loss function. A classical example is Weight Decay [54] that aims to reduce the network complexity by limiting the growth of the weights. This soft constraint is realized by adding a term, proportional to the L_2 norm of the weights to the objective function as the regularizer that penalizes large weight values. WaveQ, on the other hand, uses regularization to push the weights to the quantization levels. For the sake

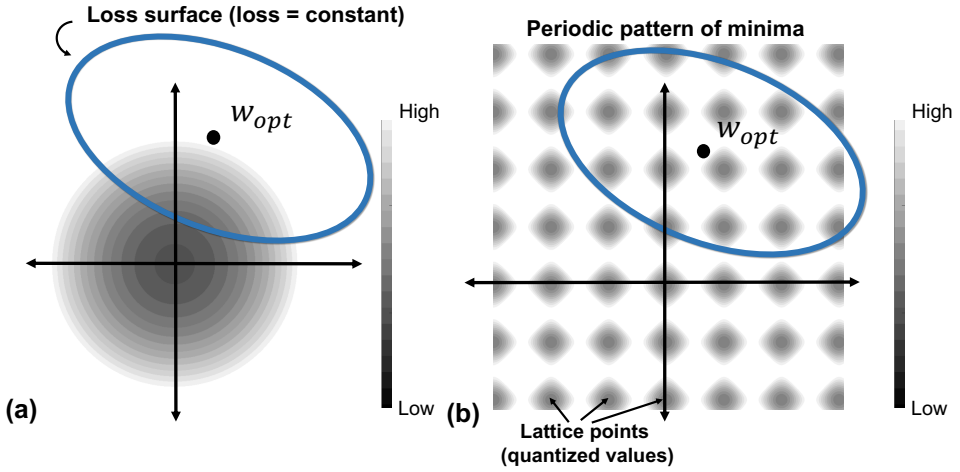

Figure 4.1: Sketch for a hypothetical loss surface (original task loss to be minimized) and an extra regularization term in 2-D weight space: for (a) weight decay, and (b) WaveQ.

of simplicity and clarity, Figure 4.1(a) and (b) illustrate a geometrical sketch for a hypothetical loss surface (original objective function to be minimized) and an extra regularization term in 2-D weight space, respectively. For weight decay regularization (Figure 4.1 (a)), the faded circle shows that as we get closer to the origin, the regularization loss is minimized. The point w_{opt} is the optimum just for the loss function alone and the overall optimum solution is achieved by striking a balance between the original loss term and the regularization loss term. Similarly, Figure 4.1(b) shows a representation of the proposed periodic regularization for a fixed bitwidth β. A periodic pattern of minima pockets are seen surrounding the original optimum point. The objective of the optimization problem is to find the best solution that is the closest to one of those minima pockets where weight values are nearly matching the desired quantization levels, hence the name quantization-friendly.

4.2.2 WaveQ Regularization

The proposed regularizer is formulated in Equation (4.1) where the first term pushes the weights to the quantization levels and the second *correlated* term aims to reduce the bitwidth of each individual layer heterogeneously.

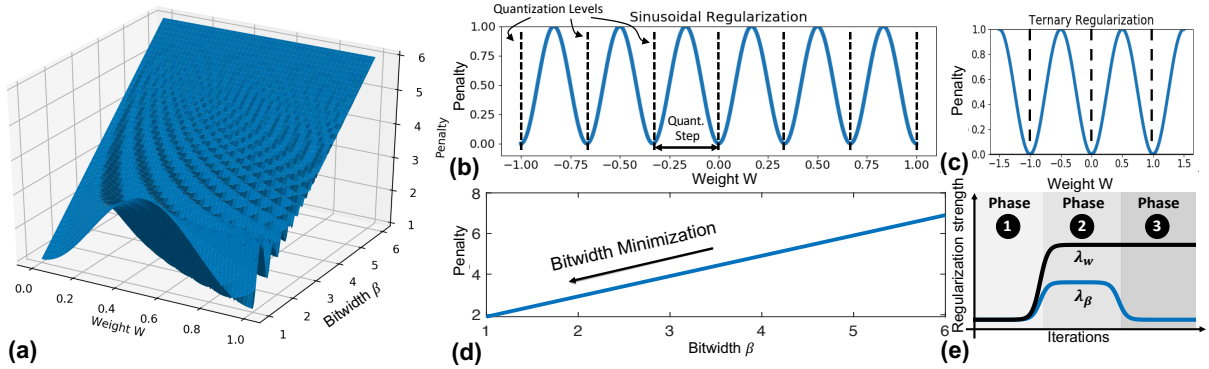

Figure 4.2: (a) 3-D visualization of the proposed generalized objective WaveQ. (b) WaveQ 2-D profile, *w.r.t* weights, adapting for arbitrary bitwidths, (c) example of adapting to ternary quantization. (d) WaveQ 2-D profile *w.r.t* bitwidth. (e) Regularization strengths profiles, λ_w, and λ_β, across training iterations.

$$R(w;\beta) = \underbrace{\lambda_w \sum_i \sum_j \frac{\sin^2\left(\pi w_{ij}(2^{\beta_i}-1)\right)}{2^{\beta_i}}}_{\text{Weights quantization regularization}} + \underbrace{\lambda_\beta \sum_i \beta_i}_{\text{Bitwidth regularization}} \tag{4.1}$$

In Equation (4.1), λ_w is the weights quantization regularization strength which governs how strongly weight quantization errors are penalized, and λ_β is the bitwidth regularization strength. The parameter β_i is proportional to the quantization bitwidth which is elaborated later in this section. Figure 4.2 (a) shows a 3-D visualization of our regularizer, R. Figure 4.2 (b), (c) show a 2-D profile w.r.t weights (w), while (d) shows a 2-D profile w.r.t the bitwidth (β).

Periodic sinusoidal regularization. As shown in Equation (4.1), the first regularization term is based on a periodic function (sinusoidal) that adds a smooth and differentiable term to the original objective, Figure 4.2(b). The periodic regularizer induces a periodic pattern of minima that correspond to the desired quantization levels. Such correspondence is achieved by matching the period to the quantization step ($1/(2^{\beta_i}-1)$) based on a particular number of bits (β_i) for a given layer i.

Learning the sinusoidal period. The parameter β_i in (Equation 4.1) controls the period of the sinusoidal regularizer. Thereby β_i is directly proportional to the actual quantization bitwidth

(b_i) of layer i as follows:

$$b_i = \lceil \beta_i \rceil, \quad \text{and} \quad \alpha_i = 2^{b_i}/2^{\beta_i} \tag{4.2}$$

In Equation (4.2) where $\alpha_i \in \mathbb{R}^+$ is a scaling factor. Note that $b_i \in \mathbb{Z}$ is the only discrete parameter, while $\beta_i \in \mathbb{R}^+$ is a continuous real valued variable, and $\lceil . \rceil$ is the ceiling operator. While the first term in Equation (4.1) is only responsible for promoting quantized weights, the second term ($\lambda_\beta \sum_i \beta_i$) aims to reduce the number of bits for each layer i individually while the overall loss is aiming to maximize accuracy. As such, this term is a soft constraint that yields heterogeneous bitwidths for different layers. The main insight here is that β_i, which also controls the period of the sinusoidal term, is a continuous valued parameter by definition. As such, β_i acts as an ideal optimization objective constraint and a proxy to minimize the actual quantization bitwidth b_i. Therefore, WaveQ avoids the issues of gradient-based optimization for discrete valued parameters. Furthermore, the benefit of learning the sinusoidal period is two-fold. First, it provides a smooth differentiable objective for finding minimal bitwidths. Second, simultaneously learning the scaling factor (α_i) associated with the found bitwidth.

Leveraging the sinusoidal properties, WaveQ learns the following two quantization parameters simultaneously: (1) a per-layer quantization bitwidth (b_i) along with (2) a scaling factor (α_i) through learning the period of the sinusoidal function. Additionally, by exploiting the periodicity, differentiability, and the local convexity profile in sinusoidal functions, WaveQ automatically propels network weights towards values that are inherently closer to quantization levels according to the jointly learned quantizer's parameters b_i, α_i defined in Equation (4.2).

Bounding the gradients. The denominator in the first term of Equation (4.1)

$$\sum_i \sum_j \frac{\sin^2\left(\pi w_{ij}(2^{\beta_i}-1)\right)}{2^{\beta_i}}$$

is used to control the range derivatives of the proposed regularization term with respect to β. This

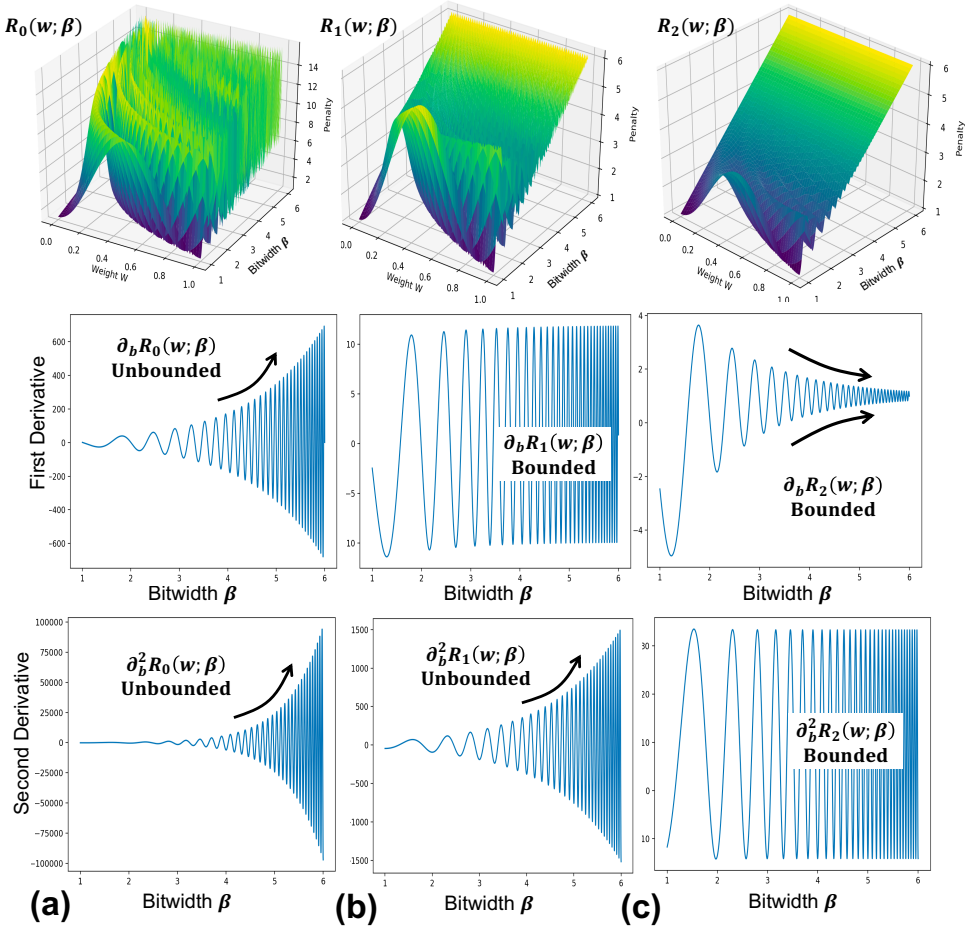

Figure 4.3: Visualization for three variants of the proposed regularization objective using different normalizations and their respective first and second derivatives with respect to β. (a) $R_0(w;\beta)$, (b) $R_1(w;\beta)$, and (c) $R_2(w;\beta)$.

denominator is chosen to limit vanishing and exploding gradients during training. To this end, we compared three variants of equation (4.1) with different normalization defined, for $k = 0$, 1, and 2, as:

$$R_k(w;\beta) = \lambda_w \sum_i \sum_j \frac{\sin^2\left(\pi w_{ij}(2^{\beta_i}-1)\right)}{2^{k\beta_i}} + \lambda_\beta \sum_i \beta_i \tag{4.3}$$

Figure 4.3 (a), (b), (c) provide a visualization on how each of the proposed scaled variants affect the first and second derivatives. For $R_{k=0}$ and $R_{k=2}$, there are regions of vanishing or exploding gradients. Only the regularization $R_{k=1}$ (the proposed one) is free of such issues.

Setting the regularization strengths. The convergence behavior depends on the setting of the regularization strengths λ_w and λ_β. Our proposed objective seeks to learn multiple quantization parameterization in conjunction. As such, the learning process can be portrayed as three phases (Figure 4.2(e)). In Phase (❶), the learning process optimizes for the original task loss E_0. Initially, the small λ_w and λ_β values allow the gradient descent to explore the optimization surface freely. As the training process moves forward, we transition to phase (❷) where the larger λ_w and λ_β gradually engage both the weights quantization regularization and the bitwidth regularization, respectively. Note that, for this to work, the strength of the weights quantization regularization λ_w should be higher than the strength of the bitwidth regularization λ_β such that a bitwidth per layer could be properly evaluated and eventually learned during this phase. After the bitwidth regularizer converges to a bitwidth for each layer, we transition to phase (❸), where we *fix* the learned bitwidths and gradually decay λ_β while we keep λ_w high. The criterion for choosing λ_w and λ_β is to balance the magnitude of regularization loss to be smaller than the magnitude of accuracy loss. The mathematical formula used to generate λ_w and λ_β profiles can be found in the supplementary material. (Figure 8).

4.3 Theoretical Analysis

The results of this section are motivated as follows. Intuitively, we would like to show that the global minima of $E = E_0 + R$ are very close to the minima of E_0 that minimizes R. In other words, we expect to extract among the original solutions, the ones that are most prone to be quantized. To establish such result, we will not consider the minima of $E = E_0 + R$, but the sequence S_n of minima of $E_n = E_0 + \delta_n R$ defined for any sequence δ_n of real positive numbers. The next theorem shows that our intuition holds true, at least asymptotically with n provided $\delta_n \to 0$.

Theorem 4. *Let $E_0, R : \mathbb{R}^n \to [0,\infty)$ be continuous and assume that the set S_{E_0} of the global minima of E_0 is non-empty and compact. As S_{E_0} is compact, we can also define $S_{E_0,R} \subseteq S_{E_0}$ as the set of minima of E_0 which minimizes R. Let δ_n be a sequence of real positive numbers, define $E_n = E_0 + \delta_n R$ and the sequence $S_n = S_{E_n}$ of the global minima of E_n. Then, the following holds true:*

1. *If $\delta_n \to 0$ and $S_n \to S_*$, then $S_* \subseteq S_{E_0,R}$,*
2. *If $\delta_n \to 0$ then there is a subsequence $\delta_{n_k} \to 0$ and a non-empty set $S_* \subseteq S_{E_0,R}$ so that $S_{n_k} \to S_*$,*

where the convergence of sets, denoted by $S_n \to S_*$, is defined as the convergence to 0 of their Haussdorff distance, i.e., $\lim_{n\to\infty} d_H(S_n, S_*) = 0$.

Proof. For the first statement, assume that $S_n \to S_*$. We wish to show that $S_* \subseteq S_{E_0,R}$. Assume that x_n is a sequence of global minima of $F + \delta_n G$ converging to x_*. It suffices to show that $x_* \in S_{E_0,R}$. First let us observe that $x_* \in S_{E_0}$. Indeed, let

$$\lambda = \inf_{x\in\mathbb{R}^n} E_0(x)$$

and assume that $x \in S_{E_0}$. Then,

$$\lambda \leq E_0(x_n) \leq (E_0 + \delta_n R)(x_n) \leq (E_0 + \delta_n R)(x) = \underbrace{\lambda + \delta_n R(x)}_{\to \lambda}.$$

Thus, since E_0 is continuous and $x_n \to x_*$ we have that $E_0(x_*) = \lambda$ which implies $x_* \in S_{E_0}$. Next, define

$$\mu = \inf_{x \in S_{E_0}} R(x).$$

Let $\hat{x} \in S_{E_0,R}$ so that $R(\hat{x}) = \mu$. Now observe that, by the minimality of x_n we have that

$$\lambda + \delta_n \mu = (E_0 + \delta_n R)(\hat{x}) \geq (E_0 + \delta_n R)(x_n) \geq \lambda + \delta_n R(x_n)$$

Thus, $R(x_n) \leq \mu$ for all n. Since R is continuous and $x_n \to x_*$ we have that $R(x_*) \leq \mu$ which implies that $R(x_*) = \mu$ since $x_* \in S_{E_0}$. Thus, $x_* \in S_{E_0,R}$. The second statement follows from the standard theory of Hausdorff distance on compact metric spaces and the first statement. □

Theorem 4 implies that by decreasing the strength of R, one recovers the subset of the original solutions that achives the smallest quantization loss. In practice, we are not interested in global minima, and we should not decrease much the strength of R. In our context, Theorem 4 should then be understood as a proof of concept on why the proposed approach leads the expected result. Experiments carried out in the next section will support this claim. For the interested reader, we provide a more detailed version of the above analysis in the supplementary material.

4.4 Experimental Results

To demonstrate the effectiveness of WaveQ, we evaluated it on several deep neural networks with different Image Classification datasets (CIFAR10, SVHN, and ImageNet), and one Transformer-based network, which is the big Transformer model from [70] on the IWSLT14 German-English dataset [1]. We provide results for two different types of quantization. First, we show quantization results for *learned* heterogeneous bitwidths using WaveQ and we provide

Table 4.1: Comparison with state-of-the-art quantization methods on ImageNet. The " W/A " values are the bitwidths of weights/activations.

W/A Quantization	Benchmark		AlexNet		ResNet-18		MobileNet-V2	
	Method	Assignment	Top-1	Top-5	Top-1	Top-5	Top-1	Top-5
W32/A32	Full Precision	Homogenous	57.1	80.2	70.1	89.5	71.8	90.3
W3/A3	PACT	Homogenous	55.6	-	68.1	88.2	-	-
	LQ-Nets	Homogenous	-	-	68.2	87.9	-	-
	DSQ	Homogenous	-	-	68.7	-	-	-
	DoReFa	Homogenous	54.1	75.1	67.9	87.5	58.3	78.1
W3/A3	**DoReFa + WaveQ**	**Preset Homogenous**	**55.8**	**77.2**	**68.9**	**89.9**	**60.4**	**83.1**
	Improvement		0.2%↑	2.1%↑	0.2%↑	1.7%↑	2.1%↑	5.0%↑
W4/A4	PACT	Homogenous	55.7	-	69.2	89.0	61.4	83.7
	LQ-Nets	Homogenous	-	-	69.3	88.8	-	-
	DSQ	Homogenous	-	-	69.6	-	64.8	-
	WRPN	Homogenous	54.9	75.4	68.8	88.1	64.3	84.5
	DoReFa	Homogenous	55.5	76.3	69.1	88.5	64.6	85.1
W4/A4	**DoReFa + WaveQ**	**Preset Homogenous**	**56.2**	**79.2**	**69.8**	**89.1**	**65.4**	**85.5**
	Improvement		0.5%↑	2.9%↑	0.2%↑	0.1%↑	0.6%↑	0.4%↑
W(***Learn***)/A4	**DoReFa + WaveQ**	**Learned Heterogenous**	**56.5**	**79.8**	**70.0**	**89.3**	**65.8**	**85.8**
			W3.85		**W3.57**		**W3.95**	
	Improvement		0.3%↑	0.6%↑	0.2%↑	0.2%↑	0.4%↑	0.3%↑
	Energy Saving		**2.08x**		**1.24x**		**1.78x**	

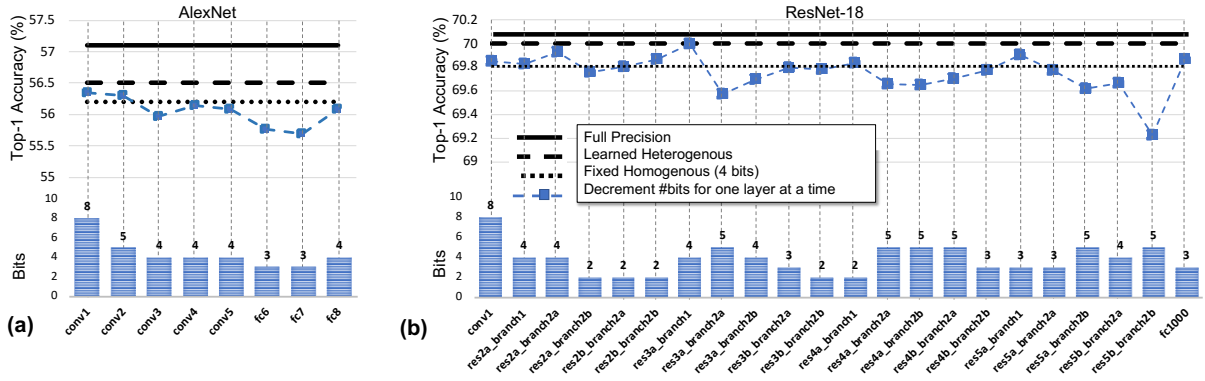

Figure 4.4: Quantization bitwidth assignments across layers. (a) AlexNet (average bitwidth = 3.85 bits). (b) ResNet-18 (average bitwidth = 3.57 bits)

different analyses to asses the quality of these learned bitwidth assignments. Second, we further provide results assuming a *preset* homogeneous bitwidth assignment as a special setting of WaveQ. This is, in some cases, a practical assumption that might stem from particular hardware requirements or constraints. Table 4.1 provides a summary of the evaluated networks and datasets for both learned heterogeneous bitwidths, and the special case of training preset homogeneous bitwidth assignments. We compare our proposed WaveQ method with PACT [15], LQ-Nets [99], DSQ [33], and DoReFa, which are current state-of-the-art methods that show results with homogeneous 3-, and 4-bit weight/activation quantization for various networks (AlexNet, ResNet-18, and MobileNet).

Experimental setup. We implemented WaveQ on top of DoReFa inside Distiller [103], an open source framework for quantization by Intel that implements various quantization techniques. The reported accuracies for DoReFa and WRPN are with the built-in implementations in Distiller, which may not exactly match the accuracies reported in their respective papers. However, an independent implementation from a major company provides an unbiased foundation for comparison. We quantize all convolution and fully connected layers, except for the first and last layers which use 8-bits. This assumption is commensurate with the previous techniques.

4.4.1 Learned Heterogeneous Bitwidths

As for quantizing both weights and activations, Table 4.1 shows that incorporating WaveQ into the quantized training process yields best accuracy results outperforming PACT, LQ-Net, DSQ, and DoReFa with significant margins. Furthermore, it can be seen that the learned heterogeneous bitwidths yield better accuracy as compared to the preset 4-bit homogeneous assignments, with lower, on average, bitwidh (3.85, 3.57, and 3.95 bits for AlexNet, ResNet-18, and MobileNet, respectively). Figure 4.4(a),(b) (bottom bar graphs) show the learned heterogeneous weight bitwidths over layers for AlexNet and ResNet-18, respectively. As seen, WaveQ parametric regularization yields a spectrum of varying bitwidth assignments to the layers which vary from 2 bits to 8 bits with an irregular pattern. These results demonstrate that the proposed regularization, WaveQ, automatically distinguishes different layers and their varying importance with respect to accuracy while learning their respective bitwidths. To assess the quality of these bitwidths assignments, we conduct a sensitivity analysis on the relatively big networks (see next subsection).

Benefits of heterogeneous quantization. Figure 4.4(a),(b) (top graphs) show various comparisons and sensitivity results for learned heterogeneous bitwidth assignments for bigger networks (AlexNet and ResNet-18). It is infeasible to enumerate these networks' respective quantization spaces. Compared to 4-bit homogeneous quantization, learned heterogeneous assignments achieve better accuracy with lower, on average, bitwidth 3.85 bits for AlexNet and 3.57 bits for ResNet-18. This demonstrates that a homogeneous (uniform) assignment of the bits is not always the desired choice to minimize accuracy loss. Furthermore, Figure 4.4 also shows that decrementing the learned bitwidth for any single layer at a time results in 0.44% and 0.24% average reduction in accuracy for AlexNet and ResNet-18, respectively. The dotted blue line with ■ markers shows how differently decrementing the bitwidth of various individual layers affect the accuracy. This trend further demonstrates the effectiveness of learning with WaveQ to find the lowest bitwidth that maximizes the accuracy.

Energy savings. To demonstrate the energy savings of the solutions found by WaveQ, we

Figure 4.5: Accuracies of different networks using plain WRPN, plain DoReFa and DoReFa + WaveQ on homogeneous weight quantization.

W/A Quantization	Benchmark	SimpleNet on CIFAR10	ResNet-20 on CIFAR10	VGG-11 On CIFAR10	SVHN-8 on SVHN
	Method	Top-1 Accuracy (%)			
W32/A32	Full Precision	74.53	93.3	94.13	96.47
W3/A32	WRPN	63.44	80.28	78.56	79.36
	DoReFa	65.13	81.57	78.78	81.45
	DoReFa + WaveQ	**73.65**	**92.52**	**93.18**	**95.32**
	Improvement	8.52%↑	11%↑	14.4%↑	13.9%↑
W4/A32	WRPN	68.23	88.16	85.07	89.24
	DoReFa	70.75	89.24	86.98	89.56
	DoReFa + WaveQ	**74.14**	**93.01**	**93.96**	**96.12**
	Improvement	3.39%↑	3.77%↑	6.98%↑	6.56%↑
W5/A32	WRPN	71.17	92.11	91.10	90.84
	DoReFa	72.41	92.24	91.68	92.56
	DoReFa + WaveQ	**74.45**	**93.13**	**94.11**	**96.42**
	Improvement	2.04%↑	0.89%↑	2.43%↑	3.86%↑

evaluate it on Stripes [49], a custom accelerator designed for DNNs, which exploits bit-serial computation to support flexible bitwidths for DNN operations. As shown in Table 4.1, the reduction in the bitwidth, on average, leads to 77.5% reduction in the energy consumed during the execution of these networks.

4.4.2 Preset Homogenous Bitwidth Quantization

We also consider a *preset* homogeneous bitwidth quantization which can also be supported by WaveQ under special settings where we fix β (to a preset bitwidth). Hence, only the first regularization term is engaged for propelling the weights to the quantization levels.

Table 4.5 shows accuracies of different networks (SimpleNet-5, ResNet-20, VGG-11, and SVHN-8) using plain WRPN, plain DoReFa and DoReFa + WaveQ for 3, 4, and 5 bitwdiths. As depicted, the results concretely show the effect of incorporating WaveQ into existing quantized training techniques and how it outperforms the previously reported accuracies.

Weight distributions during training. Figure 4.6 shows the evolution of weights distri-

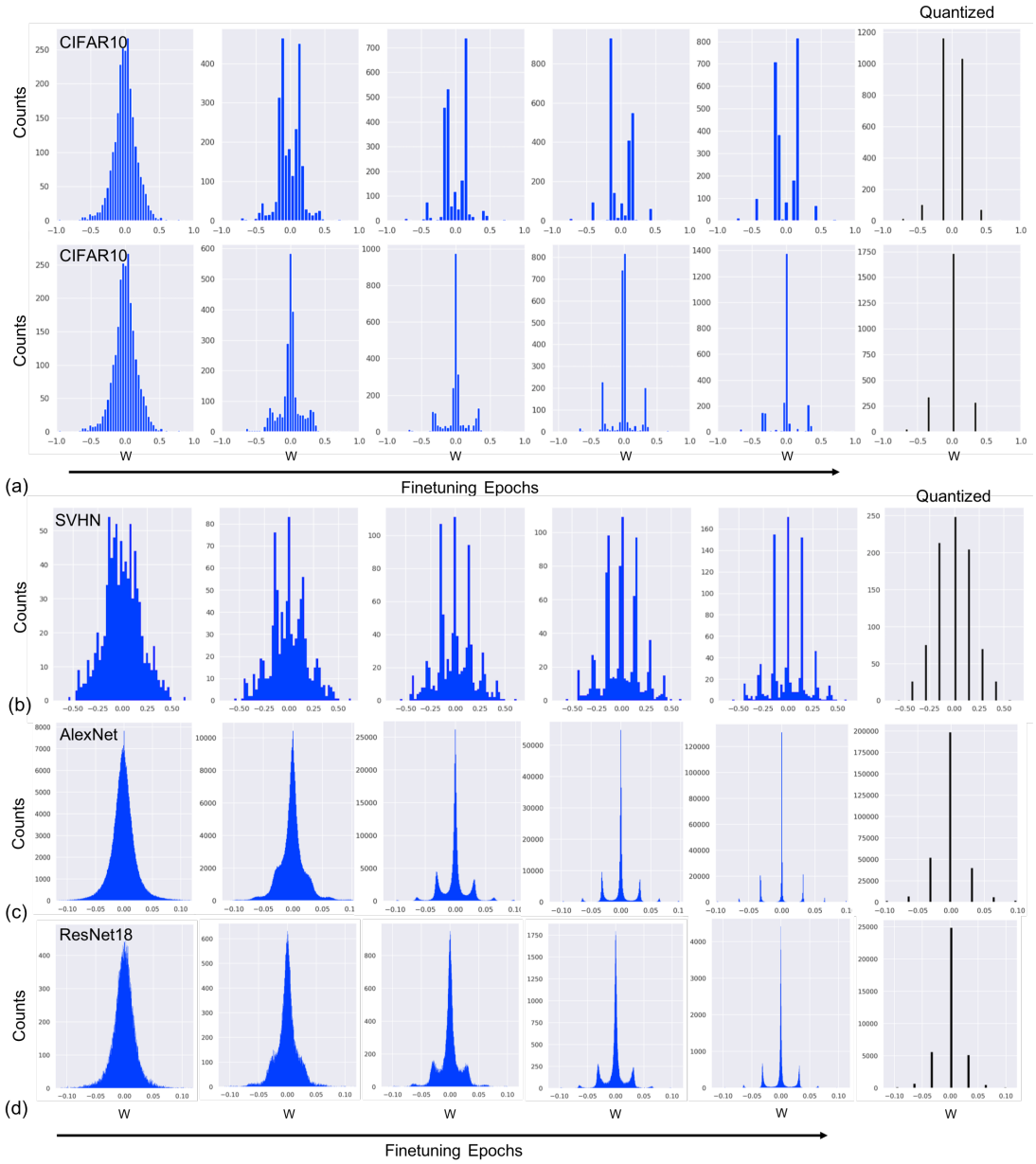

Figure 4.6: Evolution of weight distributions over training epochs at different layers and bitwidths for different networks. (a) CIFAR10, (b) SVHN, (c) AlexNet, (d) ResNet18.

Table 4.2: Performance of WaveQ for quantizing Transformers.

Bitwidth	BLEU score	
	Unregularized Training (without WaveQ)	**Regularized Training (with WaveQ)**
Full precision	34.93	
7-bit	34.79	**34.83 (0.04↑)**
6-bit	34.39	**34.53 (0.14↑)**
5-bit	32.74	**33.20 (0.46↑)**

butions over fine-tuning epochs for different layers of CIFAR10, SVHN, AlexNet, and ResNet-18 networks. The high-precision weights form clusters and gradually converge around the quantization centroids as regularization loss is minimized along with the main accuracy loss.

4.4.3 WaveQ for Transformer Quantization

Transformers (encoder-decoder architectures) have been shown to achieve best results for NLP tasks including machine translation [87] and automatic speech recognition [66]. A Transformer layer relies on a key-value self-attention mechanism for learning relationships between distant concepts, rather than relying on recurrent connections and memory cells. Herein, we extend the application of WaveQ regularization to improve the accuracy of deeply quantized (below 8 bits) Transformer models. We run our experiments on IWSLT 2014 German-English (DE-EN) dataset. We use the Transformer model implemented in the `fairseq-py` toolkit [69]. All experiments are based on the big transformer model with 6 blocks in the encoder and decoder networks. Table 4.2 shows the effect of applying WaveQ regularization into the training process for 5, 6, and 7-bit quantization on the final accuracy (BLEU score). WaveQ consistently improves the BLEU score of quantized models at various quantization bitwidths (7-5 bits). Moreover, higher improvements are obtained at lower bitwidths.

4.5 Discussion

We conduct an experiment that uses WaveQ for training from scratch. For the sake of clarity, we are considering in this experiment the case of preset bitwidth assignments (i.e., $\lambda_\beta = 0$). Figure 4.7-Row(I)-Col(I) shows weight trajectories without WaveQ as a point of reference. Row(II)-Col(I) shows the weight trajectories when WaveQ is used with a constant λ_w.

As the Figure illustrates, using a constant λ_w results in the weights being stuck in a region close to their initialization, (i.e., quantization objective dominates the accuracy objective). However, if we dynamically change the λ_w following the exponential curve in Figure 4.7-Row(III)-Col(I)) during the from-scratch training, the weights no longer get stuck. Instead, the weights traverse the space (*i.e., jump from wave to wave*) as illustrated in Figure 4.7-Cols(II) and (III) for CIFAR and SVHN, respectively. In these two columns, Rows (I), (II), (III), correspond to quantization with 3, 4, 5 bits, respectively. citepInitially, the smaller λ_w values allow the gradient descent to explore the optimization surface freely, as the training process moves forward, the larger λ_w gradually engages the sinusoidal regularizer, and eventually pushes the weights close to the quantization levels. Further convergence analysis is provided in the supplementary material.

4.6 Related Work

This research lies at the intersection of (1) quantized training algorithms and (2) techniques that discover bitwidth for quantization. The following diuscusses the most related works in both directions. In contrast, WaveQ modifies the loss function of the training to simultaneously *learn* the period of an adaptive sinusoidal regularizer through the same stochastic gradient descent that trains the network. The differentiability of the adaptive sinusoidal regularizer enables simultaneously learning both the bitwidths and pushing the weight values to the quantization levels. As such, WaveQ can be used as a complementary method to some of these efforts, which is demonstrated by experiments with both DoReFa-Net [101] and WRPN [65]. Our preliminary efforts [23]

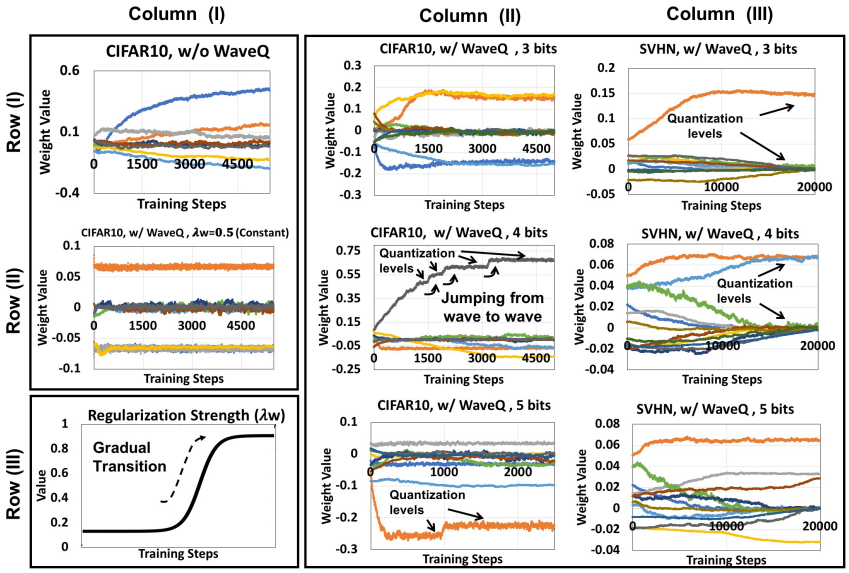

Figure 4.7: Weight trajectories. The 10 colored lines in each plot denote the trajectory of 10 different weights.

and another concurrent work [67] use a sinusoidal regularization to push the weights closer to the quantization levels. However, neither of these two works make the period a differentiable parameter nor find bitwidths during training.

Quantized training algorithms There have been several techniques [65, 101, 102] that train a neural network in a quantized domain after the bitwidth of the layers is determined manually. DoReFa-Net [101] uses straight through estimator [10] for quantization and extends it for any arbitrary k bit quantization in weights, activations, and gradients. WRPN [65] is training algorithm that compensates for the reduced precision by increasing the number of filter maps in a layer (doubling or tripling). TTQ [102] quantizes the weights to ternary values by using per layer scaling coefficients learnt during training. These scaling coefficients are used to scale the weights during inference. PACT [15] proposes a technique for quantizing activations by introducing an activation clipping parameter α. This parameter (α) is used to represent the clipping level in the activation function and is learned via back-propagation during training. More recently, VNQ [3] uses a variational Bayesian approach for quantizing neural network weights during training.

Loss-aware weight quantization. Recent works pursued loss-aware minimization ap-

proaches for quantization. [41] and [40] developed approximate solutions using proximal Newton algorithm to minimize the loss function directly under the constraints of low bitwidth weights. One effort [16] proposed to learn the quantization of DNNs through a regularization term of the mean-squared-quantization error. LQ-Net [99] proposes to jointly train the network and its quantizers. DSQ [33] employs a series of tanh functions to gradually approximate the staircase function for low-bit quantization (e.g., sign for 1-bit case), and meanwhile keeps the smoothness for easy gradient calculation. Although some of these techniques use regularization to guide the process of quantized training, none explores the use of adaptive sinusoidal regularizers for quantization. Moreover, unlike WaveQ, these techniques do not find the bitwidth for quantizing the layers.

Techniques for discovering quantization bitwidths. A recent line of research focused on methods which can also find the optimal quantization parameters, e.g., the bitwidth, the stepsize, in parallel to the network weights. Recent work [94] based on ADMM [11] runs a binary search to minimize the total square quantization error in order to decide the quantization levels for the layers. Most recently, [86] proposed to indirectly learn quantizer's parameters via Straight Through Estimator (STE) [10] based approach. In a similar vein, [27] has proposed to learn the quantization mapping for each layer in a deep network by approximating the gradient to the quantizer step size that is sensitive to quantized state transitions. On another side, recent works [25, 91] proposed a reinforcement learning based approach to find an optimal bitwidth assignment policy.

Quantizing Transformers. FullyQT [74] uses a bucketing based uniform quantization proposed by QSGD [4] and extends it to Tranformers. Q8BERT [97] quantizes all the GEMM (General Matrix Multiply) operations to 8 bit by adding an additional term for quantization loss during training, which is calculated based on the rounding effect of floating point values [84]. WaveQ, however, uses a sinusoidal regularizer to automatically push the weights towards the quantization levels.

4.7 Conclusion

This work devised WaveQ that casts the two problems of finding layer bitwidth and quantized weights as a gradient-based optimization through parametric sinusoidal regularization. WaveQ provides significant improvements over the state-of-the-art and is even applicable to the Transformers.

4.8 Broader Impact

While DNNs are pushing the boundaries in many applications, they have also become increasingly hard to train and deploy as they grow in size because of both computational intensity and large memory footprint. Quantization reduces the compute intensity as well as the memory footprint of these networks, thereby making them more pervasive to devices with limited compute capability. At the same time, this can also lead to making DNNs more accessible to people who cannot afford high performance hardware or pay for resources on the cloud. On the other hand, the effects of quantization on underrepresented groups is not well studied, and there is a possibility that quantizing DNNs would have disparate impacts on the accuracy of different subgroups of data contributors, similar to what has been shown for differentially private training of DNNs [7]. This would require further study of the fairness and societal impacts of quantization.

Acknowledgment. Chapter 4, in part, has been submitted for publication of the material as it appears in *International Conference on Learning Representations*. Ahmed T. Elthakeb, Prannoy Pilligundla, Fatemeh Mireshghallah, Tarek Elgindi, Charles-Alban Deledalle, Hadi Esmaeilzadeh, 2021; and in part, contains a re-organized reprint of the material as it appears in *ICML Workshop on Understanding and Improving Generalization in Deep Learning*. Ahmed T. Elthakeb, Prannoy Pilligundla, Fatemeh Mireshghallah, Hadi Esmaeilzadeh, 2019. The dissertation author was the primary investigator and author of both papers.

Chapter 5

Food for Thought on DNN Quantization

5.1 ΣΔ-BNN: Sigma-Delta Approach for Deep Neural Networks Binarization

5.1.1 Introduction

Deep Neural Networks (DNNs) have made waves across a variety of domains, from voice assistants to medicine and self-driving cars [2, 13, 36, 38, 56, 57]. DNN compute efficiency, however, has become a major constraint in unlocking further applications and capabilities, as these models require rather massive amounts of computation even for a single inquiry. One approach to reduce the intensity of the DNN computation is to reduce the complexity of each operation. To this end, quantization (reducing the bitwidth representation) of neural networks provides a path forward as it reduces the bitwidth of the operations as well as the data footprint [43, 47, 83]. Albeit alluring, quantization can lead to significant accuracy loss if not employed with diligence. Years of research and development has yielded current levels of accuracy, which is the driving force behind the wide applicability of DNNs nowadays. To prudently preserve this valuable feature of DNNs, accuracy, while benefiting from quantization the following two fundamental problems

need to be addressed. (1) Learning techniques need to be developed that can train or finetune quantized neural networks given a level of quantization for each layer [23]. (2) Algorithms need to be designed that can discover the appropriate level of quantization for each layer while considering the accuracy [25]. This proposed work takes on the first challenge as there are inspiring efforts that have developed techniques for quantized training.

The most extreme form of network quantization is binarization. Binarization is a 1-bit quantization where parameters can take only one of two possible values. Generally -1 and $+1$ have been used for these two values (or $-\alpha$ and $+\alpha$ when scaling is considered per certain granularity). As such, BNNs are deep neural networks that use binary values for weights and/or activations, instead of full precision values. In this preliminary study, we focus on weight binarization only. In Binary-Weight Networks, the filters are approximated with binary values resulting in $32\times$ memory saving. When weight values are binary, convolutions can be estimated by only addition and subtraction (without multiplication), resulting in $\sim 2\times$ speed up. As such, binary-weight approximations of large DNNs can fit into the memory of even small, portable devices while maintaining the same level of accuracy.

In this proposal, we introduce simple, efficient, and accurate binary global approximations to the weights of DNNs. Our binarization method aims at finding the best approximations of the convolutions using the principle of sigma-delta . The key insight of the the proposed method is that it changes the traditional local optimization objective (i.e., approximating individual weights: w_i) to a more global objective that is approximating dot products (i.e., : $\Sigma_i w_i x_j$). For the first time, the proposed method for binarization is a non-backpropagation based method, i.e., no gradient computations are involved. We evaluate our approach on three different datasets: MNIST, CIFAR10, and SVHN. The proposed method leverages the inherent accumulation in neural networks computations to cleverly average a large number of rounding steps. To the best of our knowledge this work is the first attempt to provide a method for binarizing weights of neural networks using a non-gradient based approach.

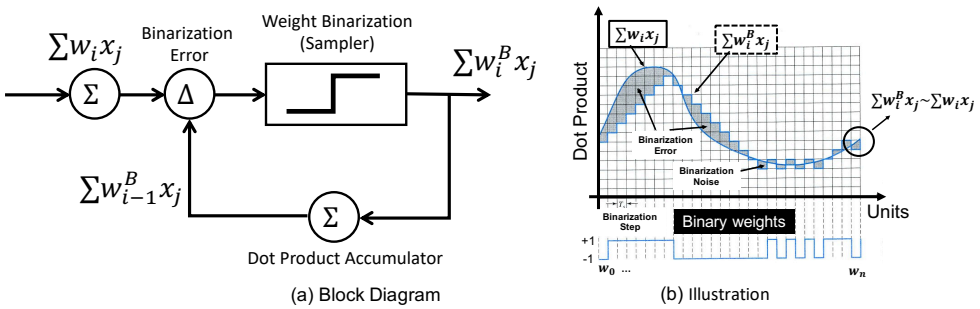

Figure 5.1: Overview of Sigma-Delta Approach for Neural Networks Binarization.

5.1.2 Method

Overview. Sigma-Delta ($\Sigma\Delta$) is a method for encoding analog signals into digital signals as commonly used in an analog-to-digital converter (ADC) [92]. In a conventional ADC, an analog signal is sampled with a sampling frequency and subsequently quantized in a multi-level quantizer into a digital signal. This process introduces quantization error noise. The first step in a delta-sigma modulation is delta modulation. In delta modulation the change in the signal (its delta) is encoded, rather than the absolute value. The result is a stream of pulses, as opposed to a stream of numbers as is the case with pulse code modulation (PCM). In delta-sigma modulation , accuracy of the modulation is improved by passing the digital output through a 1-bit DAC and adding (sigma) the resulting analog signal to the input signal (the signal before delta modulation), thereby reducing the error introduced by the delta modulation.

Estimating binary weights. Assume $\hat{W}$ and $\hat{W}^B$ vectors representing full precision, and corresponding approximate binary weights respectively. Conventionally, the optimal estimation is formulated as: $w_i^B = \alpha B_i$, and we solve the following optimization problem:

$$J(B,\alpha) = \|w_i - \alpha B_i\|^2$$

$$\alpha^*, B^* = \underset{\alpha,B}{arg\,min}[J(B,\alpha)]$$

This optimization can be solved by assigning $B_i = +1$ if $w_i \geq 0$, and $B_i = -1$ if $w_i \leq 0$. Therefore

the optimal solution is:

$$B_i^* = sign(w_i),$$

$$\alpha^* = \frac{1}{n} \left\| \hat{W} \right\|_{l1}$$

where n is the number of weights in vector $\hat{W}$. Thus, the optimal estimation of a binary weight filter can be simply achieved by taking the sign of weight values. The optimal scaling factor is the average of absolute weight values [76].

Sigma-Delta for deep neural networks binarization. Herein, we propose ΣΔ-BNN as a method to binarize the weights of neural networks. The key insight of the the proposed method is that instead of pursing the predominant approaches of finding the best binary approximation of individual weights per a particular granularity (e.g., network, layer, channel), it pursues a more wholistic approach by approximating the dot product between weights (w_i) and activations (x_j). ΣΔ-BNN leverages the inherent accumulation process in neural networks computations to cleverly average a large number of rounding steps. Such incremental averaging of many rounding steps leads to the least possible binarization error. As such, ΣΔ-BNN changes the traditional local optimization objective (i.e., approximating individual weights: w_i) to a more global objective that is approximating dot products. Thus, ΣΔ-BNN objective becomes:

$$P(\hat{W}^B) = \left\| \Sigma_j (\Sigma_i (w_i x_j) - \Sigma_i (w_i^B x_j)) \right\|$$

$$\hat{W}^{B^*} = \underset{\hat{W}^B}{arg\,min} [P(\hat{W}^B)]$$

The proposed method is an iterative method; yet, is a non-backpropagation based method. Figure 5.1 (a), (b) illustrate the proposed algorithm. For each training sample, the algorithm steps through the elements of dot product one by one. In each step, the objective is to binarize (i.e., generate +1 or -1) such that the binarization error (the difference between the full precision dot product value so far and the binarized dot product) is gradually minimized. In other words,

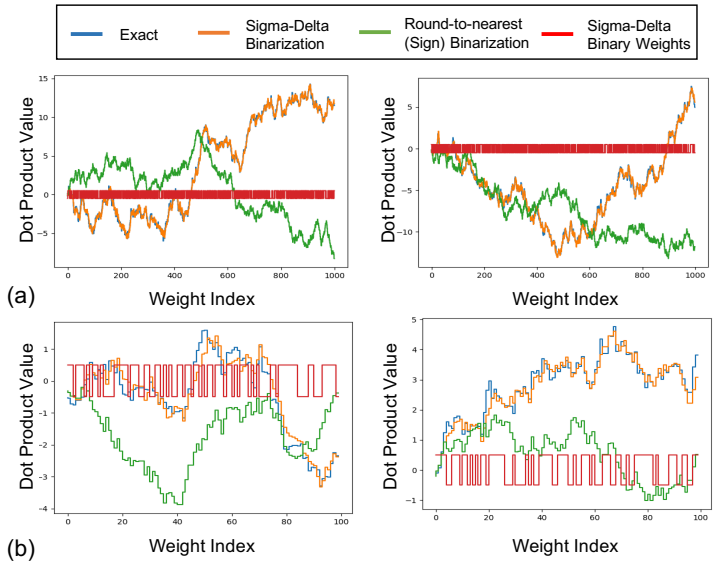

Figure 5.2: Waveforms comparison for sigma-delta binarization (ΣΔ-BNN) of LeNet layers on MNIST dataset.

Method	MNIST	CIFAR-10	SVHN
Round-to-nearest Binarization	85.30%	33.68%	45.21%
Binary Connect (det.)	1.29%	9.90%	2.30%
Binary Connect (stoch.)	1.18%	8.27%	2.15%
Sigma-Delta BNN (ours)	**0.95%**	**7.43%**	**2.06%**

Table 5.1: Classification test error rates of DNNs trained on MNIST, CIFAR10, and SVHN using different binarization methods.

if the binarization error is ≥ 0, the next product element of the dot product should be negative. In contrast, if the binarization error is ≤ 0, the next product element of the dot product should be rather positive. This can be visualized by looking at Figure 5.2. The the orange waveform (sigma-delta binarization) is consistently trying to track/follow the blue waveform (the exact computation using full precision weights). Thus, the ΣΔ-BNN generates a binary code to reverse the binarization error direction, so that it can consistently keep it as small as possible.

5.1.3 Evaluation

Figure 5.2 shows sigma-delta binarization for approximating the dot products of different layers of LeNet on MNIST dataset. Figure 5.2 (b) shows a zoomed-in version of (a) to better distinguish different waveforms across 100 weight indices. Each figure shows four waveforms. Exact dot product waveform (i.e., using full precision weights) is shown in blue. Round-to-nearest (*Sign*) binarization is shown in green. Sigma-delta binarization is overlaid in orange. The resultant ΣΔ-BNN binary weights (as -1, and +1 pulses) are shown in red. As it can be clearly seen, ΣΔ-BNN provides a pretty accurate and consistent approximation of the dot product throughout the vector indices with a significant margin as compared to the *Sign* binarization. Table 5.1 shows that ΣΔ-BNN outperforms conventional binarization schemes including rounding-to-nearest and binary conn. with a significant margin.

5.1.4 Related Work

Training algorithms for quantized neural networks. There have been several techniques [65, 101, 102] that train a neural network in a quantized domain after the bitwidth of the layers is determined manually. DoReFa-Net [101] trains quantized convolutional neural networks with parameter gradients which are stochastically quantized to low bitwidth numbers before they are propagated to the convolution layers. [65] introduces a scheme to train networks from scratch using reduced-precision activations by decreasing the precision of both activations and weights and increasing the number of filter maps in a layer. DCQ [22] proposes an unorthodox method to train quantized neural networks. The proposed approach utilizes knowledge distillation through teacher-student paradigm in a novel setting that exploits the feature extraction capability of DNNs for higher-accuracy quantization. This divide and conquer strategy makes the training of each student section possible in isolation while all these independently trained sections are later stitched together to form the equivalent fully quantized network. [102] performs the training

phase of the network in full precision, but for inference uses ternary weight assignments. For this assignment, the weights are quantized using two scaling factors which are learned during training phase. PACT [15] introduces a quantization scheme for activations, where the variable α is the clipping level and is determined through a gradient descent based method. SinReQ [23] proposes a novel sinusoidal regularization for deep quantized training. The proposed regularization is realized by adding a periodic function (sinusoidal regularizer) to the original objective function. By exploiting the inherent periodicity and local convexity profile in sinusoidal functions, SinReQ automatically propel weights towards target quantization levels during conventional training. Leveraging the sinusoidal properties further, [24] extended SinReQ to learn the quantization bitwidth during gradient-based training process. The key insight is that they leverage the observation that sinusoidal period is a continuous valued parameter. As such, the sinusoidal period serves as an ideal optimization objective and a proxy to minimize the actual quantization bitwidth, which avoids the issues of gradient-based optimization for discrete valued parameters.

Ternary and binary neural networks. These works are the most relevant to our approach. Extensive work, [43,58,75] focuses on binarized neural networks, which impose accuracy loss but reduce the bitwidth to lowest possible level. In BinaryNet [42], an extreme case, a method is proposed for training binarized neural networks which reduce memory size, accesses and computation intensity at the cost of accuracy. XNOR-Net [75] leverages binary operations (such as XNOR) to approximate convolution in binarized neural networks. Another work [58] introduces ternary-weight networks, in which the weights are quantized to -1, 0, +1 values by minimizing the Euclidian distance between full-precision weights and their ternary assigned values. However, most of these methods rely on handcrafted optimization techniques and ad-hoc manipulation of the underlying network architecture that are not easily extendable for new networks. For example, multiplying the outputs with a scale factor to recover the dynamic range (i.e., the weights effectively become -w and w, where w is the average of the absolute values of the weights in the filter), keeping the first and last layers at 32-bit floating point precision,

and performing normalization before convolution to reduce the dynamic range of the activations. Moreover, these methods [58, 75] are customized for a single bitwidth, binary only or ternary only in the case of [75] or [58], respectively, which imposes a blunt constraint on inherently different layers with different requirements resulting in sub-optimal quantization solutions. ΣΔ-BNN aims to utilize the levels between binary and 8 bits to avoid loss of accuracy while offering automation.

ΣΔ-BNN is an orthogonal technique with different objective that is providing an efficient binary approximations to the weights of the neural network given a pre-trained model. Although ΣΔ-BNN is considered an iterative method, the algorithm of finding the binary levels does not involve any gradients computations nor propagation, in contrast to the predominant back-propagation based training methods.

5.1.5 Conclusion

In this proposal, we utilize sigma-delta approach to propose an efficient binary approximation (ΣΔ-BNN) to the weights of deep neural networks. We provide a proof-of-concept demonstration of the proposed method to show its efficacy in binarizing weights of neural networks. ΣΔ-BNN changes the local optimization objective of minimizing binarization error across individual weights to a global objective that is approximating the dot product between weights and activations. To the best of our knowledge, this paper is the first attempt to provide a simple, efficient, yet accurate method for binarizing weights of neural networks using a non-gradient based approach. Preliminary results on different benchmarks demonstrated the efficiency of the proposed method as an initial step towards efficient encoding methods for binary inference of DNNs.

Acknowledgement. Chapter 5, in part, contains a re-organized reprint of the material as it appears in *MLArchSys Workshop, ISCA*. Ahmed T. Elthakeb, Hadi Esmaeilzadeh, 2020. The dissertation author was the primary investigator and author of this paper.

Bibliography

[1] IWSLT'14 german to english dataset.

[2] Imagenet classification with deep convolutional neural.

[3] ACHTERHOLD, J., KÖHLER, J. M., SCHMEINK, A., AND GENEWEIN, T. Variational network quantization. In *6th ICLR* (2018).

[4] ALISTARH, D., GRUBIC, D., LI, J., TOMIOKA, R., AND VOJNOVIC, M. Qsgd: Communication-optimal stochastic gradient descent, with applications to training neural networks.

[5] ALLEN-ZHU, Z., LI, Y., AND LIANG, Y. Learning and generalization in overparameterized neural networks, going beyond two layers. In *Advances in Neural Information Processing Systems 32*. 2019, pp. 6155–6166.

[6] AMAZON. Automatic model tuning, 2018.

[7] BAGDASARYAN, E., POURSAEED, O., AND SHMATIKOV, V. Differential privacy has disparate impact on model accuracy. In *Advances in Neural Information Processing Systems* (2019), pp. 15453–15462.

[8] BAKER, B., GUPTA, O., NAIK, N., AND RASKAR, R. Designing Neural Network Architectures using Reinforcement Learning.

[9] BARTLETT, P. L., FOSTER, D. J., AND TELGARSKY, M. J. Spectrally-normalized margin bounds for neural networks. In *Advances in Neural Information Processing Systems* (2017), pp. 6240–6249.

[10] BENGIO, Y., LÉONARD, N., AND COURVILLE, A. C. Estimating or propagating gradients through stochastic neurons for conditional computation. *CoRR abs/1308.3432* (2013).

[11] BOYD, S. P., PARIKH, N., CHU, E., PELEATO, B., AND ECKSTEIN, J. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Found. Trends Mach. Learn. 3*, 1 (2011), 1–122.

[12] BUCILA, C., CARUANA, R., AND NICULESCU-MIZIL, A. Model compression. In *Proceedings of the Twelfth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Philadelphia, PA, USA, August 20-23, 2006* (2006), T. Eliassi-Rad, L. H. Ungar, M. Craven, and D. Gunopulos, Eds., ACM, pp. 535–541.

[13] CHEN, T., DU, Z., SUN, N., WANG, J., WU, C., CHEN, Y., AND TEMAM, O. Diannao: a small-footprint high-throughput accelerator for ubiquitous machine-learning. In *ASPLOS* (2014).

[14] CHEN, T., MOREAU, T., JIANG, Z., SHEN, H., YAN, E. Q., WANG, L., HU, Y., CEZE, L., GUESTRIN, C., AND KRISHNAMURTHY, A. Tvm: End-to-end optimization stack for deep learning. *CoRR abs/1802.04799* (2017).

[15] CHOI, J., WANG, Z., VENKATARAMANI, S., CHUANG, P. I.-J., SRINIVASAN, V., AND GOPALAKRISHNAN, K. Pact: Parameterized clipping activation for quantized neural networks. *CoRR abs/1805.06085* (2018).

[16] CHOI, Y., EL-KHAMY, M., AND LEE, J. Learning low precision deep neural networks through regularization. *CoRR abs/1809.00095* (2018).

[17] CHOROMANSKA, A., HENAFF, M., MATHIEU, M., AROUS, G. B., AND LECUN, Y. The Loss Surfaces of Multilayer Networks. In *Artificial Intelligence and Statistics* (2015).

[18] CISSE, M., BOJANOWSKI, P., GRAVE, E., DAUPHIN, Y., AND USUNIER, N. Parseval networks: Improving robustness to adversarial examples. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70* (2017), JMLR. org, pp. 854–863.

[19] COLLOBERT, R., WESTON, J., BOTTOU, L., KARLEN, M., KAVUKCUOGLU, K., AND KUKSA, P. P. Natural language processing (almost) from scratch. *Journal of Machine Learning Research 12* (2011), 2493–2537.

[20] COURBARIAUX, M., BENGIO, Y., AND DAVID, J. Binaryconnect: Training deep neural networks with binary weights during propagations. In *NIPS* (2015), pp. 3123–3131.

[21] ELSKEN, T., METZEN, J. H., AND HUTTER, F. Neural architecture search: A survey. *J. Mach. Learn. Res. 20* (2019), 55:1–55:21.

[22] ELTHAKEB, A. T., PILLIGUNDLA, P., AND ESMAEILZADEH, H. Divide and Conquer: Leveraging intermediate feature representations for quantized training of neural networks. *International Conference on Machine Learning (ICML) Workshop on Understanding and Improving Generalization in Deep Learning* (2019).

[23] ELTHAKEB, A. T., PILLIGUNDLA, P., AND ESMAEILZADEH, H. SinReQ: Generalized sinusoidal regularization for low-bitwidth deep quantized training. *International Conference on Machine Learning (ICML) Workshop on Understanding and Improving Generalization in Deep Learning* (2019).

[24] ELTHAKEB, A. T., PILLIGUNDLA, P., MIRESHGHALLAH, F., ELGINDI, T., DELEDALLE, C.-A., AND ESMAEILZADEH, H. WaveQ: Gradient-based deep quantization of neural networks through sinusoidal adaptive regularization. *arXiv preprint arXiv:2003.00146* (2020).

[25] ELTHAKEB, A. T., PILLIGUNDLA, P., MIRESHGHALLAH, F., YAZDANBAKHSH, A., AND ESMAEILZADEH, H. Releq: A reinforcement learning approach for deep quantization of neural networks. *ML for Systems Workshop, NeurIPS* (2018).

[26] ELTHAKEB, A. T., PILLIGUNDLA, P., MIRESHGHALLAH, F., YAZDANBAKHSH, A., AND ESMAEILZADEH, H. Releq: A reinforcement learning approach for deep quantization of neural networks. *CoRR abs/1811.01704* (November 5, 2018).

[27] ESSER, S. K., MCKINSTRY, J. L., BABLANI, D., APPUSWAMY, R., AND MODHA, D. S. Learned step size quantization. *8th ICLR, 2020 abs/1902.08153* (2019).

[28] FEURER, M., KLEIN, A., EGGENSPERGER, K., SPRINGENBERG, J. T., BLUM, M., AND HUTTER, F. Auto-sklearn: Efficient and robust automated machine learning. In *Automated Machine Learning - Methods, Systems, Challenges*. 2019, pp. 113–134.

[29] FROMM, J., PATEL, S., AND PHILIPOSE, M. Heterogeneous bitwidth binarization in convolutional neural networks. In *NeurIPS*. (2018), pp. 4010–4019.

[30] GAO, M., PU, J., YANG, X., HOROWITZ, M., AND KOZYRAKIS, C. E. TETRIS: Scalable and Efficient Neural Network Acceleration with 3D Memory. In *ASPLOS* (2017).

[31] GHODRATI, S., SHARMA, H., YOUNG, C., KIM, N. S., AND ESMAEILZADEH, H. Bit-parallel vector composability for neural acceleration. *arXiv preprint arXiv:2004.05333* (2020).

[32] GOLOVIN, D., SOLNIK, B., MOITRA, S., KOCHANSKI, G., KARRO, J., AND SCULLEY, D. Google vizier: A service for black-box optimization. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Halifax, NS, Canada, August 13 - 17, 2017* (2017), pp. 1487–1495.

[33] GONG, R., LIU, X., JIANG, S., LI, T., HU, P., LIN, J., YU, F., AND YAN, J. Differentiable soft quantization: Bridging full-precision and low-bit neural networks. *CoRR abs/1908.05033* (2019).

[34] GOUK, H., FRANK, E., PFAHRINGER, B., AND CREE, M. Regularisation of neural networks by enforcing lipschitz continuity. *arXiv preprint arXiv:1804.04368* (2018).

[35] GUPTA, S., AGRAWAL, A., GOPALAKRISHNAN, K., AND NARAYANAN, P. Deep learning with limited numerical precision. In *ICML* (2015), pp. 1737–1746.

[36] HAUSWALD, J., LAURENZANO, M., ZHANG, Y., LI, C., ROVINSKI, A., KHURANA, A., DRESLINSKI, R. G., MUDGE, T. N., PETRUCCI, V., TANG, L., AND MARS, J. Sirius: An open end-to-end voice and vision personal assistant and its implications for future warehouse scale computers. In *ASPLOS* (2015).

[37] HE, Y., LIN, J., LIU, Z., WANG, H., LI, L.-J., AND HAN, S. AMC: AutoML for Model Compression and Acceleration on Mobile Devices. In *ECCV* (2018).

[38] HINTON, G. E., OSINDERO, S., AND TEH, Y. W. A fast learning algorithm for deep belief nets. *Neural Computation 18* (2006), 1527–1554.

[39] HINTON, G. E., VINYALS, O., AND DEAN, J. Distilling the knowledge in a neural network. *CoRR abs/1503.02531* (2015).

[40] HOU, L., AND KWOK, J. T. Loss-aware weight quantization of deep networks. In *6th ICLR* (2018).

[41] HOU, L., YAO, Q., AND KWOK, J. T. Loss-aware binarization of deep networks. In *5th ICLR* (2017).

[42] HUBARA, I., COURBARIAUX, M., SOUDRY, D., EL-YANIV, R., AND BENGIO, Y. Binarized Neural Networks. In *NIPS*. 2016.

[43] HUBARA, I., COURBARIAUX, M., SOUDRY, D., EL-YANIV, R., AND BENGIO, Y. Quantized Neural Networks: Training Neural Networks with Low Precision Weights and Activations. *J. Mach. Learn. Res.* (2017).

[44] HUBARA, I., COURBARIAUX, M., SOUDRY, D., EL-YANIV, R., AND BENGIO, Y. Quantized neural networks: Training neural networks with low precision weights and activations. *JMLR 18* (2017), 187:1–187:30.

[45] HUTTER, F., KOTTHOFF, L., AND VANSCHOREN, J., Eds. *Automated Machine Learning: Methods, Systems, Challenges*. Springer, 2018. In press, available at http://automl.org/book.

[46] JADERBERG, M., CZARNECKI, W. M., OSINDERO, S., VINYALS, O., GRAVES, A., SILVER, D., AND KAVUKCUOGLU, K. Decoupled neural interfaces using synthetic gradients. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017* (2017), D. Precup and Y. W. Teh, Eds., vol. 70 of *Proceedings of Machine Learning Research*, PMLR, pp. 1627–1635.

[47] JUDD, P., ALBERICIO, J., HETHERINGTON, T. H., AAMODT, T. M., AND MOSHOVOS, A. Stripes: Bit-serial deep neural network computing. *49th MICRO* (2016), 1–12.

[48] JUDD, P., ALBERICIO, J., HETHERINGTON, T. H., AAMODT, T. M., AND MOSHOVOS, A. Stripes: Bit-serial deep neural network computing. *2016 49th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)* (2016), 1–12.

[49] JUDD, P., ALBERICIO, J., HETHERINGTON, T. H., AAMODT, T. M., AND MOSHOVOS, A. Stripes: Bit-serial deep neural network computing. In *49th Annual IEEE/ACM International Symposium on Microarchitecture, MICRO 2016, Taipei, Taiwan, October 15-19, 2016* (2016), IEEE Computer Society, pp. 19:1–19:12.

[50] KRISHNAMOORTHI, R. Quantizing deep convolutional networks for efficient inference: A whitepaper. *CoRR abs/1806.08342* (2018).

[51] KRISHNAMOORTHI, R. Quantizing deep convolutional networks for efficient inference: A whitepaper, 2018.

[52] KRIZHEVSKY, A., SUTSKEVER, I., AND HINTON, G. E. Imagenet classification with deep convolutional neural networks. *Commun. ACM 60* (2012), 84–90.

[53] KRIZHEVSKY, A., SUTSKEVER, I., AND HINTON, G. E. Imagenet classification with deep convolutional neural networks. In *NIPS* (2012), pp. 1106–1114.

[54] KROGH, A., AND HERTZ, J. A. A simple weight decay can improve generalization. In *Proceedings of the 4th International Conference on Neural Information Processing Systems* (San Francisco, CA, USA, 1991), NIPS'91, Morgan Kaufmann Publishers Inc., p. 950–957.

[55] LECUN, Y., BENGIO, Y., AND HINTON, G. E. Deep learning. *Nature 521*, 7553 (2015), 436–444.

[56] LECUN, Y., BOSER, B. E., DENKER, J. S., HENDERSON, D., HOWARD, R. E., HUBBARD, W. E., AND JACKEL, L. D. Backpropagation applied to handwritten zip code recognition. *Neural Computation 1* (1989), 541–551.

[57] LEE, H., GROSSE, R. B., RANGANATH, R., AND NG, A. Y. Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In *ICML* (2009).

[58] LI, F., AND LIU, B. Ternary Weight Networks. *CoRR abs/1605.04711* (2016).

[59] LI, H., XU, Z., TAYLOR, G., STUDER, C., AND GOLDSTEIN, T. Visualizing the Loss Landscape of Neural Nets. In *NIPS* (2018).

[60] LI, F.F., LI, J. Cloud automl: Making ai accessible to every business, 2018.

[61] MENDOZA, H., KLEIN, A., FEURER, M., SPRINGENBERG, J. T., URBAN, M., BURKART, M., DIPPEL, M., LINDAUER, M., AND HUTTER, F. Towards automatically-tuned deep neural networks. In *Automated Machine Learning - Methods, Systems, Challenges*. 2019, pp. 135–149.

[62] MICIKEVICIUS, P., NARANG, S., ALBEN, J., DIAMOS, G. F., ELSEN, E., GARCÍA, D., GINSBURG, B., HOUSTON, M., KUCHAIEV, O., VENKATESH, G., AND WU, H. Mixed precision training. *CoRR abs/1710.03740* (2017).

[63] MISHRA, A., AND MARR, D. Apprentice: Using Knowledge Distillation Techniques To Improve Low-Precision Network Accuracy. In *ICLR* (2018).

[64] MISHRA, A. K., AND MARR, D. Apprentice: Using knowledge distillation techniques to improve low-precision network accuracy. *CoRR abs/1711.05852* (2017).

[65] MISHRA, A. K., NURVITADHI, E., COOK, J. J., AND MARR, D. WRPN: Wide Reduced-Precision Networks. In *ICLR* (2018).

[66] MOHAMED, A., OKHONKO, D., AND ZETTLEMOYER, L. Transformers with convolutional context for asr, 2019.

[67] NAUMOV, M., DIRIL, U., PARK, J., RAY, B., JABLONSKI, J., AND TULLOCH, A. On periodic functions as regularizers for quantization of neural networks. *CoRR abs/1811.09862* (2018).

[68] NVIDIA. Automatic mixed precision for nvidia tensor core architecture in tensorflow.

[69] OTT, M., EDUNOV, S., BAEVSKI, A., FAN, A., GROSS, S., NG, N., GRANGIER, D., AND AULI, M. fairseq: A fast, extensible toolkit for sequence modeling. In *Proceedings of NAACL-HLT 2019: Demonstrations* (2019).

[70] OTT, M., EDUNOV, S., GRANGIER, D., AND AULI, M. Scaling neural machine translation. *Proceedings of the Third Conference on Machine Translation: Research Papers* (2018).

[71] PEDREGOSA, F., VAROQUAUX, G., GRAMFORT, A., MICHEL, V., THIRION, B., GRISEL, O., BLONDEL, M., PRETTENHOFER, P., WEISS, R., DUBOURG, V., VANDERPLAS, J., PASSOS, A., COURNAPEAU, D., BRUCHER, M., PERROT, M., AND DUCHESNAY, E. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research 12* (2011), 2825–2830.

[72] PHAM, H., GUAN, M. Y., ZOPH, B., LE, Q. V., AND DEAN, J. Efficient neural architecture search via parameter sharing. In *ICML* (2018), pp. 4092–4101.

[73] POLINO, A., PASCANU, R., AND ALISTARH, D. Model compression via distillation and quantization. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings* (2018), OpenReview.net.

[74] PRATO, G., CHARLAIX, E., AND REZAGHOLIZADEH, M. Fully quantized transformer for improved translation. *ArXiv abs/1910.10485* (2019).

[75] RASTEGARI, M., ORDONEZ, V., REDMON, J., AND FARHADI, A. XNOR-Net: ImageNet Classification Using Binary Convolutional Neural Networks. In *ECCV* (2016).

[76] RASTEGARI, M., ORDONEZ, V., REDMON, J., AND FARHADI, A. XNOR-Net: ImageNet Classification Using Binary Convolutional Neural Networks. In *ECCV* (2016), pp. 525–542.

[77] REN, S., HE, K., GIRSHICK, R. B., AND SUN, J. Faster r-cnn: Towards real-time object detection with region proposal networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence 39* (2015), 1137–1149.

[78] ROMERO, A., BALLAS, N., KAHOU, S. E., CHASSANG, A., GATTA, C., AND BENGIO, Y. Fitnets: Hints for thin deep nets. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings* (2015), Y. Bengio and Y. LeCun, Eds.

[79] RUMELHART, D. E., HINTON, G. E., AND WILLIAMS, R. J. Learning internal representations by error propagation. In *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Volume 1: Foundations*. MIT Press, Cambridge, MA, 1986, pp. 318–362.

[80] SAKR, C., KIM, Y., AND SHANBHAG, N. Analytical guarantees on numerical precision of deep neural networks. In *Proceedings of the 34th International Conference on Machine Learning* (International Convention Centre, Sydney, Australia, 06–11 Aug 2017), D. Precup and Y. W. Teh, Eds., vol. 70 of *Proceedings of Machine Learning Research*, PMLR, pp. 3007–3016.

[81] SAMRAGH, M., JAVAHERIPI, M., AND KOUSHANFAR, F. Encodeep: Realizing bit-flexible encoding for deep neural networks. *ACM Transactions on Embedded Computing Systems (TECS)* (2020).

[82] SCHULMAN, J., WOLSKI, F., DHARIWAL, P., RADFORD, A., AND KLIMOV, O. Proximal Policy Optimization Algorithms. *arXiv preprint arXiv:1707.06347* (2017).

[83] SHARMA, H., PARK, J., SUDA, N., LAI, L., CHAU, B., CHANDRA, V., AND ESMAEILZADEH, H. Bit fusion: Bit-level dynamically composable architecture for accelerating deep neural network. *ISCA* (2018), 764–775.

[84] SHAW, P., USZKOREIT, J., AND VASWANI, A. Self-attention with relative position representations. In *NAACL-HLT* (2018).

[85] SZEGEDY, C., ZAREMBA, W., SUTSKEVER, I., BRUNA, J., ERHAN, D., GOODFELLOW, I., AND FERGUS, R. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199* (2013).

[86] UHLICH, S., MAUCH, L., YOSHIYAMA, K., CARDINAUX, F., GARCÍA, J. A., TIEDEMANN, S., KEMP, T., AND NAKAMURA, A. Mixed precision dnns: All you need is a good parametrization.

[87] VASWANI, A., SHAZEER, N., PARMAR, N., USZKOREIT, J., JONES, L., GOMEZ, A. N., KAISER, L., AND POLOSUKHIN, I. Attention is all you need, 2017.

[88] VIRMAUX, A., AND SCAMAN, K. Lipschitz regularity of deep neural networks: analysis and efficient estimation. In *Advances in Neural Information Processing Systems* (2018), pp. 3835–3844.

[89] WANG, J., BAO, W., SUN, L., ZHU, X., CAO, B., AND YU, P. S. Private model compression via knowledge distillation. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019.* (2019), AAAI Press, pp. 1190–1197.

[90] WANG, K., LIU, Z., LIN, Y., LIN, J., AND HAN, S. HAQ: hardware-aware automated quantization with mixed precision. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019* (2019), pp. 8612–8620.

[91] WANG, K., LIU, Z., LIN, Y., LIN, J., AND HAN, S. HAQ: Hardware-Aware Automated Quantization. *arXiv preprint arXiv:1811.08886* (November 21, 2018).

[92] WIKIPEDIA. Sigma-delta modulation.

[93] WU, B., WANG, Y., ZHANG, P., TIAN, Y., VAJDA, P., AND KEUTZER, K. Mixed precision quantization of convnets via differentiable neural architecture search. *CoRR abs/1812.00090* (2018).

[94] YE, S., ZHANG, T., ZHANG, K., LI, J., XIE, J., LIANG, Y., LIU, S., LIN, X., AND WANG, Y. A unified framework of dnn weight pruning and weight clustering/quantization using admm. *CoRR abs/1811.01907* (2018).

[95] YIM, J., JOO, D., BAE, J., AND KIM, J. A gift from knowledge distillation: Fast optimization, network minimization and transfer learning. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017* (2017), IEEE Computer Society, pp. 7130–7138.

[96] YING, C., KLEIN, A., CHRISTIANSEN, E., REAL, E., MURPHY, K., AND HUTTER, F. Nas-bench-101: Towards reproducible neural architecture search. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA* (2019), pp. 7105–7114.

[97] ZAFRIR, O., BOUDOUKH, G., IZSAK, P., AND WASSERBLAT, M. Q8bert: Quantized 8bit bert. *ArXiv abs/1910.06188* (2019).

[98] ZHANG, C., BENGIO, S., AND SINGER, Y. Are all layers created equal? *CoRR abs/1902.01996* (2019).

[99] ZHANG, D., YANG, J., YE, D., AND HUA, G. Lq-nets: Learned quantization for highly accurate and compact deep neural networks. In *ECCV* (2018), pp. 373–390.

[100] ZHOU, A., YAO, A., GUO, Y., XU, L., AND CHEN, Y. Incremental network quantization: Towards lossless cnns with low-precision weights. In *5th ICLR* (2017).

[101] ZHOU, S., NI, Z., ZHOU, X., WEN, H., WU, Y., AND ZOU, Y. DoReFa-Net: Training Low Bitwidth Convolutional Neural Networks with Low Bitwidth Gradients. *CoRR* (2016).

[102] ZHU, C., HAN, S., MAO, H., AND DALLY, W. J. Trained Ternary Quantization. In *ICLR* (2017).

[103] ZMORA, N., JACOB, G., AND NOVIK, G. Neural network distiller, June 2018.

[104] ZOPH, B., AND LE, Q. V. Neural Architecture Search with Reinforcement Learning. In *ICLR* (2017).

[105] ZOU, D., BALAN, R., AND SINGH, M. On lipschitz bounds of general convolutional neural networks. *IEEE Transactions on Information Theory* (2019).