



저작자표시-비영리-변경금지 2.0 대한민국

이용자는 아래의 조건을 따르는 경우에 한하여 자유롭게

- 이 저작물을 복제, 배포, 전송, 전시, 공연 및 방송할 수 있습니다.

다음과 같은 조건을 따라야 합니다:



저작자표시. 귀하는 원저작자를 표시하여야 합니다.



비영리. 귀하는 이 저작물을 영리 목적으로 이용할 수 없습니다.



변경금지. 귀하는 이 저작물을 개작, 변형 또는 가공할 수 없습니다.

- 귀하는, 이 저작물의 재이용이나 배포의 경우, 이 저작물에 적용된 이용허락조건을 명확하게 나타내어야 합니다.
- 저작권자로부터 별도의 허가를 받으면 이러한 조건들은 적용되지 않습니다.

저작권법에 따른 이용자의 권리는 위의 내용에 의하여 영향을 받지 않습니다.

이것은 [이용허락규약\(Legal Code\)](#)을 이해하기 쉽게 요약한 것입니다.

[Disclaimer](#)

Ph.D. DISSERTATION

Characterization and Optimization of Quantized Deep Neural Networks

양자화된 깊은 신경망의 특성 분석 및 최적화

BY

YOONHO BOO

AUGUST 2020

DEPARTMENT OF ELECTRICAL AND
COMPUTER ENGINEERING
COLLEGE OF ENGINEERING
SEOUL NATIONAL UNIVERSITY

Ph.D. DISSERTATION

Characterization and Optimization of Quantized Deep Neural Networks

양자화된 깊은 신경망의 특성 분석 및 최적화

BY

YOONHO BOO

AUGUST 2020

DEPARTMENT OF ELECTRICAL AND
COMPUTER ENGINEERING
COLLEGE OF ENGINEERING
SEOUL NATIONAL UNIVERSITY

Characterization and Optimization of Quantized Deep Neural Networks

양자화된 깊은 신경망의 특성 분석 및 최적화

지도교수 성원용

이 논문을 공학박사 학위논문으로 제출함

2020년 8월

서울대학교 대학원

전기 정보 공학부

부윤호

부윤호의 공학박사 학위 논문을 인준함

2020년 8월

위원장:	이정우
부위원장:	성원용
위원:	윤성로
위원:	유승주
위원:	최정욱

Abstract

Deep neural networks (DNNs) have achieved impressive performance on various machine learning tasks. However, performance improvements are usually accompanied by increased network complexity incurring vast arithmetic operations and memory accesses. In addition, the recent increase in demand for utilizing DNNs in resource-limited devices leads to a plethora of explorations in model compression and acceleration. Among them, network quantization is one of the most cost-efficient implementation methods for DNNs. Network quantization converts the precision of parameters and signals from 32-bit floating-point to 8, 4, or 2-bit fixed-point precision. The weight quantization can directly compress DNNs by reducing the representation levels of the parameters. Activation outputs can also be quantized to reduce the computational costs and working memory footprint. However, severe quantization degrades the performance of the network. Many previous studies focused on developing optimization methods for the quantization of given models without considering the effects of the quantization on DNNs. Therefore, extreme simulation is required to obtain quantization precision that maintains performance on different models or datasets.

In this dissertation, we attempt to measure the per-parameter capacity of DNN models and interpret the results to obtain insights on the optimum quantization of parameters. The uniform random vectors are sampled and used for training generic forms of fully connected DNNs, convolutional neural networks (CNNs), and recurrent neural networks (RNNs). We conduct memorization and classification tests to study the effects of the parameters' number and precision on the performance. The model and the per-parameter capacities are assessed by measuring the mutual information between the input and the classified output. To get insight for parameter quantization when performing real tasks, the training and the test performances are compared.

In addition, we analyze and demonstrate that quantization noise of weight and

activation are disparate in inference. Synthesized data is designed to visualize the effects of weight and activation quantization. The results indicate that deeper models are more prone to activation quantization, while wider models improve the resiliency to both weight and activation quantization. Considering the characteristics of the quantization errors, we propose a holistic approach for the optimization of QDNNs, which contains QDNN training methods as well as quantization-friendly architecture design.

Based on the observation that the activation quantization induces noised prediction, we propose the Stochastic Precision Ensemble training for QDNNs (SPEQ). The SPEQ is teacher-student learning, but the teacher and the student share the model parameters. We obtain the teacher's soft labels by changing the bit-precision of the activation stochastically at each layer of the forward-pass computation. The student model is trained with these soft labels to reduce the activation quantization noise. Instead of the KL-divergence, the cosine-distance loss is employed for the KD training. Since the teacher model changes continuously by random bit-precision assignment, it exploits the effect of stochastic ensemble KD. The SPEQ method outperforms various tasks, such as image classification, question-answering, and transfer learning without requiring cumbersome teacher networks.

keywords: Quantized Deep Neural Networks, Memorization Capacity, Quantization Error Visualization, Fixed-Point Optimization, Stochastic Precision Ensemble Training for Quantization (SPEQ)

student number: 2016-20915

Contents

Abstract	i
Contents	iii
List of Tables	vi
List of Figures	ix
1 Introduction	1
1.1 Quantization of Deep Neural Networks	1
1.1.1 Weight and Activation Quantization on Deep Neural Networks	2
1.1.2 Analysis of Quantized Deep Neural Networks	3
1.2 Scope of the Dissertation	4
1.2.1 Characterization of Quantization Errors	4
1.2.2 Optimization of Quantized Deep Neural Networks	6
2 Memorization Capacity of Deep Neural Networks under Parameter Quan-	
tization	8
2.1 Introduction	8
2.2 Related Works and Backgrounds	10
2.2.1 Neural Network Capacity	10
2.2.2 Fixed-Point Deep Neural Networks	11
2.3 Network Capacity Measurements of DNNs	11
2.3.1 Capacity Measurements on a Memorization Task	11

2.3.2	Network Quantization Method	13
2.3.3	Network Quantization and Parameter Capacity	14
2.4	Experimental Results on Capacity of Floating-point DNNs	15
2.4.1	Capacity of FCDNNs	15
2.4.2	Capacity of CNNs	19
2.4.3	Capacity of RNNs	19
2.5	Experimental Results of Parameter Quantization	21
2.5.1	Capacity under Parameter Quantization	21
2.5.2	Quantization Experiments on CIFAR-10 Dataset	23
2.5.3	Quantization Experiments on Shuffled CIFAR-10 Dataset	25
2.6	Concluding Remarks	28
3	Characterization and Holistic Optimization of Quantized Deep Neural Networks	30
3.1	Introduction	30
3.2	Backgrounds	32
3.2.1	Related Works on Network Quantization	32
3.2.2	Revisit of QDNN Optimization	33
3.3	Visualization of Quantization Errors using Synthetic Dataset	34
3.3.1	Synthetic Dataset Generation	34
3.3.2	Results on Synthetic Dataset	37
3.4	QDNN Optimization with Architectural Transformation and Improved Training	39
3.4.1	Architecture Transformation for Improved Robustness to Quan- tization	40
3.4.2	Cyclical Learning Rate Scheduling for Improved Generalization	41
3.4.3	Regularization for Limiting the Activation Noise Amplification	42

3.5	Experimental Results	42
3.5.1	Visualizing the Effects of Quantization on the Segmentation Task	42
3.5.2	The Width and Depth Effects on QDNNs	44
3.5.3	QDNN Architecture Selection under the Parameter Constraint	49
3.5.4	Results of Training Methods on QDNNs	51
3.6	Concluding Remarks	53
4	Parameter Shared Stochastic Precision Knowledge Distillation for Quantized Deep Neural Networks	55
4.1	Introduction	55
4.2	Background and Related Works	58
4.2.1	Quantization of Deep Neural Networks	58
4.2.2	Knowledge Distillation for Quantization	59
4.3	Stochastic Precision Ensemble Training for QDNNs	60
4.3.1	Quantization Method	60
4.3.2	Stochastic Precision Self-Distillation with Model Sharing	61
4.3.3	Stochastic Ensemble Learning	63
4.3.4	Cosine Similarity Learning	65
4.4	Experimental Results	70
4.4.1	Experiment Setup	70
4.4.2	Results on CIFAR-10 and CIFAR-100 Datasets	70
4.4.3	Results on ImageNet Dataset	73
4.4.4	Results on Transfer Learning	76
4.5	Concluding Remarks	78
5	Conclusion	80
	Abstract (In Korean)	97
	감사의 글	99

List of Tables

2.1	Accuracies on CIFAR-10 dataset according to the channel width multiplier of ResNet20	26
2.2	Full precision performance of ResNet20 (14) according to the shuffling probability, p	26
3.1	Quantization performance of various model structures on CIFAR-10 testset. ‘ L ’, ‘ D_{init} ’, and ‘ B ’ represent the number of layers, initial channel dimension, and number of blocks, respectively.	48
3.2	Performance comparison of 2-bit ResNet on ImageNet validation set. ‘ L ’ and ‘ D_{init} ’ represent the number of layers and initial channel dimension, respectively. ‘ B ’ is the stack of the residual blocks which is separated by the convolution layers with the stride of 2.	50
3.3	Performance improvements of quantized MobileNetV2 on the PASCAL VOC 2012 validation set. n_W and n_A represent the precision of the weights and activations, respectively.	52
3.4	Performance in terms of mIOU on the PASCAL VOC 2012 validation set when both weights and activations are quantized.	53
3.5	CIFAR-10 test accuracy (%) improvements when retrained with L_{Lip} and fine-tuned using CLR.	53

4.1	Test accuracy (%) in higher precision on the quantized model. Two ResNet20 models are trained with 2-bit weight or activation quantization using the CIFAR-100 dataset, then bit-precision higher than 2-bit is used for inference. This reveals interesting phenomena that weight and activation quantization affects the model differently.	56
4.2	Comparison of the test accuracy of 2-bit ResNet20 on CIFAR-10 according to the loss function for KD. The cosine similarity loss (CS-Loss) suits better than the KL-divergence loss (KL-Loss) for the proposed SPEQ method. Average test accuracy of 5 repeated experiments is reported with the standard deviation.	66
4.3	2-bit ResNet20 test accuracy according to the quantization probability for the stochastic path, u . Average test accuracy of 5 repeated experiments is reported with the standard deviation.	71
4.4	Test accuracy (%) of quantized CNNs on CIFAR-10 and CIFAR-100 datasets. ‘F’ denotes the floating-point precision.	72
4.5	Top-1 accuracy of 2-bit activation quantized CNNs on the ImageNet dataset.	73
4.6	Top-1 accuracy on the ImageNet dataset when both weights and activations are quantized to 2 bits.	74
4.7	2-bit ImageNet quantization Top-1 accuracy compared with other KD applied QDNNs. Flops and run-times are measured for a single update with the batch size of 64.	75
4.8	The performance improvement with the SPEQ scheme on the question-answering task. The BERT model is quantized and fine-tuned using the SQuAD1.1 dataset. Results are evaluated using the SQuAD1.1 dev dataset.	77

4.9	Performance comparison on the transfer learning according to the training method and the feature extractor. Values in the parentheses are the standard deviation of 5 training results.	78
-----	---	----

List of Figures

2.1	Memorization performances according to the (a) width and (b) depth of the FCDNN.	16
2.2	(a) Mutual information according to the number of inputs N . (b) The relationship between the number of parameters and the capacity of networks in FCDNNs and CNNs.	18
2.3	Quantization effect when networks use the (a) full capacity and (b) half capacity.	20
2.4	Quantization performance according to the number of data to train. (a) FCDNN, (b) CNN, and (c) RNN.	22
2.5	Weight quantization results according to the number of parameters on (a) VGG and (b) ResNet structures by changing the width and depth. All results are reported on CIFAR-10 test dataset after retraining except “Float train”.	24
2.6	CIFAR-10 train accuracy drop by the quantization on ResNet20 (14) according to the shuffling probability, p . The accuracy drop means the difference between the accuracy of 8-bit and target precision.	27

3.1	Illustrations of the dataset and prediction results from FCDNNs. (a) The synthetic dataset used to train FCDNNs. (b) An example of the evaluation results when the model is too small to learn to the data distribution. (c, d) Examples of large models. The values in parentheses are the accuracy for the correct answer.	35
3.2	Prediction results of QDNN as the width and depth increase. (W2) 2-bit weights. (A2) 2-bit activations. (W2A2) 2-bit weights and activations.	36
3.3	Prediction results of QDNNs with residual connection. (W2) 2-bit weights. (A2) 2-bit activations. (W2A2) 2-bit weights and activations.	38
3.4	Types of residual blocks. (a) basic block, (b) pre-activation block, and (c) depthwise block. QA and QRA denote the activation quantization operations.	40
3.5	Learning rate scale factor along training iterations. The red dashed box indicates a single cycle of CLR scheduling.	41
3.6	Visualization of quantization errors on the PASCAL VOC segmentation benchmark. ‘W’ and ‘A’ are abbreviations for the weight and activation, respectively. Activation outputs are retained in floating-point precision on weight quantized model, and vice-versa.	43
3.7	Performance of quantized ResNet on the CIFAR-10 testset according to the (a) width of the ResNet20 and (b) depth of the ResNet. Legends represent the precision of ‘weights’ (W) and ‘activation’ (A). ‘F’ denotes the floating-point precision.	45
3.8	CIFAR-10 test accuracy (%) of MobileNetV2 according to the width multiplier.	46
3.9	CIFAR-10 test accuracy (%) of quantized ResNet according to the block types with varying depth and width. Note that the number of parameters of all experimented models are about one million.	47

4.1	Structure of the proposed SPEQ training scheme for QDNNs. The QDNNs are trained for the target precision n_A through the ‘target precision path’. The ‘stochastic precision path’ produces the teacher logits, z_{SPP} using the same model but with randomly assigned quantization precision for activation at every iteration. Note that the weights in the model can also be quantized to n_W bits.	62
4.2	(a) The ratio of selected 8-bit precision when trained with the greedy strategy. (b) Softmax distributions generated by the stochastic precision path with the same images.	64
4.3	The gradients for the (a) KL loss and (b) cosine loss according to the student probability (x-axis) and teacher probability (y-axis). The direction change on gradients of KL loss is due to the blue region. . .	67
4.4	Examples of (a) gradients and (c) softmax outputs when the teacher is more confident than the student. (b,d) Different direction of the gradients when the teacher is less confident.	68

Chapter 1

Introduction

1.1 Quantization of Deep Neural Networks

Deep Neural Networks (DNNs) have achieved remarkable accuracy for tasks in a wide range of application domains, including image processing [1], machine translation [2], and speech recognition [3]. However, state-of-the-art DNNs have been enlarging constantly. For example, deep convolutional neural networks (CNNs) such as DenseNet [4] and PyramidNet [5] consist of more than 25 and 100 million parameters, respectively. Furthermore, the number of parameters in an attention-based neural network exceeds two billions [6]. Despite the success of the DNNs on various tasks, this complexity poses a tremendous challenge for widespread deployment and forces immense computational power for the servers. In practice, there are increasing demands for implementing DNN models in resource-constrained edge environments, such as the Internet of Things (IoT), automatic driving, and mobile devices without connecting to the server. Those requirements necessitate studies in model compressing that minimize memory footprint and computation while preserving the accuracy as much as possible. In this dissertation, we focused on network quantization as a critical method for developing efficient DNNs.

Network quantization is the most well-known technique that reduces the model size and the computational costs of DNNs. Quantization on neural networks is a function

that maps the continuous parameters and the hidden signals to discrete values. Basically, operations for DNNs are conducted using 32-bit floating-point precision. By quantizing DNN variables and signals under 32 bits, the computation and memory access costs can be reduced significantly. Therefore, quantized deep neural networks (QDNNs) are essential for implementing DNNs on resource-limited hardware. Generally, network quantization is performed on weights and activation outputs in DNNs.

1.1.1 Weight and Activation Quantization on Deep Neural Networks

Weight quantization converts the trained parameters to a low precision. The parameters in DNNs are usually stored in 32-bit floating-point precisions. Reducing the precision under 32 bits can directly reduce the storage for the implementation of DNNs. Several previous studies have shown that the weights in most DNNs can be quantized to 8-bit without losing recognition performance [7, 8]. For extreme quantization such as 1 or 2 bits, however, additional techniques are required to recover the performance degradation from the quantization. It is known that the performance of those low-precision QDNNs can be improved by the retraining technique on the quantization domain [9, 10, 11]. During the training procedure, the gradients are computed using those quantized weights, but applied to the high-precision weights. Those high-precision weights are then quantized for the next iteration. This training procedure on the quantization domain helps the accumulation of small gradients and prevents the gradient from vanishing by the quantization.

Activation quantization discretizes the outputs of the activation functions, such as sigmoid, hyperbolic tangent (tanh), and Rectified Linear Unit (ReLU). By the activation quantization, the outputs of hidden layers can be represented in low-precision, and thus the memory accesses for storing and loading the hidden features can be reduced. Furthermore, the matrix multiplications or convolution operations of hidden layers can be conducted in a low-precision level when weights and activation outputs are quantized. Unlike the weights of DNNs, activation outputs vary according to the input

values. Therefore, the quantization operation should be conducted during the inference and the complex quantization methods are not suitable. Most studies for activation quantization employ simple quantization methods, such as uniform [12] and piece-wise uniform quantization [13]. The activation quantization errors can also be alleviated by the additional training.

With the complex quantization algorithm and retraining schemes, several previous studies successfully reduced the precision of weights to 1 or 2 bits without the loss of performance [11, 9]. However, such an extreme quantization may have substantial performance drop depending on the structure of the model and the complexity of data [14, 12]. Most studies found the optimal precision for a DNN model by the extensive simulation.

1.1.2 Analysis of Quantized Deep Neural Networks

There have been several studies to minimize the quantization errors by measuring the signal-to-quantization-ratio (SQNR) [15, 16]. These studies compared the SQNR according to the quantization methods, such as the uniform, Gaussian, and Laplacian quantization [15], or quantization precision [16] to optimize the word length of parameters. However, they have not considered the resiliency of the QDNNs by the retraining procedures. It is known that information lost by the parameter quantization can be newly learned through a retraining process [9, 10]. The resiliency of QDNNs by the retraining procedure is closely related to the model capacity. Specifically, it is more difficult to recover the quantization errors when the model is small [14]. Since network quantization aims to reduce the implementation cost, it is very important to develop the quantization method for small DNNs without the loss of performance.

Recently, several studies have shown that the training methods for increasing generalization capability also improve the performance of QDNNs. The cyclical bit-precision scheduling [17] or the quantized weight averaging [18] leads the model to reach flat minima on the fixed-point domain. Considering that the over-parameterization helps the

generalization of DNNs [19], reducing the capacity of DNNs by the quantization may decrease the generalization capability. Therefore, the training methods for improving generalization help the optimization of weight quantized DNNs.

As the DNNs becomes deeper and more complex, reducing memory access becomes an important issue. Activation quantization decreases the size of the hidden representations and becomes an essential compression scheme for low-cost implementation. Unlike the weight quantization, there are few studies on the activation quantization error. Rather, various studies assumed that the weight and activation quantization errors are the same [20, 21]. A study experimentally showed that activation outputs require more bit precision than weights to achieve the same performance [12]. Meanwhile, another study demonstrated that a model quantized with activation can be more prone to the adversarial noise [22]. They showed that QDNN is more robust to small noise, but is more vulnerable to adversarial attacks when the magnitude of noise increases. They also applied a regularization technique for limiting the noise amplification to increase the noise robustness of QDNN. Several studies have proposed the quantization methods for the activation outputs, but they have not discussed when or how activation quantization errors cause the performance degradation of QDNNs [23, 13, 24].

1.2 Scope of the Dissertation

1.2.1 Characterization of Quantization Errors

As described in the previous sections, the analysis of weight quantization and activation quantization errors is insufficient and many studies only experimentally assess the performance of quantization algorithms for well-known datasets and model structures. This dissertation tackles these issues and performs explicit analyses of characteristics of quantization errors on DNNs.

In Chapter 2, the memorization capacity of DNNs is measured based on the mutual information to elucidate the performance degradation by the weight quantization. The

model is trained to perform memorization tasks to obtain the per-parameter capacity according to the structure of the model. We show that the capacity of parameters and the sensitivity of quantization are closely related. According to the experimental results, the weight quantization above a certain bit precision does not lower the capacity of the parameter and thus the quantization does not affect performance regardless of the size of DNNs. Furthermore, the limitation of the bit precision depends on the model structure. The per-parameter capacity of complex models such as long short-term memory (LSTM) based recurrent neural networks (RNNs) is higher than that of simple models like fully-connected deep neural networks (FCDNNs). As a result, the weight quantization sensitivity of RNNs is higher than that of the FCDNNs. To verify the capacity-quantization relationship on real tasks, the experiments are extended to the CIFAR-10 image classification task. The results showed that the performance of QDNNs is preserved until 4-bit weight quantization regardless of the size of the models. The performance decreases at higher precision when the complexity of the dataset increases. Since the random uniform dataset is the most complex, the bit-precision boundary obtained from the memorization task is the upper-bound that preserves the capacity of DNNs.

Characteristics of weight and activation quantization errors are visualized and compared in Chapter 3. Since most DNNs and datasets are excessively large and complex, the inference results of the DNNs can be only assessed by simple metrics, such as accuracy for unseen samples. In this chapter, a toy-example is generated to visualize and analyze the performance of DNNs more precisely. Two-dimensional vectors are synthesized as input samples for training FCDNNs. After training FCDNNs with the synthetic dataset, we quantize the weights or the activation outputs and visualize the inference results to show the characteristics of quantization errors. The results indicate that both weight and activation quantizations degrade the performance of models, but the characteristics of those errors are distinctive. The weight quantization errors distort the decision boundary of DNNs, while the activation quantization errors induce noise.

Furthermore, a deeper model is more prone to the activation quantization noise. It is known that the floating-point DNNs usually perform better when the model is deep, which is opposite to the activation quantized DNNs. Based on the analysis, various QDNNs are designed by changing the depth and width of CNNs under parameter constraints to search the quantization-friendly model structures. In addition, we demonstrate that the training methods for improving the generalization capability and the noise robustness are effective for reducing the weight and the activation quantization errors, respectively.

1.2.2 Optimization of Quantized Deep Neural Networks

Characterization of quantization errors indicates that the weight quantization decreases the model capacity and the activation quantization induces noised inference. The knowledge distillation (KD) method is known to not only increase the performance of capacity-limited DNNs [25], but also improve the robustness against the adversarial attacks [26]. Therefore, training with the KD method can improve the performance of QDNNs when both weights and activations are quantized. However, most KD methods require cumbersome large teachers or auxiliary models.

In chapter 4, an efficient QDNN training scheme is proposed using the KD to alleviate both the weight and activation quantization errors, named as Stochastic Precision Ensemble training for QDNNs (SPEQ). The SPEQ is teacher-student learning, but the teacher and the student share the model parameters. We obtain the soft labels of the teacher by changing the bit-precision of the activation stochastically at each layer of the forward-pass computation. The student model is trained with these soft labels to reduce the activation quantization noise. Since the teacher model changes continuously by random bit-precision assignment, it exploits the effect of stochastic ensemble KD. The SPEQ method outperforms on various tasks, such as image classification, question-answering, and transfer learning without the need for cumbersome teacher networks.

The significant portions of the materials in Chapter 2 were previously published in [27]. In addition, Chapter 3 was accepted to the IEEE Workshop on Signal Processing Systems (SiPS 2020) and Chapter 4 has been submitted to the Thirty-Fourth Annual Conference on Neural Information Processing Systems (NeurIPS 2020).

Chapter 2

Memorization Capacity of Deep Neural Networks under Parameter Quantization

2.1 Introduction

Deep neural networks (DNNs) have achieved impressive performance on various machine learning tasks. Several DNN architectures are known, and the most famous ones are fully connected DNNs (FCDNNs), convolutional neural networks (CNNs), and recurrent neural networks (RNNs).

It is known that neural networks do not need full floating-point precision for inference [7, 9, 15]. A 32-bit floating-point parameter can be reduced to 8-bit, 4-bit, 2-bit, or 1-bit, but this can incur performance degradation. Therefore, precision should be optimized, which is primarily conducted by extensive computer simulations using the training data. This not only takes much time for optimization but also can incorrectly predict the performance in real environments when the characteristics of input data are different from the training data.

In this chapter, we attempt to measure the capacity of DNNs, including FCDNN, CNN, and RNN, using a memorization and classification task that applies random binary input data. The per-parameter capacities of various models are estimated by measuring

the mutual information between the input data and the classification output. Then, the fixed-point performances of the models are measured to determine the relationship between the quantization sensitivity and the per-parameter capacity. The memorization capacity analysis results are extended to real models for performing image classification and language modeling, by which the parameter quantization sensitivity is compared between memorization and generalization tasks.

The contributions of this study are as follows.

- We experimentally measure the memorization capacity of DNNs and estimate the per-parameter capacity. The capacity per parameter is between 2.3 bits to 3.7 bits, according to the network structure, which is FCDNN, CNN, or RNN. The value is fairly independent of the model size.
- We show that the performance of the quantized networks is closely related to the capacity per parameter, and FCDNNs show the most resilient quantization performance while RNNs suffer most from parameter quantization. The network size hardly affects the quantization performance when DNN models are trained to use full capacity.
- We explain that severe quantization, such as binary or ternary weights, can be employed without much performance degradation when the networks are in the over-parameter region.
- We suggest the sufficient number of bits for representing weights of neural networks, which are approximately 6 bits, 8 bits, and 10 bits for FCDNNs, CNNs, and RNNs, respectively. This estimate of the number of bits for implementing neural networks is very important considering that many accelerators are designed without any specific training data or applications.
- The study with real-models shows that neural networks are more resilient to quantization when performing generalization tasks than conducting memorization.

Thus, the optimum bits obtained with the memorization tasks are conservative and safe estimate when solving real problems.

This chapter is organized as follows. In Section 2.2, previous works on neural network capacity and fixed-point optimization are briefly presented. Section 2.3 explains the capacity measurement methods for DNN models. Section 2.4 presents parameter capacity measurement results for FCDNNs, CNNs, and RNNs. The quantization performances measured on DNNs are presented in Section 2.5. Concluding remarks follow in Section 2.6.

2.2 Related Works and Backgrounds

2.2.1 Neural Network Capacity

The capacity of neural networks has been studied since the early days of DNN research. Although the capacity can be defined in many ways, it is related to the learnability of networks. The capacity of networks is shown as the number of uncorrelated random samples that can be memorized [28]. A single-layer perceptron with n parameters can memorize at least $2n$ random samples [29]. In other words, the network can always construct a hyperplane with n parameters that divides $2n$ samples. Additionally, the capacity of a three-layer perceptron is proportional to the number of parameters [30]. Recently, RNNs were trained with random data to measure the capacity per parameter [31]. Our study is strongly motivated by this research, and extends it to the quantization performance interpretation of generic DNN models, including FCDNN, CNN, and RNN. Recent studies have shown that neural networks have a generalization ability even if the expressive capacity of the model is sufficiently large [32, 33]. In this chapter, we also discuss the effect of network quantization when performing generalization tasks.

2.2.2 Fixed-Point Deep Neural Networks

Early works on neural network quantization usually employed 16-bit parameters obtained by directly quantizing the floating-point numbers [7]. Recently, a retraining technique was developed to improve the performance of quantized networks [9, 15]. Retraining-based quantization was applied to CNN and RNN models, showing superior performance compared to directly quantized ones [20, 34]. Many studies attempting extreme quantization have been published, such as 2-bit ternary [9, 35, 36], 1-bit binary weight quantization, and XNOR networks [10, 11]. Some aggressive model compression techniques also employed vector quantization or table look-up [37, 38]. However, not all CNNs show the same quantization performance. For example, AlexNet [39] shows almost the same performance with only 1-bit quantized parameters. However, the same quantization technique incurs a very severe performance loss when applied to ResNet [11]. A previous study shows that large-sized networks are more resilient to severe quantization than smaller ones [14]. Theoretical works and many practical implementation optimization techniques have been studied [40, 41, 42, 43, 44]. Recent work increases the number of network parameters to preserve the performance under low-precision quantization [45]. Our works are not targeted to a specific data or model, but introduce the general understanding of parameter quantization.

2.3 Network Capacity Measurements of DNNs

2.3.1 Capacity Measurements on a Memorization Task

We assess the network capacity of DNN models using random data memorization and classification task [31]. In this task, N random binary vectors, X , are generated and each is randomly and uniformly assigned to the output label Y . The size of the binary vector depends on the DNN model. For FCDNN, the input X is a one-dimensional vector whose size is determined by the hidden layer dimension. In CNN, the input needs to be a 2-D or 3-D tensor. Input samples of CNNs are generated by concatenating

and reshaping random binary vectors. During the training process, the DNN is trained to correctly predict the label, which is 0 or 1, of the random input X . As the number of input data size, N , increases, the classification accuracy drops because of the limited memorization capacity. Note that the accuracy of the memorization task refers to the training performance after convergence because there is no proper test dataset for the random training samples.

The capacity is measured using the mutual information, defined as a measure of the amount of information that one random variable contains about another random variable [46]. The mutual information of a trained network with N input samples is calculated as follows:

$$\begin{aligned} I(Y; \hat{Y}_\theta | X) &= H(Y|X) - H(Y|\hat{Y}_\theta, X) \\ &= N \left(1 - \left(p \log_2 \frac{1}{p} + (1-p) \log_2 \frac{1}{(1-p)} \right) \right), \end{aligned} \quad (2.1)$$

where p is the mean classification accuracy for all samples under trained parameter θ . If the training accuracy is 1, the model memorizes all random samples and the $I(Y; \hat{Y}_\theta | X)$ becomes the number of samples N . If the training accuracy is 0.5, $I(Y; \hat{Y}_\theta | X)$ goes to 0.

The network capacity is defined as

$$C = \max_{\theta} I(Y; \hat{Y}_\theta | X). \quad (2.2)$$

The accuracy, p , may vary depending on the training method of the model. We find N and p that maximize the mutual information of the networks by iteratively training the models. This optimization employs both grid search- and Bayesian optimization-based hyper-parameter tuning [47]. The optimization procedure consists of three stages. First, we try to find the largest input data size whose accuracy is slightly lower than 1. Second, we perform a grid search to determine the boundary values of the hyper-parameters. The searched hyper-parameters can include initialization, optimizer, initial learning rate, learning rate decay factor, batch size, and optimizer variables. Finally, we conduct

hyper-parameter tuning within the search space using the Scikit-learn library [48]. We add the number of training samples N as a hyper-parameter and use the mutual information of Eq. (2.1) as the metric for the optimization.

2.3.2 Network Quantization Method

The DNN quantization method is adopted from [9]. Pretrained floating-point networks are quantized to the fixed-point networks. Floating-point parameters W are approximated with shared scaling factor Δ and fixed-point parameters W^q . When the quantization precision of each parameter is n -bit, the floating-point parameters are discretized uniformly as follows:

$$W^q = \text{Clip}(\text{Round}(\frac{W}{\Delta}), -M, M), \quad M = 2^{n-1} - 1, \quad (2.3)$$

where $n \geq 2$. n -bit fixed-point can represent 2^n points. However, we use only $2^n - 1$ points for symmetry. For example, 2-bit quantized parameters have ternary points of +1, 0, and -1. When parameters are quantized to the 1-bit binary fixed-point, parameters are represented as either +1 or -1. Binary quantization is as follows:

$$W^q = \text{Sign}(W). \quad (2.4)$$

A scalar value of Δ can be shared at a different level. For example, Δ is usually chosen at channel- or layer-level for CNNs. The channel-wise scaling factor increases the dynamics of fixed-point parameters, but the computation costs more due to the summing along the channel should be executed in floating-point precision. For computational efficiency, we use layer-wise scaling factor Δ_l for all experimented models. Δ_l is set to the value that minimizes the difference between pretrained floating-point and quantized parameters. For each layer l , floating-point parameters W_l are converted to fixed-point parameters W_l^q is as follows:

$$W_l' = \Delta_l W_l^q, \quad \text{s.t.} \quad \Delta_l = \arg \min_{\Delta} \|W_l - \Delta W_l^q\|_2. \quad (2.5)$$

The optimal Δ of Eq. (2.5) can be obtained by Lloyd-Max algorithm as follows:

$$g(x) = \text{sign}(x) \otimes \min(\lceil \frac{|x|}{\Delta^i} \rceil, \frac{M-1}{2}) \quad (2.6)$$

$$\Delta^{i+1} = \frac{x \cdot g(x)}{g(x) \cdot g(x)}, \quad (2.7)$$

where \otimes and \cdot denote the element-wise and inner product, respectively. M is the number of quantization points and $\lceil \cdot \rceil$ is rounding operation.

After quantization, fixed-point networks are retrained to compensate for the error by the weight perturbation. The iteration for retraining is composed of the following three steps. First, loss and gradients are calculated using fixed-point parameters, W' . Then, the gradients are updated to floating-point parameters, W , for accumulating small gradients. Finally, W is quantized to W' following the Eq. (2.3). Δ_l is fixed during the retraining procedure.

2.3.3 Network Quantization and Parameter Capacity

Quantization of model parameters perturbs the trained network, therefore, fixed-point training or retraining with full-precision backpropagation is usually needed [9, 35, 10, 49]. However, the performance of the quantized networks does not always meet that of the floating-point models, even after retraining. This suggests that model capacity is reduced by quantization, especially when the number of bits used is very small.

In this chapter, we observe the memorization capacity degradation caused by quantization in generic FCDNN, CNN, and RNN models. The uniform quantization is used for the sake of convenient arithmetic, and the same step size is assigned to each layer in the FCDNN, each kernel in the CNN, or each weight matrix in the LSTM layer. The bias values are not quantized, because they have a large dynamic range. It is important to note that the weights connected to the output are not quantized, because their optimum bit-widths depend on the number of labels in the output. Quantization is performed from floating-point to 8-bit, 6-bit, 5-bit, 4-bit, 3-bit, and 2-bit precision, in sequence. Retraining is performed after every quantization, but requires only a small

number of epochs, because only fine-tuning is needed [9].

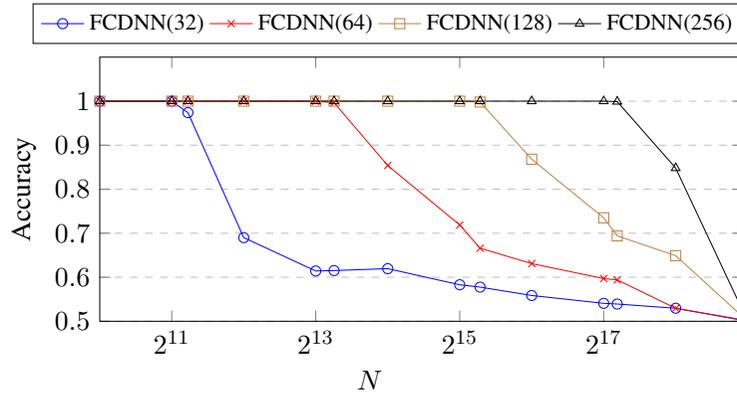
2.4 Experimental Results on Capacity of Floating-point DNNs

The capacities of FCDNNs, CNNs, and RNNs are measured via the memorization task explained in Section 2.3.1. The models used for the test employ floating-point parameters.

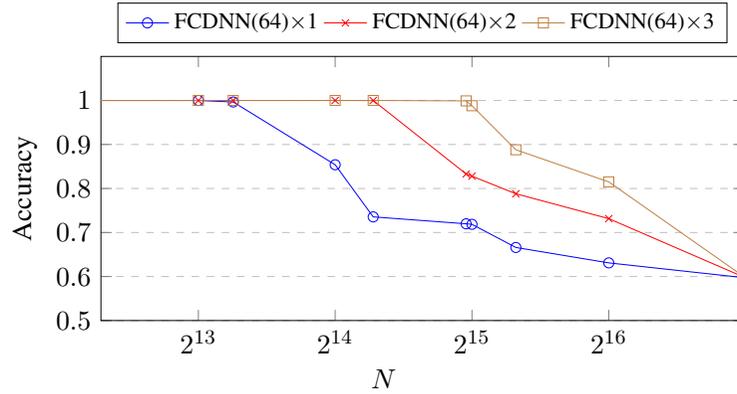
2.4.1 Capacity of FCDNNs

The training data for FCDNNs is a 1-D vector of size n_{in} . N input data are used for the training data. The output, Y , is the randomly assigned label, either 0 or 1, for each input. Thus, inputs, X and Y , are represented as $X \in \{0, 1\}^{N \times n_{in}}$ and $Y \in \{0, 1\}^N$, respectively. The input data dimension, n_{in} , should be larger than $\log_2 N$ so that no overlapped data is contained among N input data. In the experiments for FCDNNs, the input vector dimension, n_{in} , is chosen to be equal to the number of units in the hidden layer.

We conducted experiments for FCDNNs with hidden layer dimensions of 32, 64, 128, and 256, and with hidden layer depths of 1, 2, 3, and 4. The initialization method chosen is the ‘He’ initialization [50] and gradients are updated following the rule in SGD, with momentum, which shows the best performance in our grid search. The initial learning rate for hyper-parameter tuning was chosen between 0.001 and 0.05 on the log scale. The decay factor and momentum were set to have even distance values in the linear scale between 0.1 and 0.5 and between 0.6 and 0.99, respectively. For each model, experiments are conducted to measure the accuracy of memorization while increasing the size of the input data, N . Note that only the training error is measured in this memorization task, because there is no unseen data. Experimental results are based upon the best accuracy obtained when attempted with different hyper-parameters.



(a) Varying width



(b) Varying depth

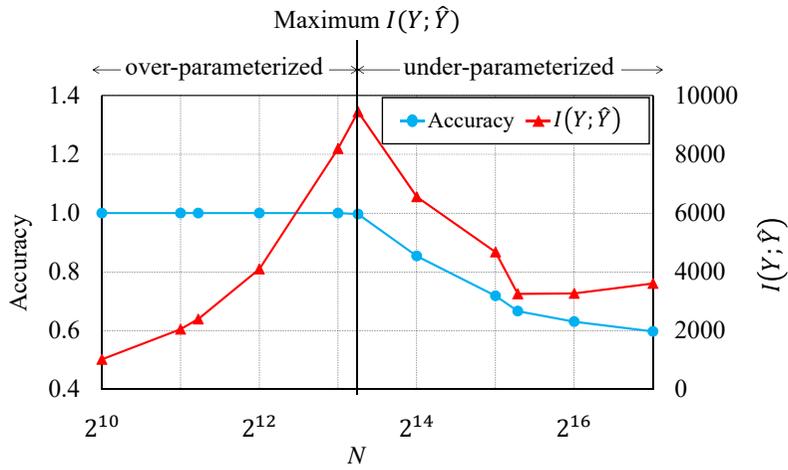
Figure 2.1: Memorization performances according to the (a) width and (b) depth of the FCDNN.

The capacity of the model was estimated according to Eq. (2.1), where p is the training accuracy.

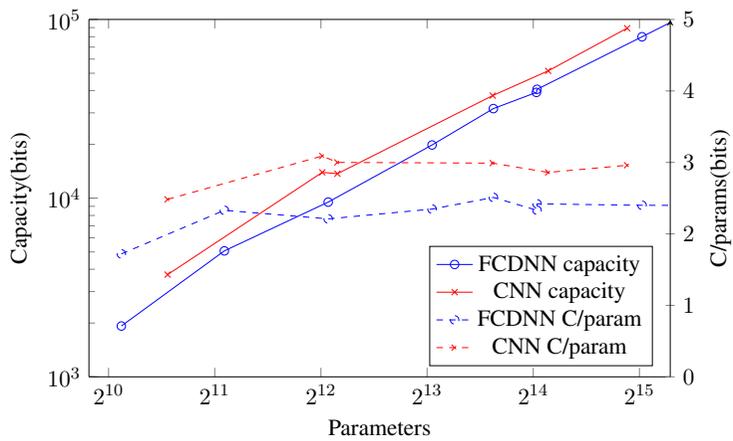
The experimentally obtained memorization capacities of the FCDNN models are presented in Figure 2.1, where depths of 1, 2, 3, and 4, and widths of 32, 64, 128, and 256 were used. When the number of hidden layers is the same, the amount of data that can be almost perfectly memorized quadruples when the dimension of the hidden layer was doubled. This means that the memorization capacity is linearly proportional to the number of parameters. Similarly, the FCDNN models with 2, 3, or 4 hidden layer depths can memorize 2, 3, or 4 times the input data as compared to the single-layer DNN, respectively.

Figure 2.2 (a) shows the memorization accuracy and the mutual information obtained using Eq. (2.1) on the FCDNN. The model is composed of three layers and the hidden layer of size 64. Here, we found that the amount of mutual information steadily increases as the input data size grows. However, it begins to drop as the input size grows farther, and the memorization accuracy drops. By analyzing the accuracy trend of the model, it is possible to distinguish the input data size into three regions: the over-parameterized, the maximum performance, and the under-parameterized sections, as shown in Figure 2.2 (a). For example, if the model is trained to memorize only 10,000 data, it can be regarded as over-parameterized. The number of data that can be memorized by maximally utilizing all the parameters is between 30,000 and 40,000. In over-parameterized regions, performance can be maintained, even if the capacity of the networks is reduced.

The per-parameter capacity of FCDNNs is shown in Figure 2.2 (b). Regardless of the width or depth, one parameter has a capacity of 1.7 to 2.5 bits, and FCDNNs have an average of 2.3-bit capacity per parameter. This result is consistent with theoretical studies [29, 30]. The total capacity of the model may be interpreted as optimal storage that can store a maximum of random binary samples [29, 51].



(a)



(b)

Figure 2.2: (a) Mutual information according to the number of inputs N . (b) The relationship between the number of parameters and the capacity of networks in FCDNNs and CNNs.

2.4.2 Capacity of CNNs

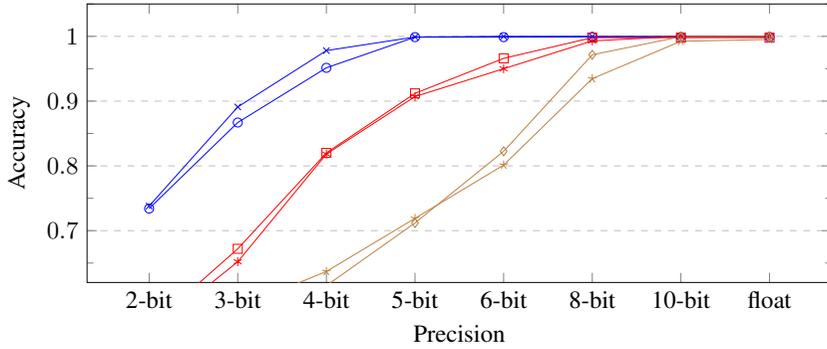
The capacity of CNNs was also measured via a similar memorization task. CNNs can have a variety of structures according to the number of channels, the size of the kernels, and the number of layers. The kernel size of CNNs in this test is either (3×3) or (5×5) , which are the same for all layers, the number of convolution layers from 3 to 9. The dimensions of the inputs are $n_{height} = n_{width} = 32$ and $n_{channel} = 1$ for all experiments. Three max-pooling operations were applied to reduce the number of parameters in the fully connected layer.

The CNN models contain not only convolution layers but also fully connected layers. Thus, the per-parameter capacity for convolution layers was calculated after subtracting the capacity for fully connected layers from the measured total capacity. We assume the per-parameter capacity of the fully connected layer as 2.3 bits to calculate the capacity for convolution layers. As shown in Figure 2.2 (b), the convolution layers have the per-parameter capacity of between 2.86 and 3.09 except the smallest model, which is higher than that of FCDNNs. The average capacity per parameter of the tested models is 3.0 bits.

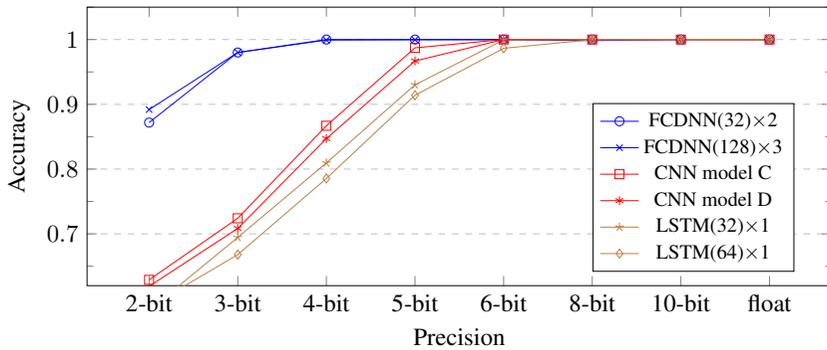
Results show that the per-parameter capacity of CNNs is higher than that of FCDNNs, even when CNNs memorize uncorrelated data. Note that one parameter of FCDNNs is used only once for each inference. However, the parameter of CNNs was used multiple times. This parameter-sharing nature of CNNs seems to increase the amount of information that one parameter can store.

2.4.3 Capacity of RNNs

It has been shown that the various structures of RNNs all have similar capacity per parameter of 3 to 6 bits [31]. We trained RNNs with a dataset with no sequence correlation to show the capacity of the parameters. The random input dataset is composed of inputs, $X \in \{0, 1\}^{N \times n_{seq} \times n_{in}}$ and labels $Y \in \{0, 1\}^N$, which are uniformly set to 0 or 1. The training loss was calculated using the cross-entropy of the label at the output of



(a)



(b)

Figure 2.3: Quantization effect when networks use the (a) full capacity and (b) half capacity.

the last step.

We trained RNNs with a single LSTM layer of 32-D. The input dimension, n_{in} , is also 32-D and the amount of unrolling sequence, n_{seq} , is five-step. It has been reported that unrolling of five-step almost saturates the performance in this setup [31]. We applied 5 input random vectors, X_0, X_1, X_2, X_3 , and X_4 , each with 32-D, and assign one label to this 160-D vector at the last time step. The error propagates from the last step only, and the outputs at intermediate time-steps are ignored. The number of parameters in the network is 8,386. In this case, the maximum mutual information was obtained when the number of samples is 32K, and the memorization accuracy is 99.52

%. Therefore, the per-parameter capacity of the model is 3.7 bits. The RNN shows a higher per-parameter capacity than FCDNNs and CNNs.

2.5 Experimental Results of Parameter Quantization

2.5.1 Capacity under Parameter Quantization

We have shown that FCDNNs, CNNs, and RNNs have different per-parameter capacities. According to the parameter-data ratio, a trained DNN can be an over-parameterized, max-capacity, or under-parameterized model. Thus, we can assume that the DNN performance under quantization would depend on not only the network structure, such as FCDNN, CNN, or RNN, but also the parameter-data ratio. The experiments are divided into two cases. The first is to measure performance degradation via quantization precision when each model is in the maximum capacity region. The second analyzes performance when the models are in the over-parameterized region.

When the FCDNN, CNN, and RNN are trained to have the maximum memorization capacity, the performances with parameter quantization are shown in Figure 2.3 (a). The FCDNN, CNN, and RNN models are shown. The fixed-point performances of two FCDNNs, two CNNs, and two RNNs are illustrated. With 6-bit parameter quantization, the FCDNN shows no accuracy drop. However, those for CNNs and RNNs are 5 % and 18 %, respectively. Because the RNN contains the largest amount of information at each parameter, the loss caused by parameter quantization seems to be the most severe. We also find that there is no decline in performance until the parameter precision is lowered to 6-bit for FCDNNs, 8-bit for CNNs, and 10-bit for RNNs, even when all models use full capacity.

Next, we show the fixed-point performance of DNNs when they are trained to be in the over-parameterized region. Note that the per-parameter capacity is lowered in the over-parameterized region. We conducted simulations with half size of the maximum number of data that can be memorized. For example, an FCDNN used for the

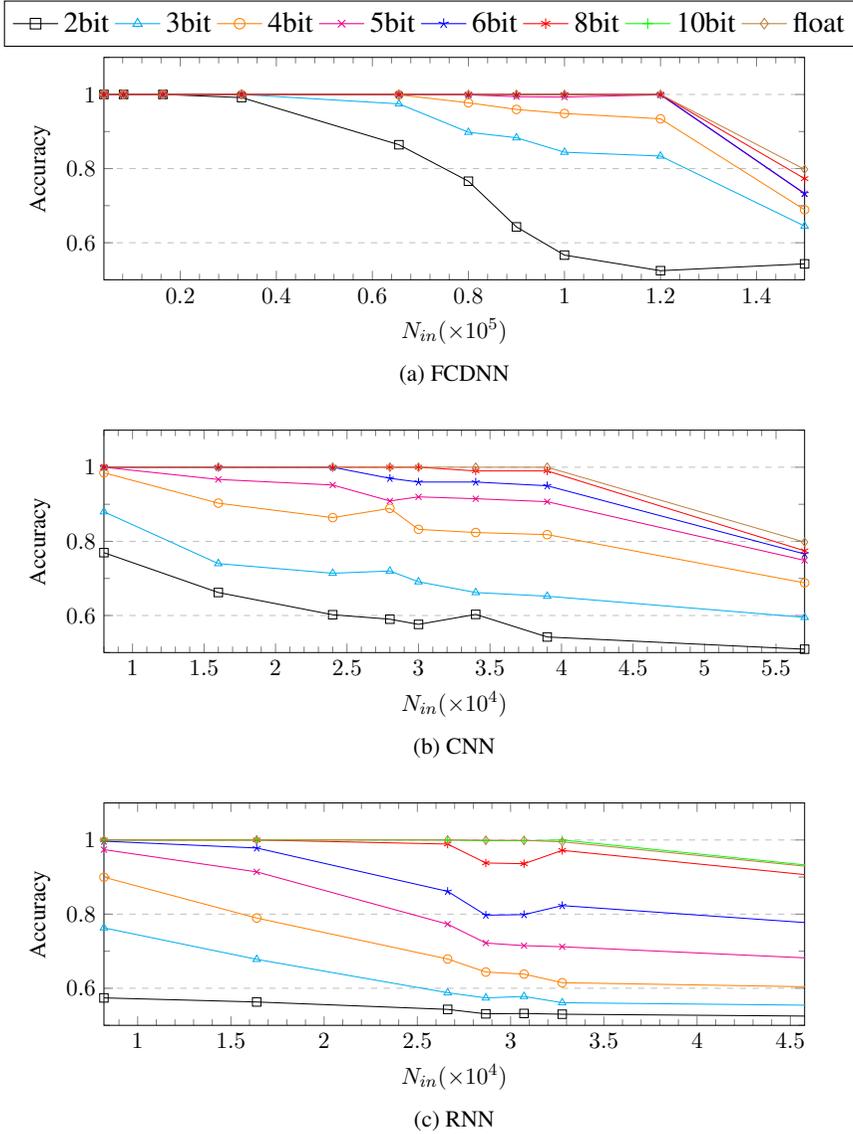


Figure 2.4: Quantization performance according to the number of data to train. (a) FCDNN, (b) CNN, and (c) RNN.

measurement has 3 hidden layers with a hidden-layer dimension of 128; the capacity of the corresponding model is about 2^{17} bits. The network is over-parameterized when the number of memorized samples is 2^{16} . Figure 2.3 (b) shows that the FCDNN model memorizes all samples even with the 4-bit parameter quantization when the model uses half the capacity. Also, over-parameterized model is less sensitive to bit-precision on CNNs and RNNs. The performances of fixed-point DNNs with the number of samples are shown in Figure 2.4. The result shows that DNNs are more robust when the networks are more over-parameterized.

2.5.2 Quantization Experiments on CIFAR-10 Dataset

We analyzed the weight quantization sensitivity using the CIFAR-10 image classification task with varying the model architecture and sizes. The CIFAR-10 dataset consists of 50K training data and 10K test data with 10 classes [52]. The shape of an input image, (h, w, c), is (32, 32, 3). We employed a simple data augmentation, cropping and flipping, as suggested in [53].

The CNNs used in this experiment are based on the ResNet [1] and VGG [54] structures. The training settings for ResNet with the CIFAR-10 dataset are as follows. The number of layers is either 20, 32, 44, 56, or 110 and the width multiplier ranges from 0.3 to 10. For all models, we employed the same training hyper-parameters. The batch size is 128. The number of epochs for pre-training is 175. The SGD optimizer with a momentum of 0.9 is used. The learning rate starts at 0.1 and decays by 0.1 times at the 75-th and 125-th epochs. The L2-loss is used at a scale of $5e-4$. The channel-wise normalization is applied to the input data. All fixed-point models are retrained for 100 epochs with an initial learning rate of 0.01, and the learning rate decreases by a factor of 0.1 at the 40-th and 80-th epochs. We did not employ L2 regularization when retraining. The experimental models are denoted as ResNet (I), where I is the number of channels in the first layer. For example, ResNet20 (32) indicates that the number of channels in the first layer is 32, which is 2 times larger than the number of channels of the original

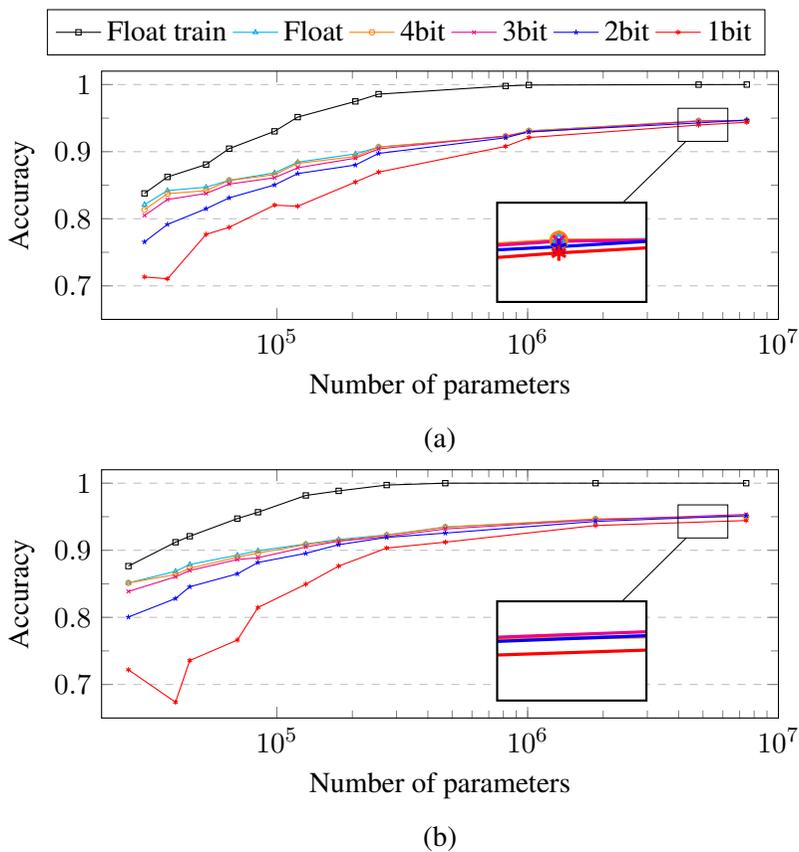


Figure 2.5: Weight quantization results according to the number of parameters on (a) VGG and (b) ResNet structures by changing the width and depth. All results are reported on CIFAR-10 test dataset after retraining except “Float train”.

ResNet20 [1]. This structure is similar to [55] except that we increased the number of channels of the first layer. When the number of channels is the same as the original model, we omitted the notation of the channel width.

For the VGG structures, we employed Adam optimizer [56]. The initial learning rate is 0.0005, and it decays with a factor of 10 at the epochs of 75 and 125. We train the model for 175 epochs. The L2-regularizer with a decay of $5e-4$ is used for the floating-point model training. We apply batch normalization [57] before every ReLU activation. Dropout [58] is applied to the hidden layers with a probability of 0.5. We then quantize

the weights and retrain the model for 175 epochs using an initial learning rate of 0.0005, and it decays with a factor of 10 at epochs of 75 and 125. For quantization, we did not employ L2-regularizer. During all experiments, QDNNs results are reported after retraining.

The effects of the model size differences, such as the width and depth, are compared on the CIFAR-10 classification task. All experimental results reported are for the test dataset unless denoted as “Train accuracy”. The performance of QDNNs with varying model sizes when the weights are quantized is shown in Figure 2.5. ResNet used in the experiments consists of 14, 20, or 32 layers with a varying number of initial channels from 6 to 64. VGGs were constructed by increasing the number of convolutional layers from 6 to 16 and the initial channels from 6 to 96. The number of hidden layers is one for all VGG structure. Note that the width of all channels in the CNNs increases at the same ratio with the width of the initial channel increase. As discussed in the previous section, the robustness of a CNN model to weight quantization increases as the depth or width increases. It is worth noting that, even if the floating-point model is very small and the training accuracy is far from 100%, the weight quantization does not cause a performance degradation until 4-bit precision. The experiment using the random dataset indicates that 8-bit weight quantization does not reduce the capacity of parameters on memorization tasks. In generalized models, a lower-bit precision QDNN is possible even if the capacity of the model is not sufficient. When the CIFAR-10 dataset is used for training ResNet and VGG structured CNNs, weight quantization with at least a 4-bit precision does not reduce the capacity. Our results imply that the weight quantization precision boundary, which does not affect the performance, can differ depending on the correlation of the dataset.

2.5.3 Quantization Experiments on Shuffled CIFAR-10 Dataset

We shuffled and trained the CIFAR-10 dataset to see the relationship between data complexity and weight quantization sensitivity. The training images of CIFAR-10

Table 2.1: Accuracies on CIFAR-10 dataset according to the channel width multiplier of ResNet20

Init channel width (I)	ResNet20 (I)				
	10	12	14	16	18
# Parameters	107.8K	154.7K	209.9K	273.7K	345.8K
Train acc(%)	97.11	98.48	99.27	99.66	99.80
Test acc(%)	90.01	90.92	91.77	92.09	92.55

Table 2.2: Full precision performance of ResNet20 (14) according to the shuffling probability, p .

p	0	0.1	0.2	0.3	0.5	0.7	1.0
Train acc(%)	99.27	91.83	84.40	76.26	61.70	50.57	34.11
Test acc(%)	91.77	90.84	90.01	89.52	88.16	85.15	14.81

is shuffled pixel-wise according to the shuffling probability, p . For example, 25,000 out of 50,000 training images are pixel-wise shuffled with different seeds when p is 0.5. The experimented value of p is in [0.0, 0.1, 0.3, 0.5, 0.7, 1.0]. When p is 0, the training dataset is the same as the original one and when p is 1.0, all the training images are shuffled. All hyper-parameters used for training are the same as described in Section 2.5.2.

The training accuracy is compared to determine when the weight quantization drops the capacity of the model. We first find the size of the model that can barely memorize the original dataset to eliminate the quantization robustness effect caused by over-parameterization. The performance of the floating-point model by changing the channel width multiplier of ResNet20 are shown in Table 2.1. When the initial channel

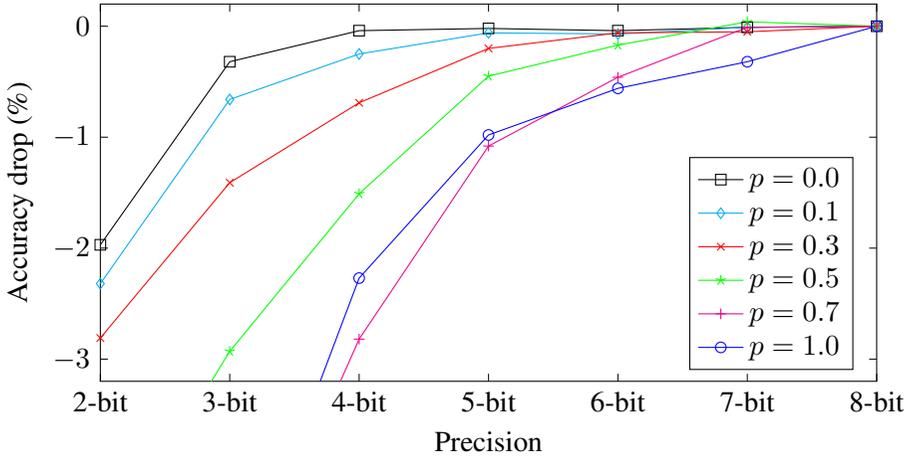


Figure 2.6: CIFAR-10 train accuracy drop by the quantization on ResNet20 (14) according to the shuffling probability, p . The accuracy drop means the difference between the accuracy of 8-bit and target precision.

width is 12, all training samples cannot be memorized, showing the training accuracy of 98.48%. When the width is 14 or larger, networks show more than 99% training accuracy. Therefore, ResNet20 with the initial channel width of 14 can be seen as a model that utilizes its full capacity for the CIFAR-10 training samples.

We employ the ResNet20 (14) to compare the quantization sensitivity by shuffling the training samples. The training accuracy of the floating-point models according to the probability p is shown in Table 2.2. When P is 0, the training accuracy of the floating-point model is about 99.2% and thus the model uses full capacity. The dataset becomes more difficult to memorize as p increases, resulting in the training accuracy drop. Therefore, ResNet20 (14) is a suitable model for analyzing the relation between the quantization sensitivity and data complexity because the quantization resiliency by over-parameterization can be ignored [14].

The degradation of the training accuracy with the bit precision of the quantized model is shown in figure y. All results were obtained after retraining with the target bit precision. The accuracy reduction is reported by subtracting the training accuracy of the

8-bit weight quantized model for each p . When p is 0, i.e. original dataset, the training accuracy is preserved until 4-bit quantization. This is the same as in which there was no performance degradation by the 4-bit quantization in the under-parameterized ResNet on the CIFAR-10 dataset. As p grows, the precision boundary that begins to lose the accuracy also increases. The training accuracy starts to decrease from 4, 5, 6, and 7 bits when p is 0.1, 0.3, 0.5, and 1.0, respectively. In other words, capacity reduction by quantization occurs at higher precision when the training dataset is more complex. Considering that the independent uniform random samples are the most complex and difficult to generalize, These results verify that the found precision boundaries on the memorization task are conservative.

2.6 Concluding Remarks

Quantization of parameters is a straightforward way of reducing the complexity of DNN implementations, especially when VLSI or special-purpose neural processing engines are used. Our study employed simulations on various DNN models. Memorization tests using random binary input data were conducted to determine the capacity by measuring the mutual information. Our simulation results show that the per-parameter capacity is not sensitive to the model size, but is dependent on the structure of the networks, such as FCDNN, CNN, and RNN. The maximum per-parameter memorization capacities of FCDNNs, CNNs, and RNNs are approximately 2.3 bits, 3.0 bits, and 3.7 bits per parameter. Thus, RNNs have a tendency of demanding more bits when compared to FCDNNs. We quantized DNNs under various capacity-utilization regions and showed that the memorization capacity of parameters are preserved up to 6 bits, 8 bits, and 10 bits on FCDNNs, CNN, and RNNs, respectively. The performance of the quantized networks was also tested with the image classification task. The results show that networks need more parameter precision when conducting memorization tasks, rather than inferring with unseen data. Thus, the precision obtained through the memorization test can be

considered a conservative estimate in implementing neural networks for solving real problems. This research not only gives valuable insights into the memorization capacity of neural networks but also provides practical strategies for training and optimizing fixed-point DNNs.

Chapter 3

Characterization and Holistic Optimization of Quantized Deep Neural Networks

3.1 Introduction

Deep neural network (DNN) applications frequently demand immense models for an improved performance, which consumes a large amount of computation power not only for training but also for inference [59, 60]. Thus, it is necessary to reduce their complexity for implementation on embedded devices. Various DNN compression methods have recently been devised to reduce the computational cost, power consumption, and storage space. Network quantization is one well-known method for substituting 32-bit floating-point weights with low bit-width numbers that usually employ one to four bits. Specifically, the performance of a quantized DNN (QDNN) is mostly maintained when retraining is applied after weight quantization [9, 10]. Meanwhile, activation quantization has also been studied to reduce the computational cost and working memory footprint [11, 12]. Therefore, activation quantization is particularly effective for DNN models with a large hidden-state dimension, such as convolutional neural networks (CNNs). Most previous studies on QDNN optimization have focused on the quantization number formats and the training methods. The goal of these previous studies has

been reducing the performance gap between the floating-point and quantized models. However, not all networks can be quantized in the same manner. Some networks are more robust to weight quantization, whereas some others are not [14]. Optimizing a QDNN requires understanding the characteristics of such quantization errors.

In this chapter, we visualize the characteristics of the quantization errors and their effects on the performance of QDNNs when the model architecture and sizes are different. We use synthetic data and DNN models for error characteristic visualization. Based on the analysis results, we adopt two simple training methods to compensate weight and activation quantization errors; fine-tuning with cyclic learning rate scheduling for improved generalization and applying regularization term that reduces noise amplification through propagation. Experiments are conducted using CIFAR-10, ImageNet, and PASCAL VOC 2012 semantic image segmentation benchmark. The contributions of this study are as follows:

- We visualize the errors from the weight and activation quantization. The results indicate that the effect of the weight quantization error reduces the generalization capability whereas activation quantization error induces noise.
- We show that increasing the width of a DNN model helps to mitigate the quantization effects of both the weight and activation whereas increasing the depth only decreases the weight quantization error.
- We reduce the weight and activation quantization errors by employing the training methods that improve the generalization capability and a regularization term that increases the noise robustness, respectively.
- This work is a holistic approach for the optimization of QDNN by examining the quantization effects of weights and activations, and also the architectural change.

3.2 Backgrounds

3.2.1 Related Works on Network Quantization

Most DNN models are trained using 32-bit floating-point numbers. Apparently, DNN models do not demand 32-bit precision. Many quantization methods have been developed, some of which use an extremely small bit-width for a weight representation, such as 1-bit binary [10, 11] or 2-bit ternary [61, 36]. The signal-to-quantization-noise ratio (SQNR) of several weight quantizers was also compared [15]. Quantization noise has been measured to find a better training scheme [62] or optimal quantization precision [63]. Activation quantization has also been developed to lower the computational costs [23]. An efficient QDNN implementation on embedded systems has also been studied [64, 65]. The weight quantization effects usually depend on the model size; small DNN models tend to show considerable performance degradation after quantization [14]. In particular, increasing the number of parameters in CNNs reduces the quantization sensitivity [45]. However, considering the purpose of model compression, the number of parameters needs to be constrained. A recent study showed that weight quantization up to certain bits does not reduce the memorization capacity [27]. During the last several years, residual connections have been developed mainly for improved training of neural networks [1]. Architectural modifications of increasing the width or moving the location of activation and batch normalization have been studied [55, 66]. These architectural changes also affect the quantization sensitivity.

The activation quantization has not been discussed as much as weight quantization, and most studies have not distinguished the effects of activation and weight quantization [11, 13]. It has been observed that activation usually demands more bits than weights [12]. The different quantization approaches for the weight and activation are applied in [24] because the latter was not suitable for cluster-based quantization. Many studies have shown that DNNs can be vulnerable to noise. Even with an extremely small amount of noise, the inference of a DNN can easily be manipulated [67, 68]. Previous

studies have shown that quantizing the input makes it robust to adversarial attacks by reducing the amount of noise [69]. Several studies have shown that a QDNN can help defend from adversarial attacks [70, 71]. However, QDNNs become more vulnerable to adversarial attacks than floating-point models when the noise exceeds a certain level [22]. The adversarial noise becomes larger at the deeper layers [72].

3.2.2 Revisit of QDNN Optimization

The process of uniform quantization for a DNN involves the following two steps, namely, clipping and quantization:

$$\hat{x} = \text{Clip}(x, \alpha, \beta), \quad Q(x) = \Delta \lfloor \frac{\hat{x}}{\Delta} + 0.5 \rfloor. \quad (3.1)$$

The parameters of the DNNs are signed values so that the clip value $\beta = -\alpha = \Delta(2^{n-1} - 1)$ where n is the number of bits used to represent each parameter. Parameter quantization is mainly applied to the weights. For the sake of simple structure, all fixed-point weights in a layer share one scale factor Δ . The activation quantization is used to lower the computational cost and the size of the working memory for inference. When using the ReLU activation, the hidden vectors are represented with unsigned values and α and β becomes 0 and $\Delta(2^n - 1)$, respectively. Low-precision quantized networks require training in a fixed-point domain to improve the performance as follows:

$$W_t^q = Q(W_t) \quad (3.2)$$

$$E_t = f(x_t, y_t, W_t^q) \quad (3.3)$$

$$W_{t+1} = W_t - \alpha \frac{\partial E_t}{\partial W_t^q}, \quad (3.4)$$

where E_t is the loss computed through the model, $f(\cdot)$, at t -th iteration. Forward and backward propagations are conducted using quantized weights and activation. However, the computed gradients need to be added to the floating-point weights because those gradients are relatively small compared to the step size Δ [10]. It is known that QDNNs perform better when retrained from a pretrained model at floating-point than trained from the scratch [12, 13].

Most QDNN studies quantize the weights or activation according to Eq (3.1), although the processes of obtaining Δ , α , and β are different [9, 23]. However, the effect of each quantization on the inference is quite different. The quantization errors are $\epsilon_W = W - Q(W)$ and $\epsilon_a = a - Q(a)$ where ϵ_W and ϵ_a are errors owing to the quantization of the weight and activation, respectively. Because the trained weights have fixed values during inferences, ϵ_W is a constant error. In other words, weight quantization can be modeled as the process of distorting the weights of the DNNs. This changes the direction of the input-prediction mapping function of the DNNs and thus causes distorted results at the inference. By contrast, ϵ_a is an error that depends on the input applied during the inference process. Depending on the remainder of the hidden vector divided by Δ , the direction or magnitude of the error may change. That is, ϵ_a induces noise with a maximum magnitude of $\frac{\Delta}{2}$.

In the rest of this chapter, we analyze how such differences in quantization errors affect the performance of QDNNs under various model architectures. The weight quantization method in [9] and the PACT activation quantization [23] are adopted for our experiments. QDNNs are retrained from pretrained floating-point models.

3.3 Visualization of Quantization Errors using Synthetic Dataset

3.3.1 Synthetic Dataset Generation

Most DNNs and their training samples used in real tasks have extremely high dimensions, and it is therefore very difficult to discern the effects of quantization. For a visualization analysis of QDNNs, we synthesized 2D inputs whose elements consist of x and y axes. The training dataset is composed of inputs $S \in \mathbb{R}^2$ and $C \in (0, 1)$, which are used to train FCDNNs for binary classification. The training dataset is synthesized through two steps. First, the core samples, s_c , mapped to a label, c , are generated using

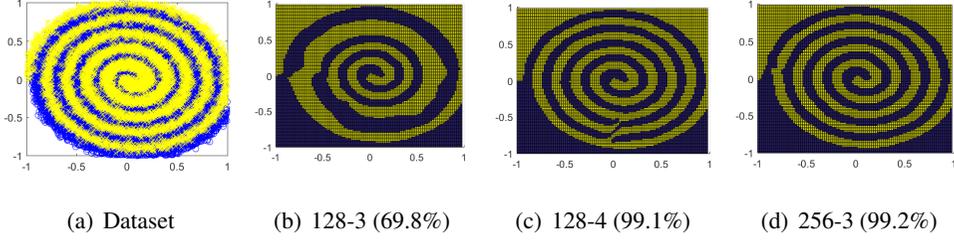


Figure 3.1: Illustrations of the dataset and prediction results from FCDNNs. (a) The synthetic dataset used to train FCDNNs. (b) An example of the evaluation results when the model is too small to learn to the data distribution. (c, d) Examples of large models. The values in parentheses are the accuracy for the correct answer.

the following equation:

$$s_0 \in \left\{ (x, y) \mid \begin{cases} x^2 + y^2 = (2i + 1)^2 r^2, & y > 0 \\ (x - r)^2 + y^2 = (2i + 2)^2 r^2, & y \leq 0 \end{cases} \right\}, \quad (3.5)$$

$$s_1 \in \left\{ (x, y) \mid \begin{cases} x^2 + y^2 = (2i + 2)^2 r^2, & y > 0 \\ (x - r)^2 + y^2 = (2i + 1)^2 r^2, & y \leq 0 \end{cases} \right\}. \quad (3.6)$$

In our experiments, i is within $\{0, 1, 2, 3, 4\}$ and r is 0.1. For each i , two semicircles correspond to one label. In each semicircle, we sample 100 points by increasing the angle linearly. Therefore, the total number of core samples is 2,000. Next, we generate subsamples by adding Gaussian noise to each core sample as follows:

$$s' = s + \epsilon, \quad \epsilon \sim \mathcal{N}\left(0, \frac{1}{3}r\mathbf{I}\right). \quad (3.7)$$

Nine subsamples for each core sample are used, and the samples are mapped to the same labels. As a result, the total number of datasets used for training is 10,000 for each label. The distribution of the generated dataset is shown in Figure 3.1 (a).

We quantize the FCDNN trained using the generated dataset and analyze the difference between the weight and activation quantization. In particular, we visualize

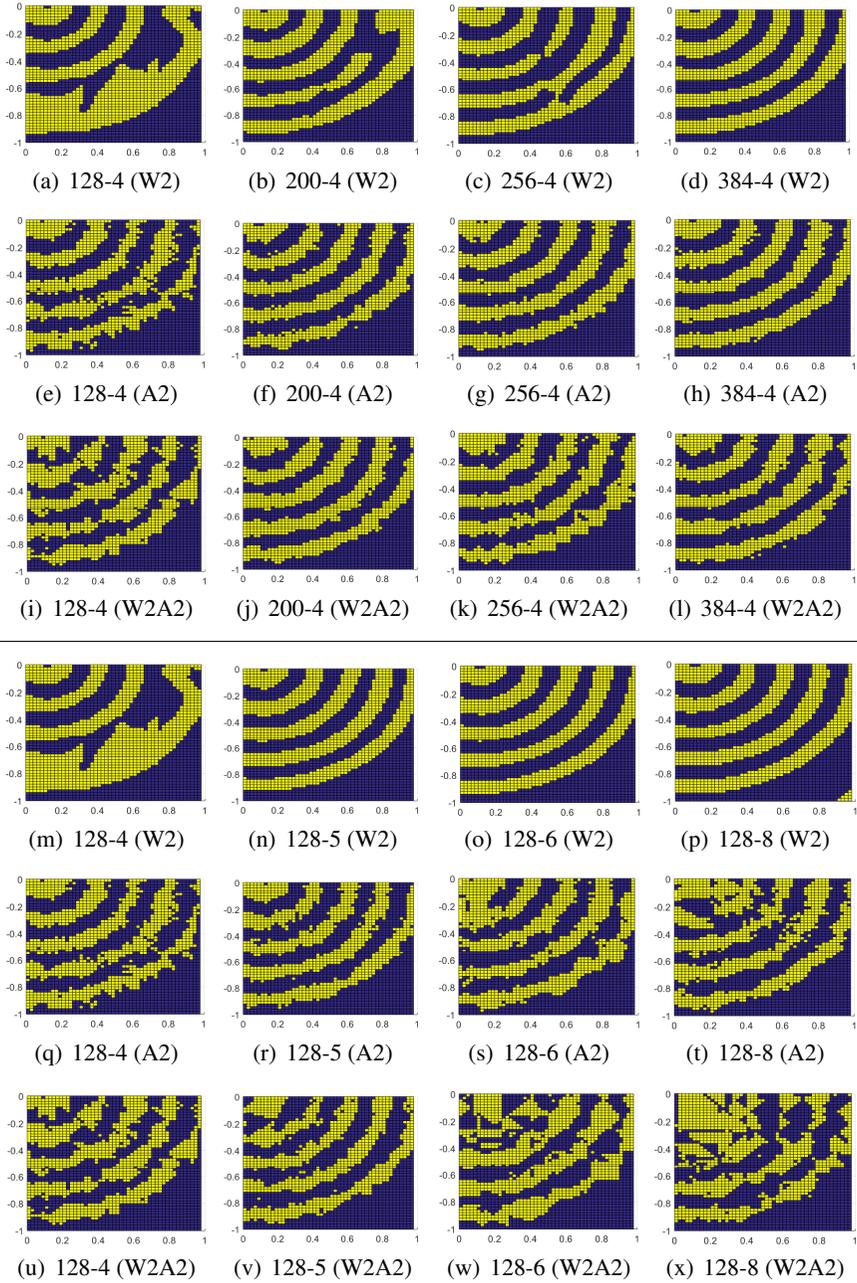


Figure 3.2: Prediction results of QDNN as the width and depth increase. **(W2)** 2-bit weights. **(A2)** 2-bit activations. **(W2A2)** 2-bit weights and activations.

the errors when varying the model depth or width. Two datasets are adopted for the evaluation of the trained model. The first is a test dataset. The correct answer dataset is constructed by dividing the area corresponding to each label by the radius of the semicircle. This dataset is used for quantitative analysis of the trained QDNNs by measuring accuracy. The other is a grid dataset that consists of $(x, y) \in \{(x, y) | 0 < x < 1, 0 < y < 1\}$. By visualizing the prediction on the x - y plane, we can analyze whether the input-prediction mapping of a DNN is distorted or as added noise.

3.3.2 Results on Synthetic Dataset

We devise artificial DNN models for testing with the synthetic dataset. Fully-connected DNN (FCDNN) models were employed with varying depth and width. In addition, models with residual connections were also considered. We indicate the experimental models as “width” - “depth” of FCDNNs. The prediction results of floating-point models using the evaluation dataset are shown in Figure 3.1. The 128-3 FCDNN shows quite a different prediction result from the actual data distribution. Figure 3.1 (c,d) show that increasing the depth to 4 or the width to 256 is sufficient to learn the synthesized dataset quite faithfully. For the remaining experiments, we represent only the bottom-right quarter circle for detailed visualization. All QDNN results are reported after retraining.

Figure 3.2 (W2) shows the effects of the weight quantization according to the width and depth of FCDNNs. We can see that weight quantization distorts the input-prediction mapping of the DNN. The evaluation results of the weight quantized models resemble that of a small floating-point model, such as the 128-3 FCDNN shown in Figure 3.1 (b). The experiment results show that the decrease in learning ability occurs similarly when the number of parameters is reduced or the weights are quantized. As studied in [19], increasing the model size helps generalization. Our experiments show that the generalization capability decreases as the precision of the parameters is lowered. Thus, the effect of reduced generalization capability due to the weight quantization is not noticeable when the model size is large enough. The distortion with 2-bit weight

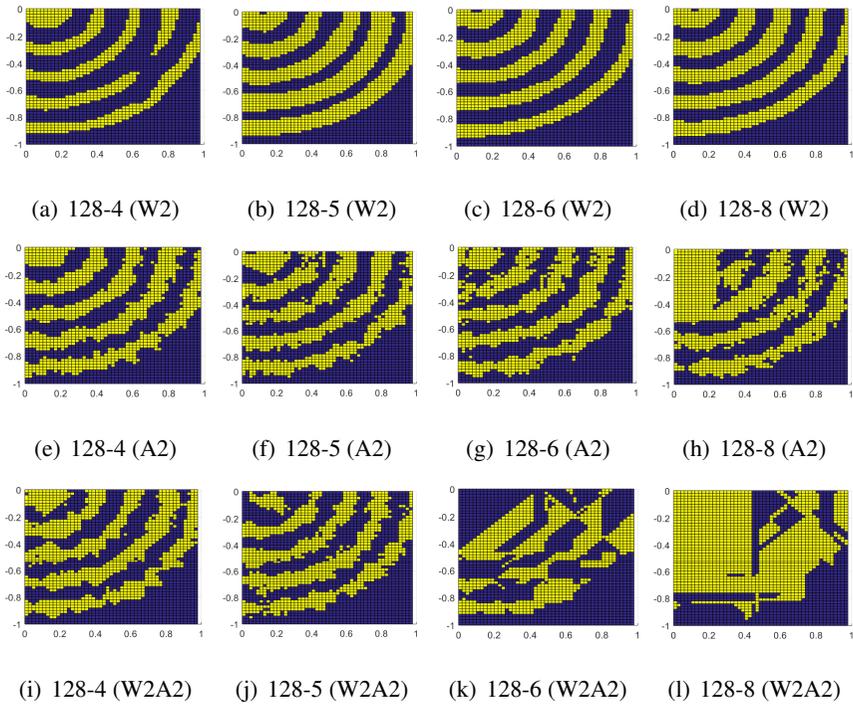


Figure 3.3: Prediction results of QDNNs with residual connection. **(W2)** 2-bit weights. **(A2)** 2-bit activations. **(W2A2)** 2-bit weights and activations.

quantization is barely found when the layer width or the number of layers is increased.

Figure 3.2 **(A2)** shows the activation quantization results. The effect of the activation quantization is very different from that of the model capacity reduction in a DNN. Activation quantization appears to add noise to the prediction results. Although both the weight and activation quantization errors degrade the performance of DNNs, they behave in a completely different manner. When the activation is quantized to 2 bits, increasing the depth does not mitigate the noise added to the prediction results. Rather, the noise tends to worsen with weight quantization when the depth increases. Activation quantization is related to the dimension of each layer rather than the capacity of the model. Activation quantization in wide FCDNNs (Figure 3.2 (e)) is more robust than in deep FCDNNs (Figure 3.2 (h)). The effect of noise from the activation quantization is

reduced because the number of dimensions of the hidden vector received as the input in each layer is increased. When quantizing both the weight and activation, the two errors are combined, and result in a noisy and distorted prediction, as shown in Figure 3.2 (W2A2).

We also analyzed the effect of residual connections on activation quantization. The residual connection helps train DNNs with an extremely large number of layers [1]. In our experiment, the residual connections were implemented by adding each hidden output to the activation of the next layer. The result is multiplied by 0.5 to preserve the scale of the intermediate results. Figure 3.3 shows the results of the quantizing activation when the residual connection was applied. Residual connections help alleviate the distortion through weight quantization. However, FCDNNs with residual connections are more sensitive to activation quantization than the original models. With residual connections, the outputs of the quantized hidden layer are summed so that the activation quantization noise is also added. As a result, applying residual connections shows more noisy prediction in deep models, such as 128-8 FCDNNs.

3.4 QDNN Optimization with Architectural Transformation and Improved Training

The visualization results with the synthesized data show that weight quantization decreases the generalization capability of DNNs, while activation quantization induces noised inference. Also, the quantization effects depend on the architecture very much. Based on this observation, we employ three approaches for QDNN optimization. The first one is modifying the architecture quantization-friendly. The second one is the training method for improved generalization. This technique is intended to reduce the effects of weight quantization. The third one is applying the regularization term that limits the amount of activation noise.

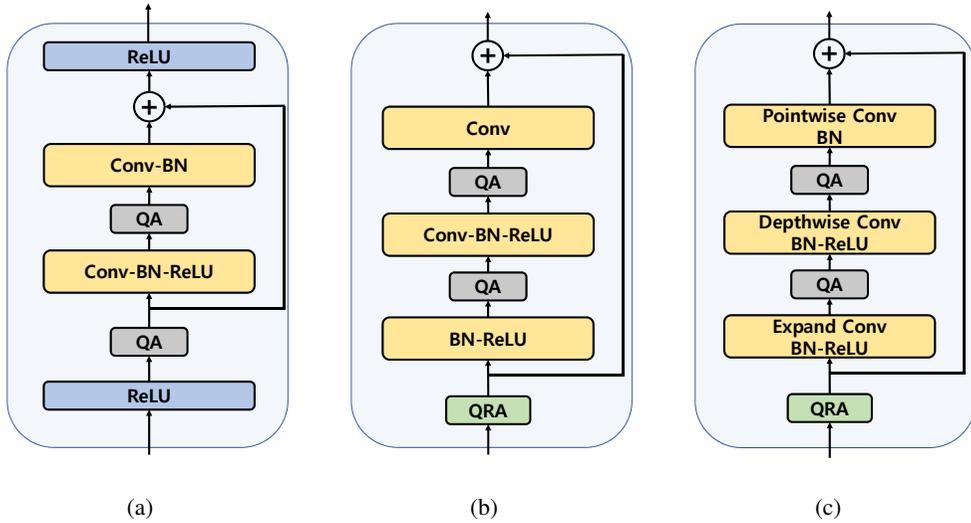


Figure 3.4: Types of residual blocks. (a) basic block, (b) pre-activation block, and (c) depthwise block. QA and QRA denote the activation quantization operations.

3.4.1 Architecture Transformation for Improved Robustness to Quantization

Deep CNN models are hard to train because of the gradient vanishing problem. The residual architecture was developed to solve this problem [1, 73, 74]. In CNN with residual connections, increasing the depth, often over 100, usually helps to improve the performance. Of course, widening the networks also increases the performance [55]. When the number of parameters is limited, increasing the depth is usually preferred because the model complexity rises in linearly proportional to the depth, but squarely proportional to the width. However, our work in Section 3.3 shows that deep CNN models are very prone to activation quantization. The conventional approach for QDNN design is developing the best performing floating-point model, and then quantizing it in the best way possible. In this case, the best performing floating-point model prefers deeper ones, which are, however, prone to activation quantization. Thus, we need to consider the effects of weight and activation quantization even for the initial floating-

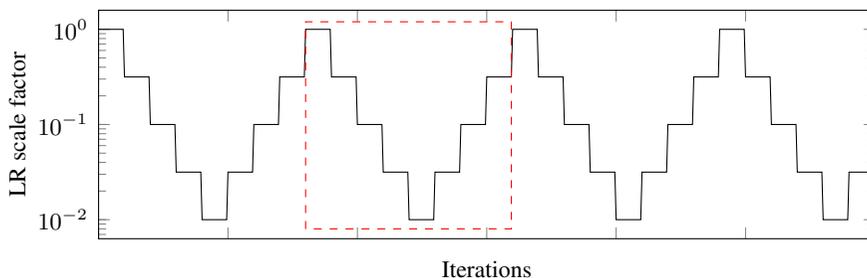


Figure 3.5: Learning rate scale factor along training iterations. The red dashed box indicates a single cycle of CLR scheduling.

point model design. Wide CNN models, which are considered parameter inefficient, often show better performance than deep ones when the activation is severely quantized.

Recent CNN models employ various residual blocks for improved performance or parameter efficiency. The most well-known residual blocks for CNNs are shown in Figure 3.4. The depthwise block employed to MobileNetV2 [75] helps to reduce the number of parameters and computations. However, the quantization performances of these blocks are not well studied.

3.4.2 Cyclical Learning Rate Scheduling for Improved Generalization

We adopt the cyclic learning rate scheduling (CLR) as a way to reduce the effects of weight quantization. CLR increases and decreases the learning rate periodically, while conventional training usually reduces the learning rate in one direction. This method is known to increase the generalization capability of the model by leading to a flat loss surface [76]. Among a few different cyclical learning rate scheduling algorithms, we choose the one that alters the learning rate discretely, which is known to be more effective for generalization [77]. The cyclic learning rate (CLR) scheduling is illustrated in Figure 3.5. The maximum and minimum boundaries of the CLR are determined between the 100 and 0.1 times of the last learning rate of the retraining procedure, respectively. The learning rate changes 8 times in one cycle and exponentially decreases

or increases. Therefore, the scale factor is multiplied to the base LR and changes 8 times in a single cycle; $\{1.0, \sqrt{0.1}, 0.1, 0.1\sqrt{0.1}, 0.01, 0.1\sqrt{0.1}, 0.1, \sqrt{0.1}\}$.

3.4.3 Regularization for Limiting the Activation Noise Amplification

Training methods to increase noise robustness of DNNs have been studied in the field of adversarial training. Parseval networks reduce the Lipschitz constant so that the noise of the input is not amplified as the layer increases [78]. In particular, [22] shows that activation quantized DNNs exacerbated the performance degradation due to adversarial noise, and added the regularization term to the loss to keep the Lipschitz constant of each layer small. The regularization term is as follows:

$$L_{Lip} = \frac{1}{2} \sum_{W_l} \|W_l^T W_l - I\|^2. \quad (3.8)$$

Note that convolution kernels were reshaped to $(k \times k \times c_{in}, c_{out})$ where k , c_{in} , and c_{out} are the kernel size, input channels, and output channels, respectively. L_{Lip} was applied to enhance the adversarial attack robustness of activation quantized DNNs [22]. We show that L_{Lip} can reduce the noise due to activation quantization itself. Also, we compared the effect of the regularization term on the performance of weight quantized DNNs.

3.5 Experimental Results

3.5.1 Visualizing the Effects of Quantization on the Segmentation Task

We first visualized the weight and activation quantization effects using a segmentation task. The PASCAL VOC 2012 dataset [79] is used. The dataset contains 1,464 training images and 1,449 validation images. Each image was labeled at pixel-level with 20 object classes and a background class. The MobileNetV2 [75] was adopted, which was

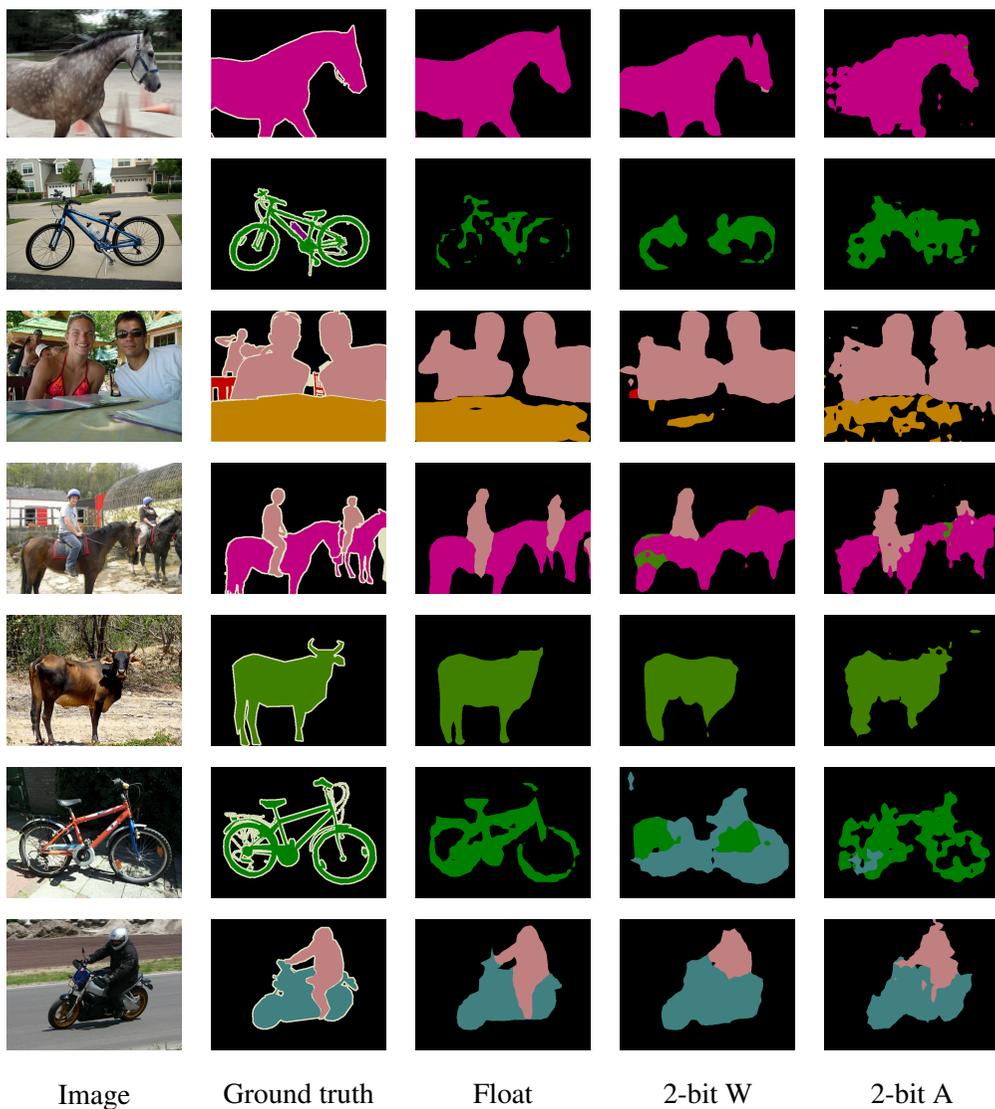


Figure 3.6: Visualization of quantization errors on the PASCAL VOC segmentation benchmark. ‘W’ and ‘A’ are abbreviations for the weight and activation, respectively. Activation outputs are retained in floating-point precision on weight quantized model, and vice-versa.

trained according to DeepLabV3 [80] with 10,582 augmented training images [81] ¹. The model was used as a floating-point pretrained model after fine-tuned using the original 1,464 training images for 30K iterations. The output stride was 16 and Atrous Spatial Pyramid Pooling (ASPP) [82] was not applied. The performance was measured in terms of mean intersection over union (mIOU) without multi-scaling and flipping input images. Only the original training images were used for retraining and fine-tuning. The retraining was conducted for 30K iterations with a batch size of 16. The initial learning rate is 1e-3 and the learning rate policy is the same as [80].

The segmentation results of the retrained QDNNs are visualized in Figure 3.6. Either weights or activations are quantized to 2 bits. When the object is simple, as shown in Figure 3.6 (**first row**), the weight quantized model seems to perform the segmentation fairly well. However, the results with the activation quantized model contain some noise on the section where the background and the object colors are similar. In the segmentation of a complex one, the weight quantized model fails to find the characteristics of the object, as shown in Figure 3.6 (**second row**). We can even consider that the activation noise corrupted model segments the bicycle more faithfully than the weight quantized model. The visualization results imply that weight quantization degrades the generalization ability, and activation quantization induces noise. The experiment with the segmentation task confirms the observation with the synthetic dataset in Section 3.3.

3.5.2 The Width and Depth Effects on QDNNs

We analyzed the weight and activation quantization sensitivities when the depth and width of ResNet vary using the CIFAR-10 dataset [52]. The depth refers to the number of layers and the width corresponds to the number of channels in a CNN. Simple data augmentation techniques, cropping and flipping, are applied as suggested in [53]. The

¹We obtained the pretrained model from <https://github.com/tensorflow/models/tree/master/research/deeplab>

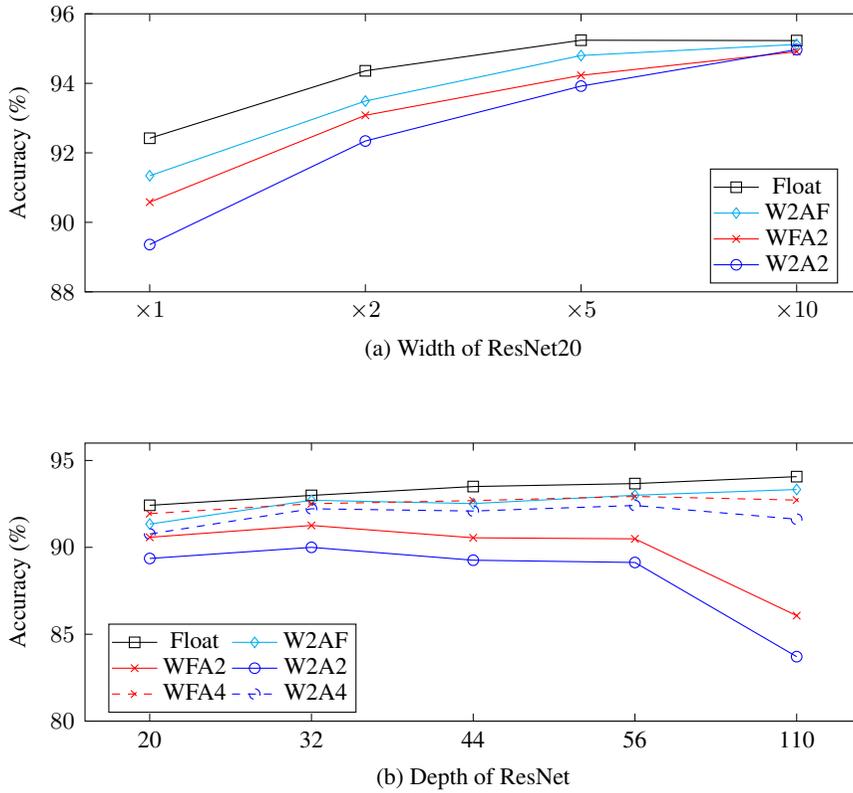


Figure 3.7: Performance of quantized ResNet on the CIFAR-10 testset according to the (a) width of the ResNet20 and (b) depth of the ResNet. Legends represent the precision of ‘weights’ (W) and ‘activation’ (A). ‘F’ denotes the floating-point precision.

batch size is 128 and the number of epochs for pretraining is 200. The SGD optimizer with a momentum of 0.9 is used. The learning rate starts at 0.1 and decays by 0.1 times at 100 and 150 epochs. The L2 regularization was applied with a scale of $5e-4$. Quantized models were retrained for 100 epochs with the initial learning rate of 0.01, and the learning rate decreased by a factor of 0.1 at 50 and 80 epochs. We do not employ the L2 regularization when retraining of the quantized model. The number of layers is either 20, 32, 56, or 110 and the width multiplier ranges from x1 to x10 times of the original ResNet. The experimental model denoted as ResNet $\times I$ employs I times larger number of channels. The width-expanded ResNets are compared in Figure 3.7 (a). As

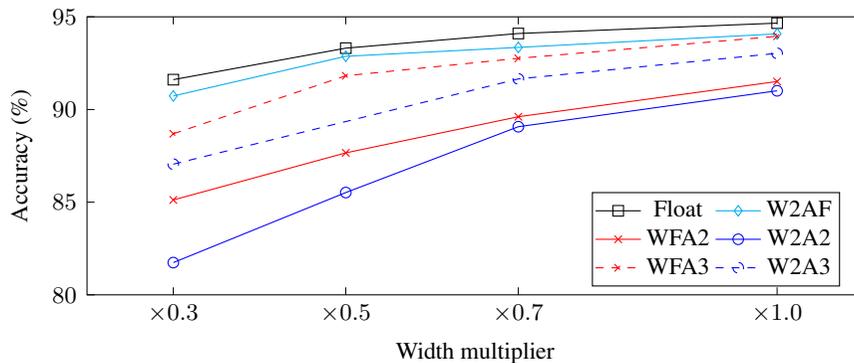
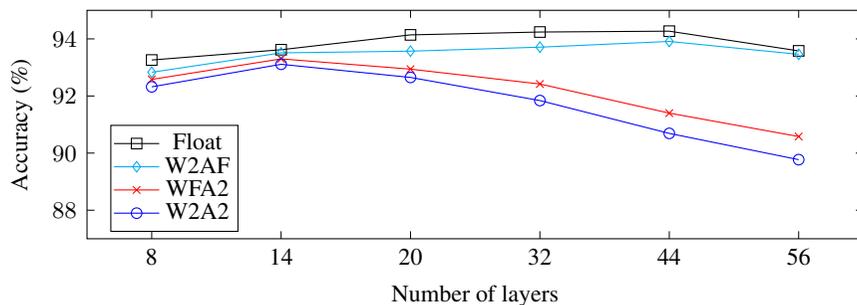


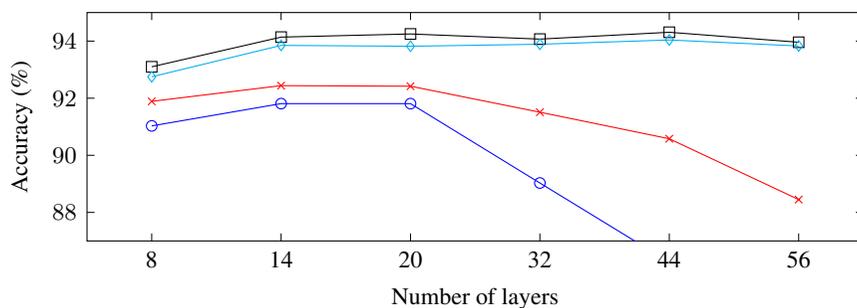
Figure 3.8: CIFAR-10 test accuracy (%) of MobileNetV2 according to the width multiplier.

the width of the ResNet20 increases, both the weight and the activation quantization errors decrease. The performance of ResNets when the depth of layers increases is shown in Figure 3.7 (b). When the depth increases, the 2-bit weight quantized models show improved performance, but those with 2-bit quantized activations exhibit degraded performance. At least, 4-bit activation quantization is needed for deep ResNet.

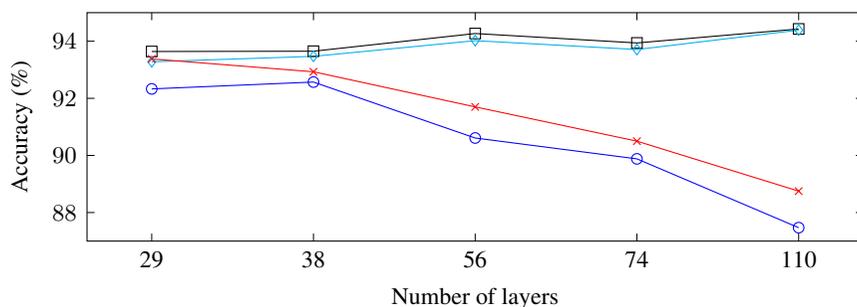
We also evaluated the quantization sensitivity of the MobileNetV2 [75]. The width multiplier is employed to control the number of parameters of MobileNetV2. The MobileNetV2 performance on CIFAR-10 with various width-multiplication factors is shown in Figure 3.8. Note that the MobileNetV2 \times 1.0 is the same with the original model except that the first stride of 2 is replaced to 1. As the width decreases, the performances of full-precision and weight quantized models degrade gradually. However, the activation quantized models exhibit severe performance loss as the width decreases. The results indicate that DNNs with small widths are more vulnerable to the activation quantization.



(a) Basic block



(b) Pre-activation block



(c) Depthwise block

Figure 3.9: CIFAR-10 test accuracy (%) of quantized ResNet according to the block types with varying depth and width. Note that the number of parameters of all experimented models are about one million.

Table 3.1: Quantization performance of various model structures on CIFAR-10 testset. ‘ L ’, ‘ D_{init} ’, and ‘ B ’ represent the number of layers, initial channel dimension, and number of blocks, respectively.

Block type	Model				Test Acc(%)			
	L	D_{init}	B	Params.	Float	W2AF	WFA2	W2A2
Basic block	8	59	1×3	1.05 M	93.26	92.83	92.58	92.32
	14	40	2×3	1.10 M	93.62	93.51	93.30	93.11
	20	31	3×3	1.03 M	94.14	93.57	92.94	92.65
	32	24	5×3	1.06 M	94.24	93.71	92.42	91.84
	44	20	7×3	1.04 M	94.27	93.91	91.40	90.69
	56	18	9×3	1.09 M	93.58	93.46	90.58	89.77
Pre-activation block	8	59	1×3	1.05 M	93.10	92.75	91.89	91.03
	14	40	2×3	1.10 M	94.14	93.85	92.44	91.81
	20	31	3×3	1.03 M	94.25	93.82	92.42	91.81
	32	24	5×3	1.06 M	94.07	93.89	91.51	89.03
	44	20	7×3	1.04 M	94.31	94.04	90.58	86.26
	56	18	9×3	1.09 M	93.96	93.83	88.45	85.45
Depthwise block	29	38	3×3	1.11 M	93.64	93.28	93.38	92.33
	38	32	4×3	1.08 M	93.65	93.47	92.93	92.57
	56	26	6×3	1.11 M	94.27	94.02	91.70	90.61
	74	22	8×3	1.08 M	93.94	93.71	90.50	89.88
	110	18	12×3	1.13 M	94.43	94.39	88.75	87.47

3.5.3 QDNN Architecture Selection under the Parameter Constraint

We compare the quantization sensitivity of CNNs according to the model structure under the constraint on the number of parameters. CNNs with three different block types: basic, pre-activation, depthwise blocks are evaluated using CIFAR-10 dataset. The number of parameters of these models is approximately one million. The performance of the quantized CNNs is shown in Figure 3.9. The experimented model configurations and results are summarized in Table 3.1. The number of parameters in all the experimented models is approximately one million. All CNNs consist of three block groups. ‘ B ’ indicates the block structure; (number of blocks in each group) \times (number of groups). The first blocks in second and third block groups have a stride of 2. ‘ D_{init} ’ represents the channel width of the first layer and the channel width increases by 2 times at the beginning of second and third block groups. When the number of parameters is comparable, increasing the depth to a certain range helps to improve the performance of full precision and weight quantized models. However, the models with 2-bit activation quantization show poor performances when the depth increases beyond certain numbers. We can also find that depthwise blocks are more robust to the activation quantization. The 2-bit CNNs with the basic or pre-activation blocks show severe performance degradation when the depth is over 32 while the performance with depthwise blocks is improved until the depth of 38. For the CIFAR-10 dataset, the best performing quantized ResNet can be designed by choosing the depth of 14 with the basic blocks or 38 with the depthwise ones.

We change the depth and width of the ResNets and evaluate the quantization performance on the ImageNet dataset [83]. Data augmentation methods and hyper-parameters are the same as [13]. Pre-activation blocks are employed and the shortcut signals are quantized to 8 bits. 4-level 2-bit weight quantization is applied.

The experimented ResNet structures are compared in Table 3.2. Since ResNets for ImageNet classification have four groups of residual blocks, which are separated by the convolution layers with the stride of 2, the number of blocks is represented as a

Table 3.2: Performance comparison of 2-bit ResNet on ImageNet validation set. ‘ L ’ and ‘ D_{init} ’ represent the number of layers and initial channel dimension, respectively. ‘ B ’ is the stack of the residual blocks which is separated by the convolution layers with the stride of 2.

Type	Model					Top-1 Acc (%)		
	L	D_{init}	B	Params. (10^6)	Ops. (10^9)	Float	W2A2	Diff.
ResNet34 [84]	34	64	[3, 4, 6, 3]	21.8	3.7	73.8	69.8	4.0
ResNet34 [85]	34	64	[3, 4, 6, 3]	21.8	3.7	73.8	70.0	3.8
ResNet34	34	64	[3, 4, 6, 3]	21.8	3.7	73.6	70.5	3.1
ResNet18 \times 1.4	18	90	[2, 2, 2, 2]	22.8	3.5	72.6	70.2	2.4
ResNet26 \times 1.1	26	70	[3, 3, 3, 3]	21.4	3.3	73.0	70.6	2.4
ResNet50 [84]	50	64	[3, 4, 6, 3]	25.6	4.1	76.4	71.5	4.9
ResNet50	50	64	[3, 4, 6, 3]	25.6	4.1	76.3	72.7	3.6
ResNet44 \times 1.1	44	72	[3, 4, 5, 2]	25.0	4.6	75.5	73.4	2.1
ResNet101	101	64	[3, 4, 23, 3]	44.6	7.9	77.5	20.1	57.4
ResNet50 \times 1.3	50	85	[3, 4, 6, 3]	44.2	7.2	77.2	73.7	3.5

list: [first, second, third, fourth] groups. For example, ResNet18 \times 1.4 means that the depth is 18 with 8 number of blocks and the initial channel width is 90. The number of channels increases by a factor of 2 at every convolution layer with the stride of 2.

The performance degradation by the quantization is much lower when the model is shallow under a comparable number of parameters. However, the top-1 accuracy of 2-bit ResNet18 \times 1.4 is 0.3% lower than the 2-bit ResNet34 because the floating-point performance is too low. When the depth, L , is 26 and the initial channel width,

D_{init} , is 70, the top-1 accuracy of the floating-point model is 0.6% lower but the 2-bit quantized model achieves 0.1% higher top-1 accuracy when compared to the ResNet34. The performance improvements are more significant when the model is deeper. The top-1 accuracy degradation of 2-bit ResNet50 is 4.9 [84] and 3.6(Ours). However, 2-bit ResNet44×1.1 shows 2.1% top-1 accuracy drop. As a result, 2-bit ResNet44×1.1 achieves the 0.7% top-1 accuracy improvement compared to the ResNet50 with a similar number of parameters and operations. The floating-point ResNet101 shows the top-1 accuracy of 77.5%, which outperforms shallow and wider models with a comparable number of parameters. However, the 2-bit quantization of ResNet101 degrades the performance significantly, showing only about 20.1% top-1 accuracy. As observed in Section 3.5.2, the 2-bit activation quantized model degrades the performance dramatically when the model is very deep. On the other hand, the ResNet with the depth of 50 achieves 73.5% top-1 accuracy even after 2-bit quantization. These results indicate that the performance of QDNNs can be improved simply by designing the proper depth and width of the model.

3.5.4 Results of Training Methods on QDNNs

We assess the effects of the training methods on QDNN optimization on PASCAL VOC segmentation and CIFAR-10 classification tasks. The cycle period and base LR for each dataset are as follows:

PASCAL VOC segmentation: The last LR of the retraining becomes 0 with the polynomial policy for this task. We set the base LR to 1e-5, which is 10 times smaller than the initial LR of the retraining. The cycle period is 3K iterations.

CIFAR-10: The base LR is 1e-4, which is 10 times larger than the last LR of the retraining. The cycle period is 8 epochs, thus the LR scaling factor changes at every epoch.

The effects of applying the CLR on the segmentation task for improved generalization and adding the regularization term, L_{Lip} , for noise robustness are summarized

Table 3.3: Performance improvements of quantized MobileNetV2 on the PASCAL VOC 2012 validation set. n_W and n_A represent the precision of the weights and activations, respectively.

Method	n_W	n_A	mIOU
Pretrained model	Float	Float	76.97
Retrain (baseline)	4-bit	Float	73.63
Fine-tune with CLR	4-bit	Float	74.15
Retrain with L_{Lip}	4-bit	Float	73.22
Retrain (baseline)	Float	4-bit	74.79
Fine-tune with CLR	Float	4-bit	74.71
Retrain with L_{Lip}	Float	4-bit	74.99

in Table 3.3. The mIOU of the floating-point pretrained model is 76.97, that of the retrained 4-bit weight is 73.63, and that of retrained 4-bit activation is 74.79. CLR is applied with a period of 3K iterations and fine-tuning was performed for 15K iterations (i.e. 5 cycles). L_{Lip} is added to the loss after multiplying the scaling factor of $1e-4$. Applying the CLR increases the mIOU of the 4-bit weight quantized model by 0.5 but it is not effective for the activation quantized model. Retraining with the regularization term that reduces the Lipschitz constant improves the mIOU of the 4-bit activation quantized model. However, the performance of the weight quantized model is decreased when L_{Lip} is applied. The performances of QDNNs when both weights and activations are quantized are shown in Table 3.4. The results show that the proposed approach for reducing the quantization effects of weights by CLR and activations by adding the Lipschitz loss works well when the precision is equal to or lower than 6-bit. But these techniques are not effective when the precision of quantization is 8-bit or larger. Even, the L_{Lip} regularization degrades the performance of the 8-bit quantized model.

We also evaluate the effects of the training methods for the classification task. The

Table 3.4: Performance in terms of mIOU on the PASCAL VOC 2012 validation set when both weights and activations are quantized.

Method	n_W / n_A (bits)			
	8/8	6/6	4/4	3/3
Retrain (baseline)	76.56	75.79	71.16	59.64
Retrain (L_{Lip})	76.34	76.23	71.93	59.85
L_{Lip} + CLR	76.46	76.40	72.56	60.74

Table 3.5: CIFAR-10 test accuracy (%) improvements when retrained with L_{Lip} and fine-tuned using CLR.

n_W, n_A	ResNet depth			
	20	32	56	110
Float	92.42	92.99	93.67	94.07
2-bit	89.36	90.00	89.13	83.71
2-bit (L_{Lip} + CLR)	89.68	90.60	90.24	87.77

performance degradation by severe quantization can be alleviated with L_{Lip} and CLR as shown in Table 3.5. L_{Lip} is added to the loss with a factor of $1e-4$ and CLR is applied for 40 epochs with the learning rates between $1e-3$ and $1e-5$. Although we can improve the performance of QDNN considerably for deep networks by applying CLR and L_{Lip} constraint, the best performing QDNN can be found when the depth is 32.

3.6 Concluding Remarks

We presented a holistic approach for the optimization of quantized deep neural networks (QDNNs). We first visualized the effects of the weight and activation quantization error using a synthetic dataset. The result clearly indicated that the effects of weight

and activation quantization are different. Especially, activation quantization severely degrades the performance of deep models. Through additional experiments with real tasks, we confirmed that the optimal model structure under a parameter constraint is different for the full-precision and quantized DNNs because floating-point models usually prefer the deep networks but QDNNs tend to show improved performances on wide networks. We also showed the effects of the DNN training schemes for improved generalization and noise reduction to optimize QDNNs. The proposed holistic approach can yield much better QDNN when compared to the conventional design approaches that start from the best performing floating-point models and optimize them using elaborate quantization and training methods.

Chapter 4

Parameter Shared Stochastic Precision Knowledge Distillation for Quantized Deep Neural Networks

4.1 Introduction

Deep neural networks (DNNs) have achieved remarkable accuracy for tasks in a wide range of applications, including image processing [1], machine translation [2], and speech recognition [3]. These state-of-the-art neural networks use very deep models, consuming hundreds of ExaOps of computation during training and GBytes of storage for model and data. This complexity poses a tremendous challenge for widespread deployment, especially in resource-constrained edge environments, leading to a plethora of explorations in model compression that minimize memory footprint and computational complexity while attempting to preserve the performance of the model. Among them, research on quantized DNNs (QDNNs) focuses on quantizing key data structures, namely weights and activations, into low-precision. Hence, we can save memory access overhead and simplify the arithmetic unit to perform reduced-precision computation. There have been extensive studies on QDNNs [61, 10, 23, 62], but most of them suffer from accuracy loss due to quantization [27].

To enhance the performance of low-capacity models, knowledge distillation (KD)

Table 4.1: Test accuracy (%) in higher precision on the quantized model. Two ResNet20 models are trained with 2-bit weight or activation quantization using the CIFAR-100 dataset, then bit-precision higher than 2-bit is used for inference. This reveals interesting phenomena that weight and activation quantization affects the model differently.

Trained precision	Inference precision			
2-bit W, float A	W2AF	W4AF	W8AF	WFAF
(W2AF)	65.74	58.01	55.85	54.70
Float W, 2-bit A	WFA2	WFA4	WFA8	WFAF
(WFA2)	66.93	68.48	68.77	68.71

has been widely adopted [25, 86]. KD employs a more accurate model as a teacher network to guide the training of a student model. For the same input, the teacher network provides its prediction as a soft label, which can be further considered in the loss function to guide the training of the student network. In the case of QDNNs, the quantized student network can compensate for its accuracy loss via supervision of the teacher model [87, 21, 88, 89]. However, the need for large and high-performance teacher models introduces significant overhead when applying KD. In particular, KD has not been successfully employed in the emerging study of on-device training for model adaptation and transfer learning, since the memory-intensive teacher models may not be available once the quantized models are deployed.

In this chapter, we propose a new practical approach to KD for QDNNs, called stochastic precision ensemble training for QDNNs (SPEQ). SPEQ is motivated by an inspiring observation on activation quantization. Table 4.1 shows that the accuracy of the FWA2 (float weight and 2-bit activation) model improves as the activation precision increases. However, the W2AF (2-bit weight and float activation) model shows the opposite characteristic. The accuracy drops as the inference precision increases. This

simple experiment reveals interesting insights regarding interpretation: the activation quantization only adds noise to the decision boundary [90]. Therefore, inference in the higher bit-precision results in the removal of such noise, leading to higher accuracy.

In SPEQ, we form a teacher network that shares the quantized weights with the student but employs different bit precision for activation. The clipping levels of activation are also shared. In fact, the activation precision for the teacher is randomly selected between the low and high precision, such as 2 and 8-bit. Since the teacher stochastically applies the target low-bit activation quantization for its soft label computation, it can experience the impact of quantization for the guidance. Furthermore, we reveal that the cosine similarity loss is essential for distilling the knowledge of the teacher of stochastic quantization to the low-precision student.

Although this form of guidance resembles KD, there is a significant difference in that the same model is shared and any other auxiliary models, such as large teacher networks, are unnecessary. Thus, SPEQ demands lower training costs compared to existing KD methods. In addition, the forward-pass computation of the teacher and student in SPEQ can be performed economically as the same weight parameters can be loaded only once.

We demonstrate the superior performance and efficiency of our SPEQ on various applications, including CIFAR-10/CIFAR-100/ImageNet image classification and also transfer learning scenarios such as BERT-based question-answering and Flower classification.

The contributions of our work are summarized as follows:

- We propose a new practical KD method called SPEQ that can specifically enhance the accuracy of QDNNs without any extra teacher models.
- We suggest cosine similarity as an essential loss function to effectively distill the knowledge of activation quantization in SPEQ training.
- We demonstrate that the proposed method outperforms the existing KD methods for training QDNNs with lower training overhead. We confirm this on various

models and tasks including image classifications, question answering, and transfer learning.

4.2 Background and Related Works

4.2.1 Quantization of Deep Neural Networks

QDNNs have been studied for a long time. Early works suggested stochastic gradient descent (SGD)-based training for QDNNs to restore the performance reduced by the quantization error [10, 9]. The quantized SGD training maintains both full-precision and quantized weights. Full-precision weights are exploited to accumulate the gradients, and the quantized weights are used for computing forward and backward propagation. Several techniques have been combined with the quantized SGD algorithm, which include data distribution [91], stochastic rounding [64], weight cluster [24], trainable quantization [84], fittable quantization scale [92], pow2-ternaization [93], stochastic weight averaging [18], increasing the size of the neural network [94], and quantization interval learning [13]. Our method is motivated by the observation that the quantization errors for weight and activation are different [90].

As a motivating example to understand the difference between activation and weight quantization, we conducted a simple experiment where the ResNet20 model is trained on the CIFAR-100 with either weight only or activation only quantization (into 2-bit). The two quantized models (W2AF or WFA2) were then employed for inference, where the same of higher bit-precision (2-8-bit) was used. The resulting inference accuracy is summarized in Table 4.1. For a higher precision of the activation quantization, we used the same clipping value, and thus the number of discretization points increases while the representation range is consistent. To increase the discretization points of 2-bit weights, we quantize the full-precision weights that are used to accumulate the gradients [10, 9]. We assume not to abandon any quantization points.

Interestingly, the weight quantized and the activation quantized models exhibit

vastly different behavior for the higher precision inference. The weight quantized model suffers significant accuracy loss when the bit-precision increases. Whereas, the activation quantized model recovers its accuracy as the bit-precision used in inference is increased. This simple experiment reveals very interesting insights into interpreting: the activation quantization just adds noise to the decision boundary [90]. Therefore, inference in the higher bit-precision results in the removal of such noise, leading to higher accuracy.

4.2.2 Knowledge Distillation for Quantization

In this setup, the teacher provided its guidance in terms of the soft label so that the student can optimize its parameters based on both hard and soft labels. Since KD has shown promising accuracy improvement, it has been utilized in various domains, including deep learning applications [95, 96, 97, 98] and learning algorithms [99, 100, 101, 102]. Some other work employed intermediate representation matching techniques such as layer-wise regularization along with KD to further enforce the similar output activation of hidden layers between the teacher and the student models [99, 103]. Recent studies tried to reduce the overhead for preparing a large teacher network [104, 105, 106]. Born-again networks [104] showed that the final trained student model could outperform the teacher network by repeatedly applying the KD technique. Online knowledge distillation methods are also proposed that train multiple student models simultaneously [105, 106].

Knowledge distillation (KD) is a method to improve the accuracy of a target model (called a student) by transferring better representation power (i.e., "knowledge") of a larger or more complex model (called a teacher) [25, 86]. Recently, several papers have adopted KD to restore the accuracy loss due to the quantization error of reduced-precision inference [107, 21, 87, 88, 89]. Apprentice [87] proposed three approaches to apply KD for enhancing the accuracy of the quantized models: 1) train the full-precision teacher and the quantized student model jointly from scratch, 2) train from scratch only

the quantized model whereas the teacher is fixed as the full-precision pre-trained model, and 3) both the teacher and student networks are pre-trained independently in full-precision, and only fine-tune the student model using the KD in the quantized domain. The importance of the hyper-parameters of KD, such as the relative importance of loss terms, was studied in [88]. More recently, quantization aware knowledge distillation [89] (QKD) is suggested, where the three training phases are coordinated as self-studying, co-studying, and tutoring.

The difference of the proposed method from the previous works is that the teacher and the student model are shared, while the outputs of the teacher are computed in stochastically high precision. There is two main advantage of this method; the teacher information contains the quantization noise induced in the target QDNN by ensemble model sharing (better performance) and pretrained teacher models or auxiliary training parameters are unnecessary (low training cost).

4.3 Stochastic Precision Ensemble Training for QDNNs

4.3.1 Quantization Method

We first introduce the quantization method that we use in this chapter. We employed a layer-wise uniform symmetric quantizer for efficiency. This quantizer consists of clipping and quantization that limit the range of variables. Weights or inputs in the same layer share a trainable scalar clipping value as suggested in [23, 13].

We employ PACT [23] for the n -bit quantization of rectified linear unit (ReLU) as follows:

$$\hat{x} = 0.5(|x| - |x - \alpha_x| + \alpha_x) \quad (4.1)$$

$$Q(x) = \lceil \hat{x} \cdot \left(\frac{2^n - 1}{\alpha_x} \right) \rceil \frac{\alpha_x}{2^n - 1}, \quad (4.2)$$

where $\lceil \cdot \rceil$ is the rounding operation. For example, 2-bit quantized activation output is one of $\{0, \alpha_x/3, 2\alpha_x/3, \alpha_x\}$. Gradients for $\lceil \cdot \rceil$ are calculated using the straight-through

estimator (STE) [108] and α_x is trained according to the following back-propagated gradients:

$$\frac{\partial L}{\partial \alpha_x} = \frac{\partial L}{\partial Q(x)} \frac{\partial Q(x)}{\partial \alpha_x} = \begin{cases} \frac{\partial L}{\partial Q(x)} & x > \alpha_x \\ 0 & \textit{else} \end{cases} \quad (4.3)$$

When all activation outputs are less than α_x , gradients for α_x becomes 0. Therefore, L2 regularization is applied to α_x so that the clipping happens at the proper range.

For weight quantization, we slightly modify the PACT algorithm as follows:

$$\hat{w} = 0.5(|w + \alpha_w| - |w - \alpha_w|) \quad (4.4)$$

$$\hat{w}' = \frac{\hat{w}}{2\alpha_w} + 0.5 \quad (4.5)$$

$$Q(w)' = \lceil \hat{w}' \cdot (2^n - 1) \rceil / (2^n - 1) \quad (4.6)$$

$$Q(w) = 2\alpha_w(Q(w)' - 0.5). \quad (4.7)$$

For example, the 2-bit weights are 4-level quantized values composed of $\{-\alpha_w, -\alpha_w/3, \alpha_w/3, \alpha_w\}$. Similar to the PACT activation quantization, the gradients for α_w becomes:

$$\frac{\partial L}{\partial \alpha_w} = \begin{cases} \frac{\partial L}{\partial Q(w)} & x > \alpha_x \\ -\frac{\partial L}{\partial Q(w)} & x < -\alpha_x \\ 0 & \textit{else} \end{cases} \quad (4.8)$$

4.3.2 Stochastic Precision Self-Distillation with Model Sharing

Changing the activation quantization precision in the same model affects the amount of noise injected into the model, as shown in Table 4.1. That is, the outputs obtained through high-precision activation have information when the model is operated without noise.

The training procedure of the SPEQ is illustrated in Figure 4.1. Two outputs were computed through different paths using the same parameters. Note that the initial

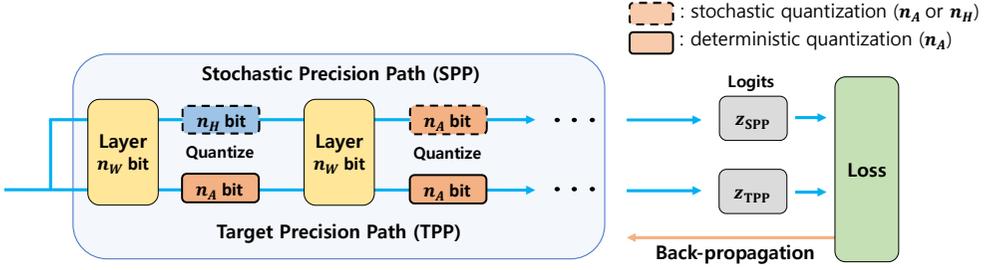


Figure 4.1: Structure of the proposed SPEQ training scheme for QDNNs. The QDNNs are trained for the target precision n_A through the ‘target precision path’. The ‘stochastic precision path’ produces the teacher logits, \mathbf{z}_{SPP} using the same model but with randomly assigned quantization precision for activation at every iteration. Note that the weights in the model can also be quantized to n_W bits.

quantized weights and clipping levels for activation were determined using conventional QDNN optimization methods [13, 23]. The details of the employed quantization method are shown in Appendix B. The first output logits, \mathbf{z}_{TPP} , were obtained through the target precision path by quantizing the activation outputs to n_A bits. The goal of the SPEQ is to increase the performance of the QDNN with this target precision path. The second output logits, \mathbf{z}_{SPP} , were computed by quantizing the activation outputs using the stochastic bit precision, n_{SPP} which is defined as follows:

$$n_{\text{SPP}}^l = \begin{cases} n_A & \text{with probability } u \\ n_H & \text{with probability } 1 - u, \end{cases} \quad (4.9)$$

where l denotes the layer index, n_A is the target precision, n_H is a precision higher than the target precision, and u is a quantization probability for the stochastic quantization path. We set the high precision, n_H , to 8 bits. The impact of u is discussed in the next section. For readability, we denote the set of n_{SPP}^l as \mathbf{n}_{SPP} , that is, $\mathbf{n}_{\text{SPP}} = \{n_{\text{SPP}}^1, \dots, n_{\text{SPP}}^L\}$.

The output probability, $\mathbf{p}((z), \mathcal{T})$, was computed using the softmax operation with temperature, \mathcal{T} . The temperature softened the distribution of the softmax outputs by

dividing the output logits [25]. Note that the soft labels, $\mathbf{p}(\mathbf{z}_{\text{SPP}}, \mathcal{T})$, were produced while sharing the parameters. Thus, they contain information of the quantization noise of the target model. We trained the QDNN using these soft labels to reduce the activation quantization noise. The loss for the SPEQ training is the sum of the cross-entropy loss, \mathcal{L}_{CE} , and the cosine similarity loss, \mathcal{L}_{CS} , as follows.

$$\mathcal{L}_{\text{SPEQ}} = \mathcal{L}_{CE}(y, \mathbf{p}(\mathbf{z}_{\text{TPP}}, 1)) + \mathcal{L}_{CS}(\mathbf{p}(\mathbf{z}_{\text{SPP}}, \mathcal{T}), \mathbf{p}(\mathbf{z}_{\text{TPP}}, \mathcal{T})) \times \mathcal{T}^2. \quad (4.10)$$

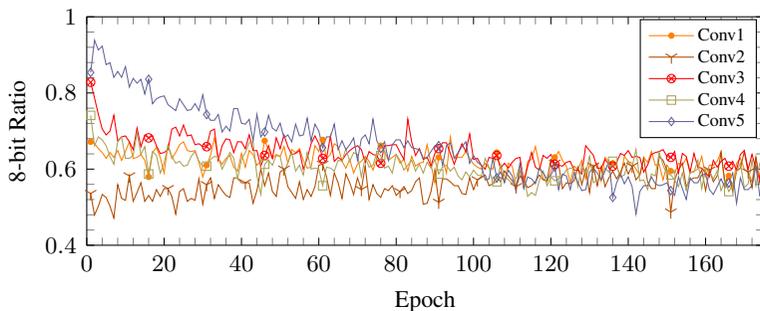
The effects of the cosine similarity loss function are discussed in Section 4.3.4. Note that the \mathbf{z}_{SPP} is only used to produce the soft label for the \mathcal{L}_{CS} , thus, the back-propagation error only flows through the target precision path, as shown in Figure 4.1. Therefore, the only computational overhead for a training step is the computation of \mathbf{z}_{SPP} by forward propagation. The SPEQ is based on the KD training but has the advantage that no other auxiliary model is required.

4.3.3 Stochastic Ensemble Learning

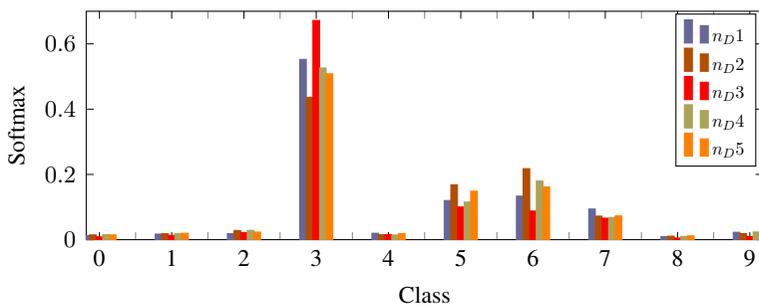
Intuitively, using a good teacher when training QDNN with KD will improve the performance [109]. In the proposed method, the activation quantization precision, \mathbf{n}_{SPP} , for each layer was determined stochastically. In this case, the total number of combinations for \mathbf{n}_{SPP} is 2^L , and among them, there will be the solution, $\mathbf{n}_{\text{SPP}}^*$, which shows the best performance. However, finding this solution is not practical for DNNs because 2^L inferences are needed for an exhaustive search. To investigate the performance of the best solution, we designed a shallow CNN that consists of five convolutional layers and train the model by selecting the \mathbf{n}_{SPP} for each step as follows:

$$\mathbf{n}_{\text{SPP}}^* = \underset{\mathbf{n}}{\operatorname{argmin}} \mathcal{L}_{CE}(y, \mathbf{z}_{\text{SPP}}|\mathbf{n}). \quad (4.11)$$

Since the experiment was performed on a five-layer CNN, $L = 5$, we employed the greedy strategy that finds the $\mathbf{n}_{\text{SPP}}^*$ by inferencing the model 2^5 times with different combinations of the quantization precision. Note that $\mathbf{n}_{\text{SPP}}^*$ can change for each training



(a)



(b)

Figure 4.2: (a) The ratio of selected 8-bit precision when trained with the greedy strategy. (b) Softmax distributions generated by the stochastic precision path with the same images.

step. The target model is trained using the soft label obtained with n_{SPP}^* . Figure 4.2 (a) shows how the ratio of 8-bit selection changes during training for each layer. The model is pretrained to the 2-bit weights and activations and the target precision is also 2 bits. The floating-point and 2-bit models show accuracies of 89.9% and 87.8%, respectively.

Surprisingly, the solution of Eq. (4.11) is not always 8-bit even at the beginning of the training. Note that the results in Table 4.1 show that using higher precision for the activation can achieve higher average accuracy. For each iteration, however, choosing 8-bit activation may not show the lowest loss. More importantly, the ratio of 8-bit selection decreases to 0.6 as the training progresses. This indicates that the best-

performing solution, $\mathbf{n}_{\text{SPP}}^*$, selects 2- and 8-bit almost uniformly. As a result, the test accuracy of the 2-bit model with greedy training is 88.4%, which is a better performance of always choosing 8-bit, 88.1%.

Another advantage of the SPEQ is that it has the effect of ensemble learning. By the stochastic selection of bit precision, soft labels with different distributions can be created for one training sample. In this case, the diversity of soft labels should be large enough to obtain the effect of the ensemble well [110]. Figure 4.2 (b) shows the computed soft labels by quantizing the activation outputs of the ResNet20 with different bit precisions for a single training sample in the CIFAR-10 dataset. Although we extract soft labels from the same parameters, the distribution of the soft labels varies according to the activation precision.

Based on our analysis, we apply the SPEQ training method with uniform n_A -bit and 8-bit selection probabilities to increase the diversity. The sensitivity of the quantization probability, u , is also examined in Section 4.4.

4.3.4 Cosine Similarity Learning

In many KD approaches, KL-divergence is commonly used as a loss function to reflect the guidance of the teacher. In the setting of SPEQ, however, we claim that the cosine similarity loss (CS-Loss) function is better than KL-divergence loss (KL-Loss). The main difference is that the teacher in SPEQ may not be more reliable than the student. Note that activations are randomly quantized in SPEQ, thus the output prediction of the teacher might be significantly affected by the quantization noise. In this setting, it is important to reflect the guidance of the teacher selectively, as there is no guarantee that the teacher’s prediction is more accurate than the student’s. In this section, we explain that CS-Loss can consider the confidence of the student whereas KL-Loss does not.

To understand the situation more concretely, we compared the back-propagation errors (i.e., gradients w.r.t. each logit) for the two loss functions. Note that the student model is guided to increase (or decrease) the logit if the corresponding gradient is

Table 4.2: Comparison of the test accuracy of 2-bit ResNet20 on CIFAR-10 according to the loss function for KD. The cosine similarity loss (CS-Loss) suits better than the KL-divergence loss (KL-Loss) for the proposed SPEQ method. Average test accuracy of 5 repeated experiments is reported with the standard deviation.

Method	Loss type	Test acc
2-bit baseline	CE-Loss	90.73
KD w/ full precision ResNet18 as teacher	CE + KL-Loss	91.24±0.06
	CE + CS-Loss	91.22±0.10
SPEQ ($u = 0$, always choose 8-bit for soft labels)	CE + KL-Loss	91.22±0.16
	CE + CS-Loss	91.18±0.07
SPEQ ($u = 0.5$, 2-bit or 8-bit for soft labels)	CE + KL-Loss	90.83±0.07
	CE + CS-Loss	91.44±0.04

negative (or positive). When the predictions of the teacher and the student are \mathbf{p} and \mathbf{q} , respectively, the KL-Loss and its back-propagation error for the i^{th} logit, z_i , is represented as follows [25]:

$$\mathcal{L}_{KL}(\mathbf{p}, \mathbf{q}) = - \sum_{i=0}^{C-1} p_i \log \frac{q_i}{p_i}, \quad (4.12)$$

$$\frac{\partial \mathcal{L}_{KL}}{\partial z_i} = q_i - p_i. \quad (4.13)$$

Eq. (4.13) indicates that the \mathcal{L}_{KL} produces back-propagation errors in the direction of decreasing the difference between the p_i and q_i . That is, KL-Loss guides the student to always follow the teacher. Such guidance is regarded as "re-weighting" [109], but it is helpful under a condition that the teacher's prediction is more confident than the student's. Since the teacher in SPEQ is not as reliable as the large teacher models in typical KD methods, the gradients from KL-Loss can be misleading.

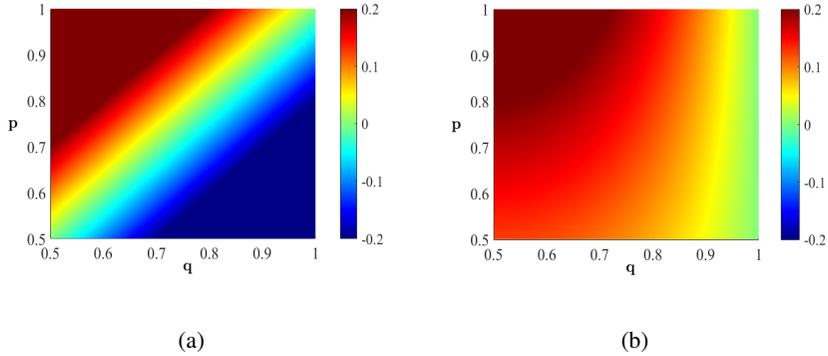


Figure 4.3: The gradients for the (a) KL loss and (b) cosine loss according to the student probability (x-axis) and teacher probability (y-axis). The direction change on gradients of KL loss is due to the blue region.

In comparison, CS-Loss between predictions of the teacher and the student after the normalization is given as follows:

$$\mathcal{L}_{CS}(\mathbf{p}, \mathbf{q}) = 1 - \mathbf{p} \cdot \mathbf{q}, \quad (4.14)$$

$$\frac{\partial \mathcal{L}_{CS}}{\partial z_i} = - \sum_{j=0}^{C-1} p_j (q_j \delta_{ij} - q_j q_i). \quad (4.15)$$

The gradients of \mathcal{L}_{CS} are more cognizant of the confidence of both the teacher and the student. Assume that the i^{th} label is the ground-truth and the teacher's prediction is also confident about it, i.e. $p_i \gg p_j$ ($i \neq j$), Eq. (4.15) is approximated as follows:

$$\frac{\partial \mathcal{L}_{CS}}{\partial z_i} \approx -p_i q_i (1 - q_i). \quad (4.16)$$

Eq. (4.16) indicates that the gradients is proportional to $q_i(1 - q_i)$. This is particularly helpful when the confidence of the student's prediction is not high; when $0 < q_i < 1$, the student is guided to increase the confidence for q_i .

In addition, as the prediction of the teacher itself is ambiguous, the back-propagation error decreases. When the prediction of the teacher goes to a uniform distribution,

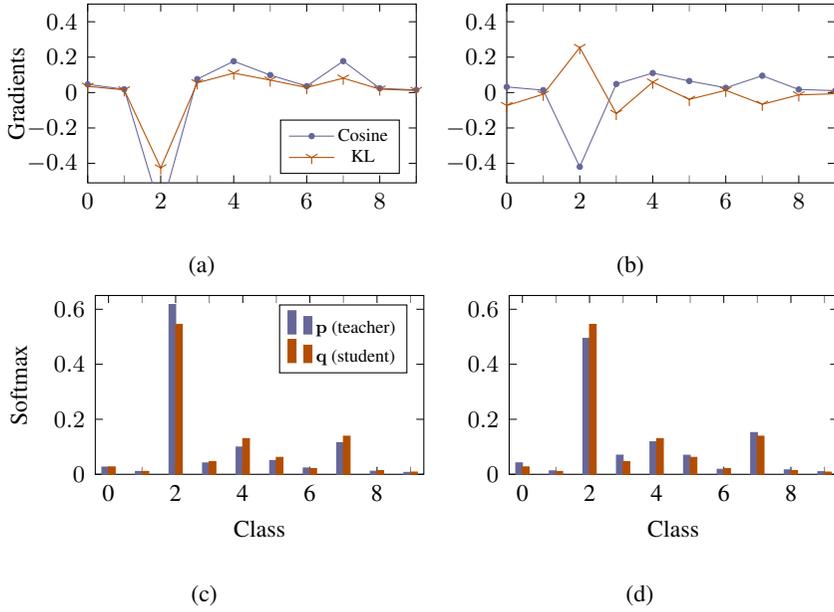


Figure 4.4: Examples of (a) gradients and (c) softmax outputs when the teacher is more confident than the student. (b,d) Different direction of the gradients when the teacher is less confident.

Eq. (4.15) can be approximated as:

$$\frac{\partial \mathcal{L}_{CS}}{\partial z_i} \approx -p_i q_i \left(1 - \sum_{j=0}^{C-1} q_j\right) = 0. \quad (4.17)$$

Note that the gradients, in this case, are almost zero, implying that the CS-Loss will be neglected when the confidence of the teacher’s prediction is small.

The impact of the relationship between the teacher and the student predictions to the gradients is illustrated in Figure 4.3. As can be seen, the gradient of KL-Loss flips its direction when the student’s confidence is higher than the teacher’s. This is detrimental for SPEQ-based knowledge distillation as the prediction of the teacher is prone to noise. Whereas, the CS-Loss allows selective adoption of the teacher’s information; the gradients guide to follow the teacher more if it has high confidence. If not, the guidance is neglected.

The proposed SPEQ method computes a teacher output with randomly selected activation precision. In this case, a bad teacher outputs, i.e., less confident to the ground-truth, can be generated. Figure 4.4 (a) and (b) show examples of softmax outputs computed with different activation precision for one training sample. Those examples are the inference results on 2-bit ResNet20 for the sample where the ground-truth label is 2. The probability of the student, q , is calculated with 2-bit activation through the target precision path (TPP). Teacher probability, p , is the softmax output computed through the stochastic precision path (SPP). Depending on the activation precision, the confidence of the teacher outputs for the ground-truth can be higher or lower than the student output. When more confident teacher output is selected, the gradients for the student’s logits are shown in Figure 4.4 (c). Conversely, gradients with a less confident teacher output are shown in Figure 4.4 (d).

The CS-Loss and KL-Loss produce similar gradients when the teacher’s confidence in the ground-truth is higher than the student’s. However, when the confidence of the teacher is lower than that of the student, KL-Loss creates a positive gradient for the logit corresponding to the ground-truth. Note that this gradient lowers the student’s logit for the ground-truth. Therefore, the direction of gradients changes depending on the selected teacher. This hinders training the student model in a consistent direction. In results, the effects of KD diminish with the proposed SPEQ method when using the KL-Loss. Experimentally, applying KL-Loss to SPEQ (90.83%) showed similar performance compared to the training without KD (90.73%). The results in the parentheses are shown in Table 4.2.

On the other hand, the direction of the gradient for the ground-truth logit does not change when CS-Loss is employed. This is because the teacher’s probability acts as a scaling factor. When the confidence of the teacher is small, the gradient for the ground-truth logit also becomes small as shown in Figure 4.4 (d). In results, employing CS-Loss instead of the KL-Loss showed much better performance for the proposed SPEQ method.

4.4 Experimental Results

4.4.1 Experiment Setup

QDNNs have shown better performance when initial parameters are set to a pretrained full-precision model [12, 13]. The all training procedures in our experiment follow the three steps; train the floating-point DNN (pretrain), train the QDNN to the target precision initialized from the floating-point parameters (retrain [9]), and train the QDNN using the SPEQ method initialization with the retrained parameters.

The ReLU6 [111] operation was used instead of ReLU when applied in floating-point pretrained models. The activation clipping value α_x was initialized to 6 for retraining. The weight clipping value α_w is initialized to the value that minimizes L2-distance before and after the quantization of pretrained weights using Lloyd-algorithm [9]. To reduce the variance in the training process, the 100 times lower learning rate was applied to α_w [13].

We also quantized the hidden features passing through the shortcut when the model adopts the residual connection. The shortcut outputs were quantized in the same way as the weight quantization. Because it is known that the shortcut signal quantization significantly degrades the performance compared to activation or weight quantization [112, 113], we quantized the shortcut signal to 8-bit. The clipping value for a shortcut, α_{sc} , is initialized to 6 for retraining. We quantized all weights and inputs of convolution layers except the first and last layers following [13]. For the SPEQ training, we initialized the training parameters from the retrained QDNN with the target precision. The training hyper-parameters for SPEQ training was the same as the QDNN training.

4.4.2 Results on CIFAR-10 and CIFAR-100 Datasets

We assessed the proposed method on VGG16, ResNet models, and MobileNetV2. VGG16 and ResNet20 are trained using the CIFAR-10 dataset. For CIFAR-100 dataset,

Table 4.3: 2-bit ResNet20 test accuracy according to the quantization probability for the stochastic path, u . Average test accuracy of 5 repeated experiments is reported with the standard deviation.

u	0.0	0.1	0.2	0.3
Test Acc.	91.18±0.07	91.23±0.10	91.22±0.04	91.29±0.12
u	0.4	0.5	0.6	0.7
Text Acc.	91.39±0.07	91.44±0.04	91.43±0.02	92.21±0.15
u	0.8	0.9	1.0	Mix
Text Acc.	91.24±0.08	90.96±0.08	90.74±0.12	91.23±0.13

ResNet32 and MobileNetV2 were employed. Training images were augmented by horizontally flipping and cropping [53]. All models for CIFAR-10 and CIFAR-100 datasets are trained using the same optimizer and hyper-parameters. The SGD optimizer was used with the momentum factor of 0.9 and the batch size was 128. We first trained full-precision models with the initial learning rate of 0.1. The learning rate decayed by the factor of 0.1 at 100 and 150 epochs. The total number of training epochs was 175. L2-loss was applied to the scale of $5e-4$. The hyper-parameters for the retraining and SPEQ methods are the same as follows. QDNNs are trained for 175 epochs with the initial learning rate of 0.01. The learning rate decreased by 0.1 times at 100 and 150 epochs. L2-loss was applied only for the activation clipping values with a scale of $5e-4$. The hyper-parameters for SPEQ methods were the same as that of retraining. The temperature, \mathcal{T} , was set to 5.0 and 3.0 for the CIFAR-10 and CIFAR-100 datasets, respectively. The ReLU6 operation was used instead of ReLU for developing floating-point pretrained models. The activation clipping value α_x was initialized to 6 for retraining. The weight clipping value α_w was initialized to the value that minimizes L2-distance before and after the quantization of pretrained weights using

Table 4.4: Test accuracy (%) of quantized CNNs on CIFAR-10 and CIFAR-100 datasets. ‘F’ denotes the floating-point precision.

Methods	Precision (W / A)	CIFAR-10		CIFAR-100	
		VGG16	ResNet20	ResNet32	MobileNetV2
Baseline	F / F	93.6	92.1	70.3	76.8
Retrain	2 / 2	92.5	90.7	66.9	73.0
PACT-SWAB-8brc [113]	2 / 2	-	90.7	-	-
QKD [89]	2 / 2	-	90.5	66.4	-
SPEQ	2 / 2	93.1	91.4	69.1	74.4
Retrain	F / 2	92.9	91.8	67.9	74.5
SPEQ	F / 2	93.5	92.1	69.7	75.2

Lloyd-algorithm [9]. To reduce the variance in the training process, the 100 times lower learning rate was applied to α_w [13].

We first investigated how the stochastic quantization probability, u , affects the performance of the SPEQ method. To this end, we train 2-bit quantized ResNet20 models with various values of u from 0.0 to 1.0 on the CIFAR-10 dataset. The results are reported in Table 4.3. It should be noted that 0.0 and 1.0 of u represent that the stochastic precision path selects only 8- and 2-bit precisions, respectively. The best test accuracy is observed when u is between 0.4 and 0.6. This result indicates that the SPEQ shows the best performance when the stochastic precision path is selected to some degree evenly rather than being biased to either precision. For comparison, the results of training by uniformly selecting all precisions between n_A and n_H instead of two is reported as ‘Mix’. The result is 0.21% lower than that of the training using only two precisions. This is because the softmax distributions corresponding to 4, 6,

Table 4.5: Top-1 accuracy of 2-bit activation quantized CNNs on the ImageNet dataset.

Method	AlexNet	ResNet18
Float baseline	60.8	70.3
BalancedQ [91]	56.5	62.1
QN [114]	-	65.7
DoReFa [†] [12]	54.1	66.9
PACT [23]	54.9	67.5
SPEQ	60.8	68.4

Results with the symbol [†] are from [23].

and 8 bits are very similar. For all the rest of the experiments, we set the quantization probability, u , to 0.5.

We evaluated the proposed SPEQ scheme using the CIFAR-10 and CIFAR-100 datasets. The performance of the SPEQ and existing methods are shown in Table 4.4. The test accuracy of 2-bit ResNet20 before applying SPEQ, denoted as ‘Retrain’, is 90.7% on the CIFAR-10 dataset. This result is similar to previous works. The SPEQ significantly improves the performance of 2-bit ResNet20 and achieves 91.4% test accuracy. This result is better than the QKD [89], which employs a large teacher. Furthermore, when only the activations are quantized to 2 bits, SPEQ shows almost the same performance as the full-precision models for CIFAR-10. The SPEQ shows consistent improvements on various CNNs.

4.4.3 Results on ImageNet Dataset

For the ImageNet dataset, we evaluated our method on AlexNet, ResNet18, and ResNet34. The SGD optimizer was employed with the momentum of 0.9. The learning rate for the full-precision training was 0.4 with the batch size of 1024. The initial

Table 4.6: Top-1 accuracy on the ImageNet dataset when both weights and activations are quantized to 2 bits.

Method	AlexNet	ResNet18	ResNet34
Float baseline	60.8	70.3	73.6
DoReFa [†] [12]	46.4	62.6	-
QIL [13]	58.1	65.4	70.6
PACT_SWAB [113]	57.2	67.0	-
Retrain	56.9	66.6	70.5
SPEQ	59.3	67.4	71.5

Results with the symbol [†] are from [23].

learning rate for low-precision training was 0.04. We trained all the models for 90 epochs and the learning rate decayed by a factor of 10 at 30, 50, 60, 70, and 80 epochs. The training images were augmented using random cropping and horizontal flipping. The input sizes for AlexNet and ResNet were 227×227 and 224×224 , respectively. For the AlexNet, the batch-normalization was used instead of layer-normalization and we changed the position of max-pooling and activation layer to find the max value before quantization [11]. The ReLU6 was employed instead of ReLU when training floating-point models. The initial clipping values for quantization were obtained using the same method as the CIFAR-10/CIFAR-100 settings. The temperature, \mathcal{T} was set to 1.0, 1.0, and 2.0 for AlexNet, ResNet18, and ResNet34, respectively.

The performance of the SPEQ on the ImageNet dataset is shown in Table 4.5 and Table 4.6. The retraining scheme shows 56.9%, 66.6%, and 70.5% top-1 accuracy for the 2-bit AlexNet, ResNet18, and ResNet34, respectively. By SPEQ training, the top-1 accuracy increases approximately 1% for ResNet18 and ResNet34. The SPEQ training on 2-bit AlexNet improves the top-1 accuracy noticeably, showing 59.3% top-1 accuracy. This result is only an 1.1% accuracy drop compared to the full-precision

Table 4.7: 2-bit ImageNet quantization Top-1 accuracy compared with other KD applied QDNNs. Flops and run-times are measured for a single update with the batch size of 64.

Method	ResNet18			
	Teacher	Flops(10^{12})	Time(ms)	Acc(%)
Retrain	w/o KD	0.70	111.7	66.6
AP [87]	ResNet34	1.17	162.8	66.8
QKD [89]	ResNet34	2.11	270.2	67.4
SPEQ	-	0.93	135.4	67.4
SPEQ+AP	ResNet34	1.38	198.8	67.8

Method	ResNet34			
	Teacher	Flops(10^{12})	Time(ms)	Acc(%)
Retrain	w/o KD	1.41	176.1	70.5
AP [87]	ResNet50	1.93	266.7	71.1
QKD [89]	ResNet50	2.98	481.5	71.6
SPEQ	-	1.87	220.3	71.5
SPEQ+AP	ResNet50	2.56	294.7	72.1

AlexNet. The results for 2-bit activation-quantized CNNs indicate that the proposed SPEQ method is very effective for reducing the activation quantization noise.

The efficiency of SPEQ training is also compared with other KD training methods for QDNNs in Table 4.7. The 2-bit ResNet performances are reported with the training costs. The total number of floating-point operations (flops) and time (ms) are measured for one iteration of the training procedure with the batch size of 64. The time consumption is measured using a single NVIDIA TITAN Xp GPU [115]. We ignored the overhead of training a large teacher network, which is advantageous for Apprentice and QKD. Note that the QKD requires more training times and memory footprints for the co-training of the teacher and student models.

Although our approach is based on the self-distillation, the larger teacher can also be employed to further improve the performance of the target model. We applied the KD by combining the Apprentice [87] (AP) and the SPEQ scheme. The KL-Loss was computed using the soft label of the large teacher and added to Eq. (4.10). The combined training improves the top-1 accuracy of ResNet18 and ResNet34 by 0.4% and 0.6%, respectively.

4.4.4 Results on Transfer Learning

Since our method is very simple and requires little computational overhead, it can be applied to the transfer learning from a very large model such as BERT [60]. We optimized the low-precision BERT using SPEQ training. The pretrained BERT-Base model was obtained from [116], and then fine-tuned using the Stanford Question Answering Dataset (SQuAD) [117]. We followed the quantization methods and hyperparameters for BERT proposed in [118]. All weights except the last output layer were quantized according to the weight quantization precision, n_W , and all hidden signals including the attention scores were quantized with the activation quantization precision, n_A . We fixed the precision of attention scores to n_A bits when computing soft labels because the stochastic quantization of attention scores rather decreases the performance

Table 4.8: The performance improvement with the SPEQ scheme on the question-answering task. The BERT model is quantized and fine-tuned using the SQuAD1.1 dataset. Results are evaluated using the SQuAD1.1 dev dataset.

Method	W3/A3		W4/A4	
	EM	F1	EM	F1
FixedBERT [118]	71.5	81.4	74.2	83.1
SPEQ	76.4	85.1	78.0	86.6

in our experiments. Note that the attention-scores were always quantized to n_A bits for the stochastic precision path (SPP). The temperature, \mathcal{T} , is set to 2.0 for all experiments.

The performance improvements of the SPEQ on quantized BERT are shown in Table 4.8. The fine-tuned floating-point BERT shows 81.1% F1 and 88.6% EM scores. When the activation is quantized to n_A bits, the stochastic precision for computing soft labels is chosen between n_A and 8 bits. Applying the SPEQ method on low-precision models significantly improves the performance of quantized BERT on the SQuAD1.1 dataset. As a result, the 4-bit BERT trained using SPEQ achieves 86.6% F1 score, which is only 2.0 drops from the full-precision model. The results show that the SPEQ training is also effective when retraining QDNNs with a smaller number of training samples compared to the pretraining samples.

We expanded the experiment for transfer learning using Oxford Flowers-102 [119]. The Oxford Flowers-102 dataset consists of 8189 images the number of labels of 102. The number of images for each class varies from 40 to 258. Considering the user-adaptation, we used a very small number of images as training samples, such as 5, 10, and 20 samples per label. The rest of the images were used as validation samples. All images were re-sized to 224×224 and normalized channel-wisely. Note that data augmentation methods such as random cropping and flipping are not applied.

Table 4.9: Performance comparison on the transfer learning according to the training method and the feature extractor. Values in the parentheses are the standard deviation of 5 training results.

Flowers-102 # samples per label	Float ResNet18 (CE loss)	2-bit ResNet18 (CE loss)	2-bit ResNet18 (SPEQ)
5 (610 training samples)	69.18±0.39	70.67±0.31	71.21±0.26
10 (1020 training samples)	78.04±0.30	77.99±0.16	78.36±0.14
20 (2040 training samples)	84.57±0.12	83.91±0.15	85.02±0.11

We employed the ResNet18 model trained using the ImageNet dataset as a feature extractor and it was frozen when fine-tuning. The last output layer was newly added and trained for 50 epochs with a batch size of 16. The SGD optimizer was employed and the learning rate is 0.1.

The SPEQ was evaluated by changing the number of training samples and the results are shown in Table 4.9. SPEQ improves the performance significantly compared to the 2-bit model retraining using the cross-entropy loss. Moreover, the SPEQ-trained 2-bit ResNet18 achieves better results than the floating-point model. In practice, the conventional KD method is hard to be applied due to the need for auxiliary models. Therefore, the SPEQ method is a good solution to apply KD on transfer learning.

4.5 Concluding Remarks

In this chapter, we proposed a novel KD method for quantized DNN training. The proposed method, SPEQ, does not require a cumbersome teacher model; it assigns the same parameters for the teacher and student networks. The teacher model is formed by assigning the stochastic precision to the activation of each layer, by which it can produce

the soft labels of stochastically ensembled models. The cosine similarity loss was used for KD training to render reliable operation even when the confidence of the teacher is lower than that of the student. The SPEQ outperforms the existing quantized training methods in various tasks. Furthermore, the SPEQ can be easily used for low-precision training even when no larger teacher model is available.

Chapter 5

Conclusion

In this dissertation, we analyzed the quantization errors on deep neural networks (DNNs). Specifically, the different characteristics of the weight and activation quantization errors were visualized. Based on the analysis, the stochastic precision ensemble distillation (SPEQ) technique was developed to train quantized DNNs (QDNNs). The SPEQ method can reduce the activation quantization noise of QDNNs with high efficiency by sharing model parameters for the teacher and student network.

We first revealed when and why the weight quantization degrades the performance of DNNs in Chapter 2. The memorization capacity of DNNs was formulated and measured by training DNNs using the uniform random training samples. We found that the model capacity is proportional to the number of parameters, and the per-parameter capacity was compared among the various model structures, such as fully-connected DNNs (FCDNNs), convolutional neural networks (CNNs), and recurrent neural networks (RNNs). The per-parameter capacity is highly related to the limitation of quantization bit precision, in which the parameters maintain the information. Therefore, the performance of the QDNN decreases when the weights are quantized under the bit limitation. By extensive simulation, we found that the weights can be quantized without the loss of information up to 5, 8, and 10 bits for FCDNNs, CNNs, and RNNs, respectively.

We also analyzed the activation quantization errors in Chapter 3. The different

characteristics of the weight and activation quantization errors were visualized using a synthesized dataset. We found that the activation quantization induces noise while the weight quantization reduces the capacity of DNNs. Both quantizations degrade the performance of DNNs, but their characteristics are disparate. Our results indicate that deeper models are more prone to the activation quantization because each layer's quantization noise is overlapped.

In Chapter 4, the stochastic precision ensemble QDNN (SPEQ) training technique was proposed. The SPEQ is a knowledge distillation (KD) based training, but the teacher model shares parameters with the student model. The soft label of the teacher was computed by changing the precision of activation. Because the bit-precision was chosen randomly at every iteration, the SPEQ has the effect of stochastic ensemble KD. The SPEQ improved the performance of QDNNs significantly without any auxiliary models such as cumbersome teachers. As a result, the SPEQ outperformed the existing quantized training methods in various tasks, such as image classifications, question-answering, and transfer learning.

Bibliography

- [1] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2016, pp. 770–778.
- [2] J. Gehring, M. Auli, D. Grangier, D. Yarats, and Y. N. Dauphin, “Convolutional sequence to sequence learning,” in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR.org, 2017, pp. 1243–1252.
- [3] Y. Zhang, M. Pezeshki, P. Brakel, S. Zhang, C. L. Y. Bengio, and A. Courville, “Towards end-to-end speech recognition with deep convolutional neural networks,” *arXiv preprint arXiv:1701.02720*, 2017.
- [4] F. Iandola, M. Moskewicz, S. Karayev, R. Girshick, T. Darrell, and K. Keutzer, “Densenet: Implementing efficient convnet descriptor pyramids,” *arXiv preprint arXiv:1404.1869*, 2014.
- [5] D. Han, J. Kim, and J. Kim, “Deep pyramidal residual networks,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 5927–5935.
- [6] D. Adiwardana, M.-T. Luong, D. R. So, J. Hall, N. Fiedel, R. Thoppilan, Z. Yang, A. Kulshreshtha, G. Nemade, Y. Lu *et al.*, “Towards a human-like open-domain chatbot,” *arXiv preprint arXiv:2001.09977*, 2020.

- [7] G. Dunder and K. Rose, “The effects of quantization on multilayer neural networks,” *IEEE Transactions on Neural Networks*, vol. 6, no. 6, pp. 1446–1451, 1995.
- [8] B. Jacob, S. Kligys, B. Chen, M. Zhu, M. Tang, A. Howard, H. Adam, and D. Kalenichenko, “Quantization and training of neural networks for efficient integer-arithmetic-only inference,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 2704–2713.
- [9] K. Hwang and W. Sung, “Fixed-point feedforward deep neural network design using weights +1, 0, and -1,” in *Signal Processing Systems (SiPS), 2014 IEEE Workshop on*. IEEE, 2014, pp. 1–6.
- [10] M. Courbariaux, Y. Bengio, and J.-P. David, “Binaryconnect: Training deep neural networks with binary weights during propagations,” in *Advances in Neural Information Processing Systems (NIPS)*, 2015, pp. 3123–3131.
- [11] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, “Xnor-net: Imagenet classification using binary convolutional neural networks,” in *Proceedings of the European Conference on Computer Vision (ECCV)*. Springer, 2016, pp. 525–542.
- [12] S. Zhou, Y. Wu, Z. Ni, X. Zhou, H. Wen, and Y. Zou, “Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients,” *arXiv preprint arXiv:1606.06160*, 2016.
- [13] S. Jung, C. Son, S. Lee, J. Son, J.-J. Han, Y. Kwak, S. J. Hwang, and C. Choi, “Learning to quantize deep networks by optimizing quantization intervals with task loss,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 4350–4359.
- [14] W. Sung, S. Shin, and K. Hwang, “Resiliency of deep neural networks under quantization,” *arXiv preprint arXiv:1511.06488*, 2015.

- [15] D. Lin, S. Talathi, and S. Annapureddy, “Fixed point quantization of deep convolutional networks,” in *International Conference on Machine Learning*, 2016, pp. 2849–2858.
- [16] A. Ansari and T. Ogunfunmi, “Empirical analysis of fixed point precision quantization of cnns,” in *2019 IEEE 62nd International Midwest Symposium on Circuits and Systems (MWSCAS)*. IEEE, 2019, pp. 243–246.
- [17] S. Shin, J. Park, Y. Boo, and W. Sung, “Hlhlp: Quantized neural networks training for reaching flat minima in loss surface,” in *Thirty-Fourth AAAI Conference on Artificial Intelligence*, vol. 1, no. 2, 2020, p. 6.
- [18] S. Shin, Y. Boo, and W. Sung, “Sqwa: Stochastic quantized weight averaging for improving the generalization capability of low-precision deep neural networks,” *arXiv preprint arXiv:2002.00343*, 2020.
- [19] B. Neyshabur, Z. Li, S. Bhojanapalli, Y. LeCun, and N. Srebro, “Towards understanding the role of over-parametrization in generalization of neural networks,” *arXiv preprint arXiv:1805.12076*, 2018.
- [20] S. Anwar, K. Hwang, and W. Sung, “Fixed point optimization of deep convolutional neural networks for object recognition,” in *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2015, pp. 1131–1135.
- [21] A. Polino, R. Pascanu, and D. Alistarh, “Model compression via distillation and quantization,” in *International Conference on Learning Representations*, 2018. [Online]. Available: <https://openreview.net/forum?id=S1XolQbRW>
- [22] J. Lin, C. Gan, and S. Han, “Defensive quantization: When efficiency meets robustness,” *International Conference on Learning Representations (ICLR)*, 2019.

- [23] J. Choi, Z. Wang, S. Venkataramani, P. I.-J. Chuang, V. Srinivasan, and K. Gopalakrishnan, “PACT: Parameterized clipping activation for quantized neural networks,” *arXiv preprint arXiv:1805.06085*, 2018.
- [24] E. Park, J. Ahn, and S. Yoo, “Weighted-entropy-based quantization for deep neural networks,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 5456–5464.
- [25] G. Hinton, O. Vinyals, and J. Dean, “Distilling the knowledge in a neural network,” *arXiv preprint arXiv:1503.02531*, 2015.
- [26] N. Papernot, P. McDaniel, X. Wu, S. Jha, and A. Swami, “Distillation as a defense to adversarial perturbations against deep neural networks,” in *2016 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2016, pp. 582–597.
- [27] Y. Boo, S. Shin, and W. Sung, “Memorization capacity of deep neural networks under parameter quantization,” in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 1383–1387.
- [28] T. M. Cover, “Geometrical and statistical properties of systems of linear inequalities with applications in pattern recognition,” *IEEE transactions on electronic computers*, no. 3, pp. 326–334, 1965.
- [29] E. Gardner and B. Derrida, “Optimal storage properties of neural network models,” *Journal of Physics A: Mathematical and general*, vol. 21, no. 1, p. 271, 1988.
- [30] S. Akaho and S. Amari, “On the capacity of three-layer networks,” in *Neural Networks, 1990., 1990 IJCNN International Joint Conference on*. IEEE, 1990, pp. 1–6.

- [31] J. Collins, J. Sohl-Dickstein, and D. Sussillo, “Capacity and trainability in recurrent neural networks,” in *International Conference on Learning Representations (ICLR)*, 2017.
- [32] C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals, “Understanding deep learning requires rethinking generalization,” *Proceedings of the International Conference on Learning Representations (ICLR)*, 2017.
- [33] D. Arpit, S. Jastrzebski, N. Ballas, D. Krueger, E. Bengio, M. S. Kanwal, T. Maharaj, A. Fischer, A. Courville, Y. Bengio *et al.*, “A closer look at memorization in deep networks,” in *International Conference on Machine Learning*, 2017, pp. 233–242.
- [34] S. Shin, K. Hwang, and W. Sung, “Fixed-point performance analysis of recurrent neural networks,” in *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on*. IEEE, 2016, pp. 976–980.
- [35] F. Li, B. Zhang, and B. Liu, “Ternary weight networks,” *arXiv preprint arXiv:1605.04711*, 2016.
- [36] C. Zhu, S. Han, H. Mao, and W. J. Dally, “Trained ternary quantization,” *International Conference on Learning Representations (ICLR)*, 2017.
- [37] S. Han, H. Mao, and W. J. Dally, “Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding,” *International Conference on Learning Representations (ICLR)*, 2016.
- [38] Y. Boo and W. Sung, “Structured sparse ternary weight coding of deep neural networks for efficient hardware implementations,” in *Signal Processing Systems (SiPS), 2017 IEEE International Workshop on*. IEEE, 2017.

- [39] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” in *Advances in Neural Information Processing Systems (NIPS)*, 2012, pp. 1097–1105.
- [40] S. Han, J. Pool, J. Tran, and W. Dally, “Learning both weights and connections for efficient neural network,” in *Advances in neural information processing systems*, 2015, pp. 1135–1143.
- [41] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer, “Squeezenet: Alexnet-level accuracy with 50x fewer parameters and; 0.5 mb model size,” *arXiv preprint arXiv:1602.07360*, 2016.
- [42] M. Kim and P. Smaragdis, “Bitwise neural networks,” *arXiv preprint arXiv:1601.06071*, 2016.
- [43] C. Sakr, Y. Kim, and N. Shanbhag, “Analytical guarantees on numerical precision of deep neural networks,” in *Proceedings of the 34th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, D. Precup and Y. W. Teh, Eds., vol. 70. International Convention Centre, Sydney, Australia: PMLR, 06–11 Aug 2017, pp. 3007–3016.
- [44] C. Louizos, K. Ullrich, and M. Welling, “Bayesian compression for deep learning,” in *Advances in Neural Information Processing Systems*, 2017, pp. 3290–3300.
- [45] A. Mishra, E. Nurvitadhi, J. J. Cook, and D. Marr, “WRPN: wide reduced-precision networks,” *arXiv preprint arXiv:1709.01134*, 2017.
- [46] T. M. Cover and J. A. Thomas, *Elements of information theory*. John Wiley & Sons, 2012.

- [47] E. Brochu, V. M. Cora, and N. De Freitas, “A tutorial on bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning,” *arXiv preprint arXiv:1012.2599*, 2010.
- [48] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg *et al.*, “Scikit-learn: Machine learning in python,” *Journal of machine learning research*, vol. 12, no. Oct, pp. 2825–2830, 2011.
- [49] A. Zhou, A. Yao, Y. Guo, L. Xu, and Y. Chen, “Incremental network quantization: Towards lossless cnns with low-precision weights,” *Proceedings of the International Conference on Learning Representations (ICLR)*, 2017.
- [50] J. He, M. Lan, C.-L. Tan, S.-Y. Sung, and H.-B. Low, “Initialization of cluster refinement algorithms: A review and comparative study,” in *Neural Networks, 2004. Proceedings. 2004 IEEE International Joint Conference on*, vol. 1. IEEE, 2004, pp. 297–302.
- [51] D. Bollé, P. Dupont, and J. Van Mourik, “The optimal storage capacity for a neural network with multi-state neurons,” *EPL (Europhysics Letters)*, vol. 15, no. 8, p. 893, 1991.
- [52] A. Krizhevsky and G. Hinton, “Learning multiple layers of features from tiny images,” Citeseer, Tech. Rep., 2009.
- [53] C.-Y. Lee, S. Xie, P. Gallagher, Z. Zhang, and Z. Tu, “Deeply-supervised nets,” in *Artificial Intelligence and Statistics*, 2015, pp. 562–570.
- [54] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” *International Conference on Learning Representations (ICLR)*, 2015.

- [55] S. Zagoruyko and N. Komodakis, “Wide residual networks,” *arXiv preprint arXiv:1605.07146*, 2016.
- [56] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” *arXiv preprint arXiv:1412.6980*, 2014.
- [57] S. Ioffe and C. Szegedy, “Batch normalization: Accelerating deep network training by reducing internal covariate shift,” *arXiv preprint arXiv:1502.03167*, 2015.
- [58] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, “Dropout: a simple way to prevent neural networks from overfitting,” *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.
- [59] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le, “Learning transferable architectures for scalable image recognition,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 8697–8710.
- [60] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “Bert: Pre-training of deep bidirectional transformers for language understanding,” *arXiv preprint arXiv:1810.04805*, 2018.
- [61] L. Fengfu, Z. Bo, and L. Bin, “Ternary weight networks,” in *NIPS Workshop on EMDNN*, vol. 118, 2016, p. 119.
- [62] L. Hou and J. T. Kwok, “Loss-aware weight quantization of deep networks,” *International Conference on Learning Representations (ICLR)*, 2018.
- [63] C. Sakr and N. Shanbhag, “Minimum precision requirements for deep learning with biomedical datasets,” in *2018 IEEE Biomedical Circuits and Systems Conference (BioCAS)*. IEEE, 2018, pp. 1–4.

- [64] S. Gupta, A. Agrawal, K. Gopalakrishnan, and P. Narayanan, “Deep learning with limited numerical precision,” in *International Conference on Machine Learning (ICML)*, 2015, pp. 1737–1746.
- [65] R. Andri, L. Cavigelli, D. Rossi, and L. Benini, “Yodann: An ultra-low power convolutional neural network accelerator based on binary weights,” in *2016 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*. IEEE, 2016, pp. 236–241.
- [66] K. He, X. Zhang, S. Ren, and J. Sun, “Identity mappings in deep residual networks,” in *European conference on computer vision*. Springer, 2016, pp. 630–645.
- [67] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, “Intriguing properties of neural networks,” *arXiv preprint arXiv:1312.6199*, 2013.
- [68] A. Kurakin, I. Goodfellow, S. Bengio, Y. Dong, F. Liao, M. Liang, T. Pang, J. Zhu, X. Hu, C. Xie *et al.*, “Adversarial attacks and defences competition,” in *The NIPS’17 Competition: Building Intelligent Systems*. Springer, 2018, pp. 195–231.
- [69] W. Xu, D. Evans, and Y. Qi, “Feature squeezing: Detecting adversarial examples in deep neural networks,” *arXiv preprint arXiv:1704.01155*, 2017.
- [70] A. S. Rakin, J. Yi, B. Gong, and D. Fan, “Defend deep neural networks against adversarial examples via fixed and dynamic quantized activation functions,” *arXiv preprint arXiv:1807.06714*, 2018.
- [71] A. Galloway, G. W. Taylor, and M. Moussa, “Attacking binarized neural networks,” in *International Conference on Learning Representations (ICLR)*, 2018.

- [72] F. Liao, M. Liang, Y. Dong, T. Pang, X. Hu, and J. Zhu, “Defense against adversarial attacks using high-level representation guided denoiser,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 1778–1787.
- [73] Y. Wu, M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey *et al.*, “Google’s neural machine translation system: Bridging the gap between human and machine translation,” *arXiv preprint arXiv:1609.08144*, 2016.
- [74] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, “Attention is all you need,” in *Advances in neural information processing systems*, 2017, pp. 5998–6008.
- [75] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, “MobileNetV2: Inverted residuals and linear bottlenecks,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 4510–4520.
- [76] L. N. Smith, “Cyclical learning rates for training neural networks,” in *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE, 2017, pp. 464–472.
- [77] S. Jastrzebski, Z. Kenton, D. Arpit, N. Ballas, A. Fischer, Y. Bengio, and A. Storkey, “Three factors influencing minima in SGD,” *arXiv preprint arXiv:1711.04623*, 2017.
- [78] M. Cisse, P. Bojanowski, E. Grave, Y. Dauphin, and N. Usunier, “Parseval networks: Improving robustness to adversarial examples,” in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, 2017, pp. 854–863.

- [79] M. Everingham, S. A. Eslami, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman, “The pascal visual object classes challenge: A retrospective,” *International journal of computer vision*, vol. 111, no. 1, pp. 98–136, 2015.
- [80] L.-C. Chen, G. Papandreou, F. Schroff, and H. Adam, “Rethinking atrous convolution for semantic image segmentation,” *arXiv preprint arXiv:1706.05587*, 2017.
- [81] B. Hariharan, P. Arbeláez, L. Bourdev, S. Maji, and J. Malik, “Semantic contours from inverse detectors,” in *2011 International Conference on Computer Vision*. IEEE, 2011, pp. 991–998.
- [82] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, “Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 40, no. 4, pp. 834–848, 2017.
- [83] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein *et al.*, “Imagenet large scale visual recognition challenge,” *International Journal of Computer Vision*, vol. 115, no. 3, pp. 211–252, 2015.
- [84] D. Zhang, J. Yang, D. Ye, and G. Hua, “LQ-Nets: Learned quantization for highly accurate and compact deep neural networks,” in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 365–382.
- [85] R. Gong, X. Liu, S. Jiang, T. Li, P. Hu, J. Lin, F. Yu, and J. Yan, “Differentiable soft quantization: Bridging full-precision and low-bit neural networks,” in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 4852–4861.

- [86] C. Bucilua, R. Caruana, and A. Niculescu-Mizil, “Model compression,” in *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2006, pp. 535–541.
- [87] A. Mishra and D. Marr, “Apprentice: Using knowledge distillation techniques to improve low-precision network accuracy,” in *International Conference on Learning Representations*, 2018. [Online]. Available: <https://openreview.net/forum?id=B1ae1lZRb>
- [88] S. Shin, Y. Boo, and W. Sung, “Empirical analysis of knowledge distillation technique for optimization of quantized deep neural networks,” *arXiv preprint arXiv:1909.01688*, 2019.
- [89] J. Kim, Y. Bhalgat, J. Lee, C. Patel, and N. Kwak, “Qkd: Quantization-aware knowledge distillation,” *arXiv preprint arXiv:1911.12491*, 2019.
- [90] Y. Boo, S. Shin, and W. Sung, “Quantized neural networks: Characterization and holistic optimization,” *arXiv preprint arXiv:2006.00530*, 2020.
- [91] S.-C. Zhou, Y.-Z. Wang, H. Wen, Q.-Y. He, and Y.-H. Zou, “Balanced quantization: An effective and efficient approach to quantized neural networks,” *Journal of Computer Science and Technology*, vol. 32, no. 4, pp. 667–682, 2017.
- [92] Z. Cai, X. He, J. Sun, and N. Vasconcelos, “Deep learning with low precision by half-wave gaussian quantization,” in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2017, pp. 5406–5414.
- [93] J. Ott, Z. Lin, Y. Zhang, S.-C. Liu, and Y. Bengio, “Recurrent neural networks with limited numerical precision,” *arXiv preprint arXiv:1608.06902*, 2016.
- [94] S. Kapur, A. Mishra, and D. Marr, “Low precision RNNs: Quantizing RNNs without losing accuracy,” *arXiv preprint arXiv:1710.07706*, 2017.

- [95] Y. Chebotar and A. Waters, “Distilling knowledge from ensembles of neural networks for speech recognition.” in *Interspeech*, 2016, pp. 3439–3443.
- [96] G. Chen, W. Choi, X. Yu, T. Han, and M. Chandraker, “Learning efficient object detection models with knowledge distillation,” in *Advances in Neural Information Processing Systems*, 2017, pp. 742–751.
- [97] A. v. d. Oord, Y. Li, I. Babuschkin, K. Simonyan, O. Vinyals, K. Kavukcuoglu, G. v. d. Driessche, E. Lockhart, L. C. Cobo, F. Stimberg *et al.*, “Parallel wavenet: Fast high-fidelity speech synthesis,” *arXiv preprint arXiv:1711.10433*, 2017.
- [98] D. Dai and L. Van Gool, “Dark model adaptation: Semantic image segmentation from daytime to nighttime,” in *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2018, pp. 3819–3824.
- [99] A. Romero, N. Ballas, S. E. Kahou, A. Chassang, C. Gatta, and Y. Bengio, “Fitnets: Hints for thin deep nets,” *arXiv preprint arXiv:1412.6550*, 2014.
- [100] M. Kulkarni, K. Patil, and S. Karande, “Knowledge distillation using unlabeled mismatched images,” *arXiv preprint arXiv:1703.07131*, 2017.
- [101] W. Park, D. Kim, Y. Lu, and M. Cho, “Relational knowledge distillation,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 3967–3976.
- [102] J. Yim, D. Joo, J. Bae, and J. Kim, “A gift from knowledge distillation: Fast optimization, network minimization and transfer learning,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 4133–4141.
- [103] A. Koratana, D. Kang, P. Bailis, and M. Zaharia, “Lit: Learned intermediate representation training for model compression,” in *International Conference on Machine Learning*, 2019, pp. 3509–3518.

- [104] T. Furlanello, Z. C. Lipton, M. Tschannen, L. Itti, and A. Anandkumar, “Born again neural networks,” *arXiv preprint arXiv:1805.04770*, 2018.
- [105] Y. Zhang, T. Xiang, T. M. Hospedales, and H. Lu, “Deep mutual learning,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 4320–4328.
- [106] X. Zhu, S. Gong *et al.*, “Knowledge distillation by on-the-fly native ensemble,” in *Advances in neural information processing systems*, 2018, pp. 7517–7527.
- [107] B. Zhuang, C. Shen, M. Tan, L. Liu, and I. Reid, “Towards effective low-bitwidth convolutional neural networks,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 7920–7928.
- [108] Y. Bengio, N. Léonard, and A. Courville, “Estimating or propagating gradients through stochastic neurons for conditional computation,” *arXiv preprint arXiv:1308.3432*, 2013.
- [109] J. Tang, R. Shivanna, Z. Zhao, D. Lin, A. Singh, E. H. Chi, and S. Jain, “Understanding and improving knowledge distillation,” *arXiv preprint arXiv:2002.03532*, 2020.
- [110] D. Chen, J.-P. Mei, C. Wang, Y. Feng, and C. Chen, “Online knowledge distillation with diverse peers,” in *Thirty-Fourth AAAI Conference on Artificial Intelligence*, 2020.
- [111] A. Krizhevsky and G. Hinton, “Convolutional deep belief networks on cifar-10,” *Unpublished manuscript*, vol. 40, no. 7, pp. 1–9, 2010.
- [112] E. Park, D. Kim, S. Yoo, and P. Vajda, “Precision highway for ultra low-precision quantization,” *arXiv preprint arXiv:1812.09818*, 2018.

- [113] J. Choi, S. Venkataramani, V. Srinivasan, K. Gopalakrishnan, Z. Wang, and P. Chuang, “Accurate and efficient 2-bit quantized neural networks,” in *Proceedings of the 2nd SysML Conference*, 2019.
- [114] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio, “Quantized neural networks: Training neural networks with low precision weights and activations.” *The Journal of Machine Learning Research*, vol. 18, no. 187, pp. 1–30, 2017.
- [115] NVIDIA, *NVIDIA TITAN X Pascal GPU*. [Online]. Available: <https://www.nvidia.com/en-us/titan/titan-xp/?cjevent=c847bd62a63511ea816700880a180511>
- [116] G. Research, *BERT-Base model url*. [Online]. Available: <https://github.com/google-research/bert>
- [117] P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang, “Squad: 100,000+ questions for machine comprehension of text,” *arXiv preprint arXiv:1606.05250*, 2016.
- [118] Y. Boo and W. Sung, “Fixed-point optimization of transformer neural network,” in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 1753–1757.
- [119] M.-E. Nilsback and A. Zisserman, “Automated flower classification over a large number of classes,” in *2008 Sixth Indian Conference on Computer Vision, Graphics & Image Processing*. IEEE, 2008, pp. 722–729.

초 록

최근 깊은 신경망(deep neural network, DNN)은 다양한 분야에서 매우 인상적인 성능을 보이고 있다. 그러나, 신경망의 복잡도가 함께 증가하면서, 점점 더 많은 계산 및 메모리 접근 비용이 발생하고 있다. 인공신경망의 양자화(quantization)는 깊은 신경망의 동작 비용을 줄일 수 있는 효과적인 방법 중 하나이다. 일반적으로, 신경망의 가중치(weights) 및 활성화된 신호(activation outputs)는 32 비트 부동 소수점(floating-point) 정밀도를 가진다. 고정 소수점 양자화는 이를 더 낮은 정밀도로 표현함으로써 신경망의 크기 및 연산 비용을 줄인다. 그러나, 1또는 2비트 등 매우 낮은 정밀로도 양자화된 신경망은 부동 소수점 신경망과 비교하여 큰 성능 하락을 보인다. 기존의 연구들은 양자화 에러(error)에 대한 분석 없이 주어진 데이터와 모델에 대한 최적화 방법을 제시한다. 이러한 연구 결과를 다른 모델과 데이터에 적용하기 위해서는 수많은 시뮬레이션을 수행하여 성능을 유지할 수 있는 양자화 정밀도의 한계를 찾아야 한다.

본 연구에서는 신경망에서의 양자화 특성을 분석하고, 양자화로 인한 신경망의 성능 저하 원인을 제시한다. 신경망의 양자화는 크게 가중치 양자화(weight quantization)와 활성화 함수 양자화(activation quantization)로 나뉜다. 먼저, 가중치 양자화의 특성을 분석하기 위해 무작위 훈련 샘플을 생성하고, 이 데이터로 신경망을 훈련시키면서 신경망의 암기 능력(memorization capacity)을 정량화 한다. 신경망이 자신의 암기 능력을 최대한 활용하도록 훈련시킨 뒤 성능이 하락하는 양자화 정밀도의 한계를 분석한다. 분석 결과, 가중치가 정보량을 잃기 시작하는 양자화 정밀도는 파라미터의 수와 관계가 없음을 확인하였다. 뿐만 아니라, 파라미터에 저장된 정보를

유지할 수 있는 한계 양자화 정밀도는 모델의 구조에 따라 달라진다.

또한, 본 연구에서는 활성화 함수 양자화와 가중치 양자화로 인한 에러의 차이점을 분석한다. 합성 데이터(synthesized data)를 생성하고, 이 데이터로 훈련된 모델을 양자화 한 뒤 양자화 에러를 시각화 한다. 분석 결과 가중치 양자화는 신경망의 용량(capacity)을 감소시키며, 신경망의 파라미터 수를 증가시키면 가중치 양자화 에러가 감소한다. 반면, 활성화 함수의 양자화는 추론 과정(inference)에서 잡음(noise)을 유발하며 신경망의 깊이가 깊어질 수록 활성화 함수의 에러가 증폭된다. 본 연구에서는, 두 양자화 에러의 차이를 바탕으로 양자화 친화적 아키텍처 설계와 고정 소수점 훈련 방법을 포함하는 포괄적인 고정 소수점 최적화 방법을 제안한다.

뿐만 아니라, 활성화 함수가 양자화된 신경망의 성능 복원력을 높이는 방법으로 SPEQ 훈련 방법을 제안한다. 제안하는 훈련 방법은 지식 증류(knowledge distillation, KD) 기반 학습 방법으로, 매 훈련 단계마다 서로 다른 선생 모델의 정보를 활용한다. 선생 모델의 파라미터는 학생 모델과 동일하며, 활성화 함수의 양자화 정밀도를 확률적으로 선택함으로써 선생 모델의 소프트 라벨(soft label)을 생성한다. 따라서 선생 모델은 학생 모델에서 유발되는 양자화 잡음을 고려한 지식을 제공해 준다. 학생 모델은 훈련 단계마다 다른 종류의 양자화 잡음을 고려한 지식으로 훈련되기 때문에 앙상블 학습(ensemble training) 효과를 얻을 수 있다. 제안하는 SPEQ 훈련 방법은 다양한 분야에서 양자화된 신경망의 성능을 크게 향상시켰다.

주요어: 양자화된 깊은 신경망, 암기 캐패시티, 양자화 에러 시각화,
고정 소수점 최적화, SPEQ 훈련법

학번: 2016-20915

감사의 글

약 4년 반의 시간동안 수 많은 분들의 도움으로 무사히 박사학위 과정을 마칠 수 있었습니다. 먼저, 많이 부족한 저를 믿어주시고 연구자의 자세가 무엇인지, 어떻게 올바른 연구를 할 수 있는 지를 알려주신 성원용 교수님께 깊은 감사의 말씀을 드립니다. 항상 마음 속에 교수님의 말씀을 새기면서 살아가도록 하겠습니다. 또한, 학위 논문 심사에 귀중한 시간을 할애해 주신 이정우 교수님, 윤성로 교수님, 유승주 교수님, 한양대학교 최정욱 교수님께 깊은 감사의 말씀을 올립니다. 교수님들께서 해주신 훌륭한 조언들이 졸업 논문을 완성하는 데 큰 도움이 되었습니다.

연구실에 입학하여 아무것도 모르는 저에게 많은 조언을 해주셨던 준희형, 규연이형, 사지드, 동윤이형, 민재형, 성욱이형에게도 감사를 드립니다. 함께 같은 주제로 연구를 진행하며, 저의 논문 작성에 가장 많은 도움을 주었던 성호형, 학부 동기이자 연구실 선배 규홍이, 연구 분야는 다르지만 연구실에서 가장 오랜시간을 함께 지낸 진환이, 저보다 늦게 연구실에 들어왔지만 먼저 떠난 윤진이, 루카스형, 영민이와 함께 졸업을 하는 천설이, 그리고 앞으로의 미래가 기대되는 석현이, 마지막으로 항상 뒤에서 연구실과 관련된 많은 업무를 도와주셨던 미순씨 모두 많이 감사드리고, 항상 가시는 길 좋은 일들만 가득하기를 바라겠습니다.

마지막으로, 오랜 기간동안 연구실 생활을 하면서 걱정만 시켜드리고 자주 찾아뵙지도 못한 아버지 어머니께 항상 죄송하고도 감사한 마음을 전하고 싶습니다. 항상 걱정해 주시는 할머니께도 자랑스러운 손주가 될 수 있어 많이 기쁘고 감사합니다. 박사과정을 하는 동안 많은 응원을 해 주신 삼촌과 고모에게도 감사드립니다. 가족분들의 응원과 격려가 있었기에 박사학위 논문을 완성할 수 있었습니다.

2020년 7월 23일

부윤희