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

Ahmed T. Elthakeb [1] **Prannoy Pilligundla** [2] **Fatemehsadat Mireshghallah** [2]
Alexander Cloninger [3] **Hadi Esmaeilzadeh** [2]

Abstract

The deep layers of modern neural networks extract a rather rich set of features as an input propagates through the network, this paper sets out to harvest these rich intermediate representations for quantization with minimal accuracy loss while significantly reducing the memory footprint and compute intensity of the DNN. This paper utilizes knowledge distillation through teacher-student paradigm (Hinton et al., 2015) in a novel setting that exploits the feature extraction capability of DNNs for higher-accuracy quantization. As such, our algorithm logically divides a pretrained full-precision DNN to multiple sections, each of which exposes intermediate features to train a team of students independently in the quantized domain. This divide and conquer strategy, makes the training of each student section possible in isolation, which offers additional speedup through enabling parallelization, while all these independently trained sections are later stitched together to form the equivalent fully quantized network.

Experiments on various DNNs (AlexNet, LeNet, MobileNet, ResNet-18, ResNet-20, SVHN and VGG-11) show that, this approach—called DCQ (**D**ivide and **C**onquer **Q**uantization)—on average, improves the performance of a state-of-the-art quantized training technique, DoReFa-Net (Zhou et al., 2016) by 21.6% and 9.3% for binary and ternary quantization, respectively. Additionally, we show that incorporating DCQ to existing quantized training methods leads to improved accuracies as compared to previously reported by multiple state-of-the-art quantized training methods.

[1]Department of Electrical and Computer Engineering, University of California San Diego. [2]Department of Computer Science, University of California San Diego. [3]Department of Mathematics, University of California San Diego. Correspondence to: Ahmed T. Elthakeb <elthakeb@ucsd.com>.

Proceedings of the 37th International Conference on Machine Learning, Online, PMLR 119, 2020.

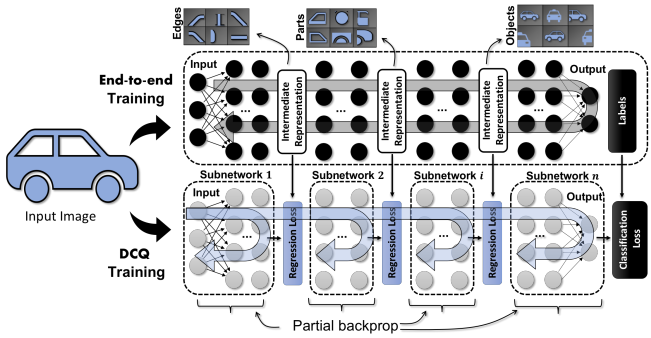

Figure 1: Overview of Divide and Conquer Quantization.

1. Introduction

Today deep learning, with its superior performance, dominates a wide range of real life inference tasks including image recognition, voice assistants, and natural language processing (Hauswald et al., 2015; Krizhevsky et al., 2012; LeCun et al., 2015; 1989). However, the shear complexity of deep learning models and the associated heavy compute and memory requirement appears as a major challenge as the demand for such services rapidly scale. Quantization, which can reduce the complexity of each operation as well as the overall storage requirements of the DNN, has proven to be a promising path forward. Nevertheless, quantization requires carefully tailored training and recovery algorithms (Courbariaux et al., 2015; Gupta et al., 2015; Hubara et al., 2017a; Zhou et al., 2017; 2016) to even partially overcome its losses in accuracy. In this paper, we set out to devise an algorithm that enables quantization with much less accuracy degradation. The key insight is that the intermediate layers of a deep network already extract a very rich set of features and these intermediate representations can be used to train/teach a quantized network more effectively. To that end, we define a new approach towards knowledge distillation through teacher-student paradigm (Hinton et al., 2015; Bucila et al., 2006) focusing on teaching the knowledge of intermediate features to a corresponding quantized student. Knowledge distillation (Hinton et al., 2015) is a generic approach to reduce a large model down to a simpler or smaller distilled model. At a high level, a softened version of the final output is used to train a small model (student) to mimic the behavior of the original large model

(teacher). FITNETS (Romero et al., 2015) extends this idea and takes hints from an intermediate layer of the teacher to pretrain the first few layers of the student and then apply knowledge distillation to the entire student network. We, on the other hand, tap into the multiple intermediate layers and apply knowledge distillation through sectioning. The sectioning enables DCQ to train each section of the students independently in isolation to deliver a quantized counterpart for the teacher, which enables parallelization. As such, DCQ offers additional speedup through parallelization on the algorithmic level and independent of improvements in hardware accelerators. In fact, the hints as proposed in FITNETS are complementary and can potentially be used in our sectional knowledge distillation. The proposed algorithm, DCQ, employs a divide and conquer approach that divides a pretrained full-precision network into multiple sections, each of which exposes a set of intermediate features. As Figure 1 illustrates, DCQ allocates a student section to each teacher counterpart and independently trains them using the intermediate feature representations. DCQ calculates the loss of each student section by comparing it with the output activations of the corresponding teacher section of the full precision network. Loss is optimized through a sectional multi-backpropagation scheme using conventional gradient-based training as shown in Figure 1. These trained student sections are then sewed back together to form the corresponding quantized DNN.

We validate our method through experiments on a variety of DNNs including AlexNet, LeNet, MobileNet, ResNet-18, ResNet-20, SVHN and VGG-11 with binary and ternary weights. Results show that DCQ, on average, improves the performance of a state-of-the-art quantized training technique, DoReFa-Net (Zhou et al., 2016) by 21.6% and 9.3% for binary and ternary quantization, respectively, which further helps in closing the accuracy gap between state-of-the-art quantized training techniques and the full-precision runs. Additionally, we show that our approach, DCQ, can improve performance of existing knowledge-distillation based approaches (Mishra et al., 2018) and multiple state-of-the-art quantized training methods. These encouraging results suggest that leveraging the inherent feature extraction ability of DNNs for knowledge distillation can lead to significant improvement in their efficiency, reducing their bitwidth in this particular case.

The contributions of this paper can be summarized as follows.

- **Extending knowledge distillation.** DCQ enables leveraging *arbitrary number of intermediate layers* relying on the inherent hierarchical learning characteristic of deep neural networks in contrast to only the output layer or hint layer. As such, distillation learning and hint learning fall as special cases of the proposed *divide and conquer strategy*.
- **Enabling parallelization towards training quantized networks.** DCQ applies knowledge distillation through sectioning. As such it trains each section of the students independently in isolation to deliver a quantized counterpart for the teacher, which can occur in parallel.
- **Complementary to other methods.** DCQ is a complementary method as it acts as an auxiliary approach to boost performance of existing training techniques by applying whatever the underlying training technique but in a stage-wise fashion with defining a regression loss per stage.
- **Theoretical analysis.** We provide a theoretical analysis/guarantee of the error upper bound across the network through a chaining argument.

2. DCQ: Divide and Conquer for Quantization

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

Weight and activation quantization. The proposed technique is orthogonal to the quantity of interest for quantization, as it's basically applying whatever the underlying/used training technique but in a stage-wise fashion with defining a new regression loss per stage. In fact, the regression loss is defined to match the respective activation maps for each stage. As such, DCQ can be equally applied for weight and/or activation quantization alike. Section 3.2 presents results for both weight and activation quantization.

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

Knowledge distillation utilization. DCQ extends the concept of knowledge distillation to its limits by leveraging multiple intermediate layers as opposed to limiting it to the output layer only as in (Mishra & Marr, 2018), (Hinton et al., 2015) or the output layer and hint layer as in (Romero et al., 2015).

Other performance benefits. DCQ enables per-network training "parallelization" by enabling training different sections/stages in isolation (stage-wise fashion). Moreover,

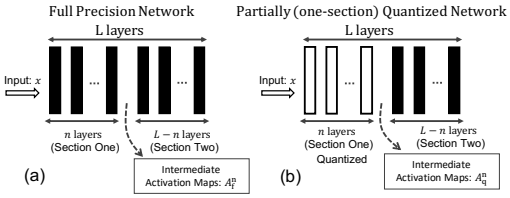

Figure 2: DCQ two stage split example

it applies the standard back propagation in a simpler settings (small subnetworks) which enables both faster convergence time and higher accuracy than existing conventional fine-tuning methods in the quantized domain.

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

2.1. Matching Activations for Intermediate Layers

Figure 2 (a) shows a sketch representing a full precision network of L layers, whereas Figure 2 (b) is a deeply quantized version of the same network where first n layers are quantized and the remaining $L-n$ layers are at full precision. When we pass the same input image x to both these networks, if the output activations of layer n for full precision network, i.e., A_f^n, are equivalent to the output activations of layer n for the semi-quantized network, A_q^n, then both the networks classify the input to a same class because rest of the $L-n$ layers are same for both the networks and their input activations are same as well. Therefore, if both these networks shown in Figure 2 (a) and (b), have similar output activations for all the input images, then the network with first n layers quantized has learnt to represent similar features as the first n layers of the original network and it will have the same classification accuracy as the full precision network. We can extend this argument further and say that if we quantize the remaining $L-n$ layers of the network in Figure 2 (b) while keeping it's output same as the corresponding $L-n$ layers of the full precision network, then we now have a deeply quantized network with the same accuracy as the full precision network. This is the underlying principle for our proposed quantization technique DCQ. In the above example, the network was split into two sections of n and $L-n$ layers, instead DCQ splits the original network into multiple sections and trains those sections individually to output same activations as the corresponding section in the full precision network. Following subsections explain the DCQ methodology in more detail.

2.2. Splitting, Training and Merging

Splitting the full precision network. As described in Section 2.1, DCQ splits the original network into multiple sections and trains them in isolation and in parallel. Figure 3 shows an overview of the entire process. As shown in the figure, after splitting the full precision network into m sub sections, DCQ quantizes and trains these subsections independently. After training, DCQ puts them all together again to get the deeply

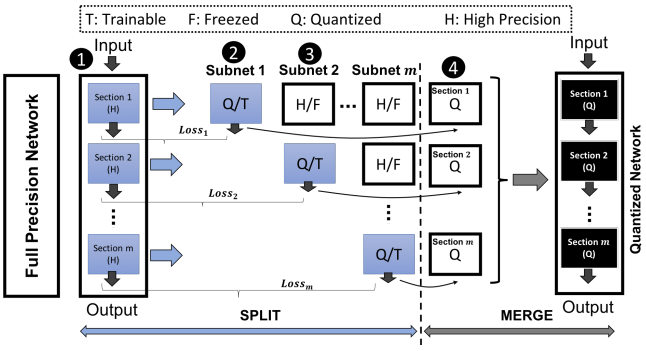

Figure 3: Divide and Conquer approach overview showing SPLIT phase; dividing the teacher full precision network into smaller subnetworks, and MERGE; by combining the training results of each subnetwork to form a fully quantized network

quantized version of the entire original network. As discussed in Section 2.1, because each of these sections is trained to capture the same features as the full precision network, although these sections are trained independently, they can be put together at the end to give similar accuracy as the full precision network.

If the original network has L layers then m decides how many layers will be part of each section (sections need not be equal in terms of number of layers). In this work, we used a configuration of two layers per every section and then decided m according to the total number of layers in the network. Although, for networks like ResNet which have logical splits in terms of basic blocks, we split the network in a way that each section corresponds to a basic block. We leave the task of deciding the optimal number of sections (splits) and how many layers per section for a given network to future work. However, we provide some empirical analysis to this regard in Section 3.5.

Training the sub-networks. As Figure 3 illustrates, ❶ we create m sections in order to train each of the m sub-networks. For each section i, the sub-network i (or subnet i for short) consists of all the sections preceding it. Subnet 1 column in Figure 3 ❷ shows a subnet for section 1. To train this section, the output activations of the quantized version of section 1 are compared with the output activations of the full precision version of section 1 and the loss is calculated accordingly. Section 2.3 gives more details on how the loss is calculated for each subnet. Similarly, Subnet 2 column ❸ shows the subnet for section 2 and it comprises of both section 1 and section 2. Output activations of section 2 are used to calculate the loss in this case. Since section 2 is being trained in this subnet, weights for section 1 are frozen(not trainable) in this subnet and backpropagation based on the loss only affects section 2. Similarly there are subnets for sections 3 up to last section m and the last subnet m is basically similar to the full precision network except that the section m is quantized and all the other sections from 1 to $m-1$ are frozen.

Merging the sections. ❹ After training all the sections, since each of these sections has been trained independently to learn

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

2.3. Loss Function for Training Sub Networks

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

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

where y is the target value, and $\hat{y}$ is the predicted value, and the summation is across all samples. For Huber loss, δ (delta) is a hyperparameter which can be tuned. Huber loss approaches MAE when $\delta \sim 0$ and MSE when $\delta \sim \infty$ (large numbers). Section 3.5 provides experimental results for each of the above loss formulations.

2.4. Overall Algorithm

Algorithm 1 outlines the step by step procedure for DCQ putting together all the steps described in Sections 2.2 and 2.3. Since each iteration of the loop, shown in the algorithm, is independent, all the sections can potentially be trained in parallel leading to an overall reduction in training time.

3. Experimental Results

3.1. Experimental Setup

In this section, we evaluate the efficacy of our proposed approach on various DNNs (AlexNet, LeNet, MobileNet, ResNet-18, ResNet-20, SVHN and VGG-11) and different datasets: CIFAR10, ImageNet, MNIST, and SVHN. We compare our approach to conventional end-to-end training approach. We consider DoReFa-Net (Zhou et al., 2016) as our baseline but also show comparision with BWN (Rastegari et al., 2016b) in Section 3.2, and Apprentice (Mishra & Marr, 2018) , in addition to state-of-the-art quantized training methods: PACT (Choi et al., 2018), LQ-Net (Zhang et al., 2018), DSQ (Gong et al., 2019) in Section 3.3.

Algorithm 1 Divide and Conquer for Quantization: Training Procedure

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

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

Table 1: Summary of results comparing DCQ (our appraoch) to DoReFa-Net for different networks considering binary and ternary weight quantization.

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

Table 2: Summary of results comparing our approach (DCQ) to state-of-the-art quantized training methods.

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

Table 3: Comparing DCQ to a knowledge distillation based quantization method, Apprentice.

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

For all the experiments, we use an open source framework for quantization, Distiller (Zmora et al., 2018). While reporting accuracies in their paper, DoReFa-Net doesn't quantize first and last layers of the network whereas in our case, we quantize all the layers including the first and last layers. Because of this difference in quantization and using built-in implementation of Distiller, the accuracies we report might not exactly match the accuracies reported in their paper.

3.2. Binarization and Ternarization using DCQ

Table1 shows summary of results comparing plain DoReFa to DoReFa + DCQ for different networks considering binary {-1,1} and ternary{-1, 0, 1} weight quantization for various networks: LeNet, ResNet-20, SVHN and VGG-11. As seen, integrating DCQ into DoReFa outperforms the conventional approach and achieves a consistent improvements across the different networks with average 22.45% for binarization and 9.7% for ternarization.

Delving into the results, the reported improvements can be attributed to the following reasons. First, deep multi-hidden-layer neural networks are much more difficult to tackle as compared to shallower ones. Furthermore, end-to-end backpropagation can be inefficient (Jaderberg et al., 2017). Thus, adopting such divide and conquer approach yields simpler subproblems that are easier to optimize. Second, matching intermediate learning objectives also guides the optimization as compared to following a single global objective that indirectly specifies learning objectives to the intermediate layers.

Comparison with BWN. BWN (Rastegari et al., 2016b) proposes approximate convolutions using binary operations for a set of networks. We show comparison on LeNet as it is the only common benchmark between both the works. As Table 1 shows, our technique achieves an accuracy of 99.3%, which is close to the accuracy of 99.2% reported by BWN. However, BWN involves restructuring the original network architecture whereas our implementation does not introduce any changes to the architecture.

3.3. Comparison with Quantized Training Methods

Here, we provide comparison to multiple state-of-the-art quantized training methods considering both weights and activation quantization. Table 2 summarizes the results of comparing to PACT, LQ-Net, DSQ, and DoReFa (the baseline) for several networks (AlexNet, ResNet-18, MobileNet). As seen, DCQ outperforms these previously reported accuracies and achieves on average improvements of 0.98%, and 0.96% for W4/A4 and W3/A3, respectively.

We also provide a comparison against knowledge distillation-based method Apprentice (Mishra & Marr, 2018), a recent work which also combines knowledge distillation with quantization. Table 3 shows that our technique outperforms Apprentice for both ResNet-20 on CIFAR10, and ResNet-18 on ImageNet considering ternary weights quantization. The reported improvement can be attributed to the fact that DCQ combines the conventional knowledge distillation approach, as in (Mishra & Marr, 2018), in addition to its unique intermediate learning approach by regressing the quantized network intermediate feature maps to the corresponding full precision ones in a stage wise fashion. Moreover, the network architecture of the student network in (Mishra & Marr, 2018) is typically different from that of the teacher network as opposed to DCQ where same network architecture is utilized for the student network but with quantized weights. From one side, this saves a huge amount of effort designing a student network architecture which might incur significant hyperparameter tuning. On the other side, it enables a direct finetuning instead of a complete training from scratch as a result of preserving the original network architecture.

3.4. Analysis: DCQ vs Conventional Binary Kernels

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

Figure 4 shows a visualization of a subset of weight kernels from the second convolutional layer of LeNet and AlexNet. (a) is the original full precision kernels, (b) direct binarization of full precision kernels, and (c) binarization after training (applying DCQ). In the figure, weights that are different between the trained binary kernel and the directly binarized kernel are highlighted with square rectangles across the three visualizations. Spatially contrasting those highlighted altered weights on the full precision kernels, it can be noticed that they mostly share a common feature that is being low in magnitude (shown as white squares in (a)). From statistical point of view, Figure 4 (d) shows the original full precision weights histogram (in blue) and overlaying the portion of the altered weights (in light orange). We can observe the following. First, during training, only very small percentage of the weights are actually altered relative to the total number of weights. Specifically, in this example, it is around 3.5% and 2.25% for LeNet and AlexNet respectively, of the total weights got impacted by

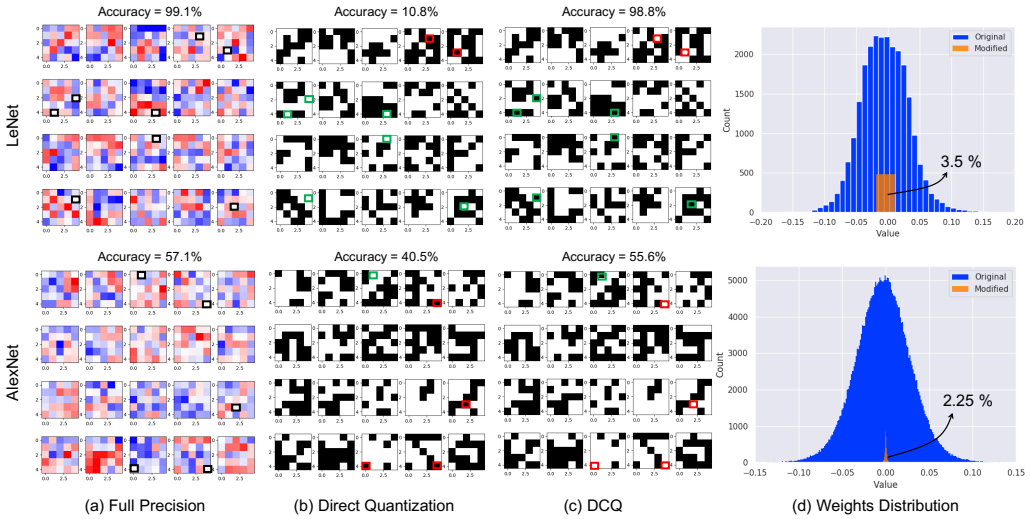

Figure 4: Visualization of a subset of weight kernels of the second convolutional layer of LeNet (top row), and AlexNet (bottom row), highlighting the differences between different versions of binary weight kernels: (a) Full precision weight kernels, (b) binary weight kernels upon direct binarization from full precision, (c) binary weight kernels obtained using our method DCQ, and (d) weights histogram of the convolutional layer highlighting the altered binary weights after training (using DCQ) relative to the original distribution

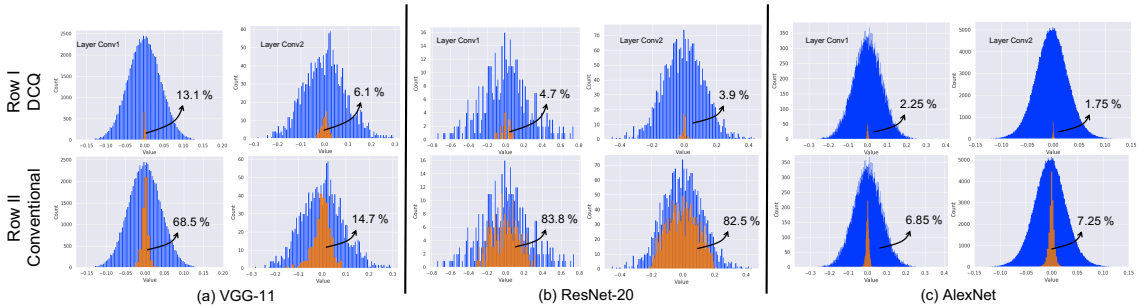

Figure 5: Weights histograms of the first two convolutional layers of three different DNNs: (a) VGG11, (b) ResNet-20, and (c) AlexNet, highlighting the altered portion of the trained binary weights (depicted percentages indicate the exact portion in orange) relative to the directly binarized weights. Original total weights histograms are shown in blue. Row I shows the results using our method (DCQ), and Row II shows for the conventional end-to-end taining method.

training. Moreover, despite the marginal difference between the binary kernels, they experience dramatic accuracy difference: 10.8% vs 98.1% for kernels in (b) and (c) respectively, for LeNet, and 40.5% and 55.6% for AlexNet.

Now, to check whether this is a general trend and whether different training algorithms has an impact on this, we extend our statistical analysis to more networks. Figure 5 shows weight histograms of the first two convolutional layers of AlexNet, ResNet-20, and VGG-11. As seen in the figure, first, for Figure 5 Row I (DCQ), the altered portion of binary weights during training is consistently small in both number and magnitude across different layers and different networks. Second, contrasting that behavior using DCQ vs using the conventional end-to-end quantized training, as shown in Figure 5 Row II (Conventinoal), we see that binary weight kernels clearly encounter much more variations during the conventional end-to-end training as compared to our approach, DCQ.

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

3.5. Exploratory Studies

Impact of different loss formulations for intermediate learning. As mentioned in section 2.3, we have examined three of the most commonly used loss formulations. Namely: (1) Mean Square Error (MSE); (2) Mean Absolute Error (MAE);

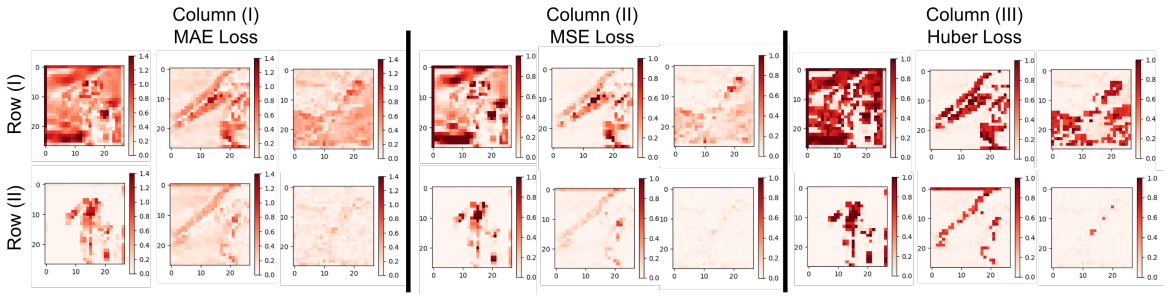

Figure 6: Loss visualization of intermediate feature maps samples. Row(I): before DCQ training, Row(II): after DCQ training. Columns show results for different loss formulations. Col(I) MAE, Col(II) MSE, and Col(III) Huber loss. The results are for the second convolution layer in AlexNet with binary quantization.

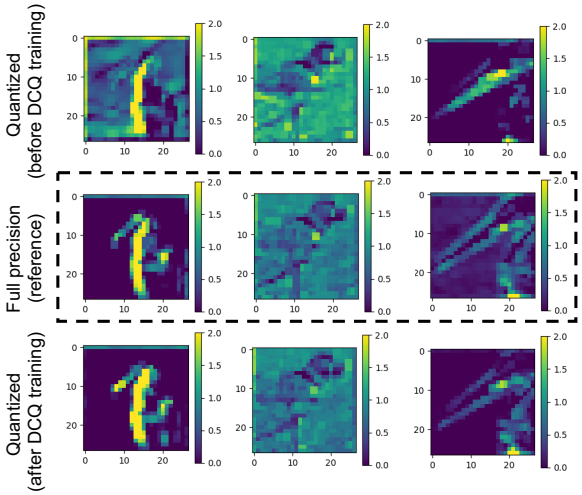

Figure 7: Feature maps before and after DCQ training compared to full precision maps. The results are for the second convolution layer in AlexNet with binary quantization.

(3) Huber Loss. Figure 6 shows different samples of feature maps losses (for the second convolution layer of AlexNet wtih binary weights). Row(I) shows different samples of feature map losses before DCQ training. Row(II) shows the losses for the same samples after DCQ training (matching feature maps). Different columns show different loss formulations. Col(I): MSE Loss; Col(II): MAE Loss; and Col(III): Huber Loss. As it can be seen, the feature map losses (the amount of redness) significantly decreases after DCQ training as a result of regressing the quantized model intermediate feature maps to the full precision counterparts. We can also notice that the behavior is consistent across different regression losses. Nevertheless, based on our experimentation, among the considered formulations, MSE seems to be the most effective during the intermediate learning process. The trends are similar for the other networks. Figure 7 compares visualizations of different samples of actual feature maps before and after DCQ training with respect to the full precision ones demonstrating the effectiveness of the proposed approach. Lastly, divide and conquer is a very basic and universal engineering principle that is commonly and widely applied across

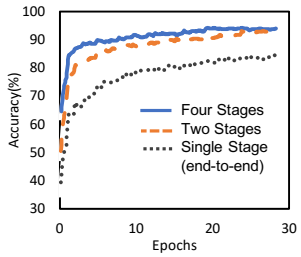

Figure 8: Impact of different splitting on the convergence behavior for VGG-11 (ternary quantization).

a variety of fields. Here, we propose a procedure that extends such effective principle to quantized training of neural networks.

Impact of the number of splitting points. As number of splitting points increases, the large optimization problem gets divided into smaller subproblems. Thus, on one side, it becomes easier to solve each subproblem separately. On the other side, however, the complexity overhead increases as well. We leave the optimal choice of how many stages a network should be divided and how many layers per stage to future work. Here, we provide one experimental example to give some intuition about the impact of different splitting points. Figure 8 shows the convergence behavior for different splittings of VGG-11: four-stage and two-stage splitting as compared to single stage (conventional knowledge distillation). As seen in the figure, not only the convergence is faster as number of stages increases but also it eventually converges to a higher final accuracy as compared to lesser number of stages or no splitting at all.

3.6. Memory Analysis

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

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

4. Theoretical Analysis

One issue that arises as a result of the strategy of splitting into sections and training each section separately is accumulation of error residuals through sections which may impact the overall performance of the proposed technique. Here, we theoretically derive an upper bound on the total accumulated error across the resulting subnetworks after splitting using a chaining argument and utilizing Lipschitz continuity. A rigorous analysis bounding the Lipschitz constant of a deep network can be found in (Virmaux & Scaman, 2018) for arbitrary networks and (Zou et al., 2019) for particular convolutional networks.

We provide a worst case upper bound on the error, but it is also possible to establish probabilistic bounds on the error under the assumption that the quantization error on the weights is uniformly random. In particular, one can directly apply the bounds from (Sakr et al., 2017) to attain probabilistic bounds on the classification error even for our layer-wise quantization framework.

4.1. Upper Bounding Network-wide Error

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

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

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

Assume the application of our quantization scheme leads to an error in the output of size $\|\phi^{(i)}(x) - \phi_q^{(i)}(x)\| < \delta$. This comes from the quantization error guarantee of the used technique. Unless otherwise stated, $\|\cdot\|$ refers to the 2-norm. Further, assume that $\phi^{(i)}$ has Lipschitz constant L_i. Every one layer network is always a Lipschitz function, where L_i is always bounded by the norm of the weights matrix (see Appendix A.1 for a full description). Under this model, we can use a simple triangle inequality to get $\|\phi_q^{(i)}(x) - \phi_q^{(i)}(y)\| < L_i\|x-y\| + 2\delta$. Using this fact, and chaining it together across multiple layers, we are able to bound the pointwise error between the full precision network and the quantized network.

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

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

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

The proof can be found in the Appendix A.2.

As the Lipschitz constant of the network is the product of its individual layers' Lipschitz constants, L can grow exponentially if $L_i \geq 1$. This is the common case for normal network training (Cisse et al., 2017), and thus the perturbation will be amplified for such a network. Therefore, to keep the Lipschitz constant of the whole network small, we need to keep the Lipschitz constant of each layer $L_i < 1$. This is often done using regularization or weight clipping (Szegedy et al., 2013; Bartlett et al., 2017; Cisse et al., 2017; Gouk et al., 2018) to suppress network's accumulation of error. We call a network with $L_i < 1, \forall i = 1, ..., L$ a non-expansive network. Experimentally, Lipschitz constant of each layer is found empirically by taking $\max_{x,y} \|\phi_i(x) - \phi_i(y)\| / \|\phi_{i-1}(x) - \phi_{i-1}(y)\|$.

4.2. Lipschitz Constants in Classification Networks

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

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

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

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

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

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

Theorem 2. *Let f_{fp} and f_q be the full precision and quantized m layer networks as in Section 4.1. Let $L = \prod_{i=1}^{m} L_i$ be the Lipschitz constant of f_{fp}. Let (x_i, y_i) be a data point where f_{fp} correctly classifies x_i with output margin $r_i > 0$. Then for any*

perturbation η such that

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

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

The proof can be found in the Appendix A.2.

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

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

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

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

The proof can be found in the Appendix A.2. We note that r_i can be easily checked for a given full precision network by examining the last regression layer across all points in the data set.

5. Related Work

Knowledge distillation. Knowledge distillation (Hinton et al., 2015) is proposed to attain a smaller/shallower neural network (student) from one or an ensemble of bigger deep networks (teacher). The student network is trained on a softened version of the *final* output of teacher(s) (Bucila et al., 2006). FITNETS (Romero et al., 2015) extends knowledge distillation by extracting a hint from the teacher to train even a deeper but thinner student. The hint is an intermediate feature representation of the teacher, that is used as a regularizer to pretrain the first few layers of the deep and thin student network. After the pretraining phase, the full knowledge distillation is used to finish the training of the student. FITNETS (Romero et al., 2015) does not explore hints from more than one intermediate layer of the teacher. Furthermore, FITNETS applies the knowledge distillation pass over the entire student network at once. FITNETS are a complementary approach to our sectional knowledge distillation and similar hints can be utilized for each section. Nonetheless, the following discusses the differences. In contrast to this technique, DCQ (1) partitions the neural network to multiple independent sections and (2) applies knowledge distillation to each section in isolation and trains them independently, (3) not utilizing the intermediate representations as hint for pretraining. (4) After the sections are trained through knowledge distillation, they are put together instead of applying another phase of training as done in FITNETS (Romero et al., 2015). (5) Moreover, DCQ, exclusively, applies various regression losses in matching the quantized network intermediate feature maps to the corresponding full precision ones in a stage wise fashion. (6) Last but not least, the objective differ as the knowledge distillation and FITNETS aim to compress the network while DCQ quantizes it preserving the teacher's original network architecture.

Other work (Yim et al., 2017) proposes an information metric, in terms of inter-layer flow (the inner product of feature maps), using which a teacher DNN can transfer the distilled knowledge to other student DNNs.

Knowledge distillation is also used for training a lower bitwidth student network from a full-precision teacher (Mishra & Marr, 2018; Polino et al., 2018; Wang et al., 2019). However, these works do not partition the network as DCQ does and also do not utilize teacher's intermediate layers.

Other quantization techniques. Several techniques have been proposed for quantizing DNNs: algorithmic-wise (Zhou et al., 2016; Mishra et al., 2018; Zhu et al., 2017; Elthakeb et al., 2018; 2020), and hardware-wise (Ghodrati et al., 2020; Samragh et al., 2020).

DoReFa-Net (Zhou et al., 2016) uses straight through estimator (Bengio et al., 2013) for quantization and extends it for any arbitrary k bit quantization. DoReFa-Net also proposes a method to train a CNNs with low bitwidth weights and activations, low bitwidth parameter gradients using deterministic quantization of weights, activations and stochastic quantization of activations. TTQ (Zhu et al., 2017) proposes a method to reduce the weights to ternary values by adding scaling coefficients to each layer. These scaling coefficients are learnt during training and during deployment, weights are directly quantized to ternary bitwidths and these scaling coefficients are used to scale the weights during inference. PACT (Choi et al., 2018) proposes a technique for quantizing activations using an activation clipping parameter which is optimized during training. There have also been a lot of efforts (Rastegari et al., 2016a; Li & Liu, 2016; Hubara et al., 2017b) to binarize neural networks at the cost of some accuracy loss.

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

6. Conclusion

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

Acknowledgements

This work was in part supported by National Science Foundation (NSF) awards CN#1703812, ECCS#1609823, CCF#1553192, DMS#2012266, DMS#1819222, Semiconductor Research Corporation (SRC) contract #2019-SD-2884, Russel Sage Foundation award #2196, Air Force Office of Scientific Research (AFOSR) Young Investigator Program (YIP) award #FA9550-17-1-0274, National Institute of Health (NIH) award #R01EB028350, Air Force Research Laboratory (AFRL) and Defense Advanced Research Project Agency (DARPA) under agreement number #FA8650-20-2-7009 and #HR0011-18-C-0020, and gifts from Microsoft, Google, Qualcomm, Xilinx. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of Microsoft, Google, Qualcomm, Xilinx, SRC, Sage Foundation, NSF, AFSOR, NIH, AFRL, DARPA or the U.S. Government.

References

Bartlett, P. L., Foster, D. J., and Telgarsky, M. J. Spectrally-normalized margin bounds for neural networks. In *Advances in Neural Information Processing Systems*, pp. 6240–6249, 2017.

Bengio, Y., Léonard, N., and Courville, A. C. Estimating or propagating gradients through stochastic neurons for conditional computation. *CoRR*, abs/1308.3432, 2013. URL http://arxiv.org/abs/1308.3432.

Bucila, C., Caruana, R., and Niculescu-Mizil, A. Model compression. In Eliassi-Rad, T., Ungar, L. H., Craven, M., and Gunopulos, D. (eds.), *Proceedings of the Twelfth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Philadelphia, PA, USA, August 20-23, 2006*, pp. 535–541. ACM, 2006. ISBN 1-59593-339-5. doi: 10.1145/1150402.1150464. URL https://doi.org/10.1145/1150402.1150464.

Choi, J., Wang, Z., Venkataramani, S., Chuang, P. I.-J., Srinivasan, V., and Gopalakrishnan, K. Pact: Parameterized clipping activation for quantized neural networks. *CoRR*, abs/1805.06085, 2018.

Cisse, M., Bojanowski, P., Grave, E., Dauphin, Y., and Usunier, N. Parseval networks: Improving robustness to adversarial examples. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 854–863. JMLR. org, 2017.

Courbariaux, M., Bengio, Y., and David, J. Binaryconnect: Training deep neural networks with binary weights during propagations. In Cortes, C., Lawrence, N. D., Lee, D. D., Sugiyama, M., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pp. 3123–3131, 2015.

Elthakeb, A. T., Pilligundla, P., Mireshghallah, F., Yazdanbakhsh, A., and Esmaeilzadeh, H. ReLeQ: A reinforcement learning approach for deep quantization of neural networks. *Advances in Neural Information Processing Systems (NeurIPS) Workshop on Machine Learning for Systems*, 2018. URL https://arxiv.org/abs/1811.01704.

Elthakeb, A. T., Pilligundla, P., Mireshghallah, F., Elgindi, T., Deledalle, C.-A., and Esmaeilzadeh, H. WaveQ: Gradient-based deep quantization of neural networks through sinusoidal adaptive regularization. *arXiv preprint arXiv:2003.00146*, 2020.

Ghodrati, S., Sharma, H., Young, C., Kim, N. S., and Esmaeilzadeh, H. Bit-parallel vector composability for neural acceleration. *arXiv preprint arXiv:2004.05333*, 2020.

Gong, R., Liu, X., Jiang, S., Li, T., Hu, P., Lin, J., Yu, F., and Yan, J. Differentiable soft quantization: Bridging full-precision and low-bit neural networks. *CoRR*, abs/1908.05033, 2019. URL http://arxiv.org/abs/1908.05033.

Gouk, H., Frank, E., Pfahringer, B., and Cree, M. Regularisation of neural networks by enforcing lipschitz continuity. *arXiv preprint arXiv:1804.04368*, 2018.

Gupta, S., Agrawal, A., Gopalakrishnan, K., and Narayanan, P. Deep learning with limited numerical precision. In Bach, F. R. and Blei, D. M. (eds.), *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, volume 37 of *JMLR Workshop and Conference Proceedings*, pp. 1737–1746. JMLR.org, 2015. URL http://jmlr.org/proceedings/papers/v37/gupta15.html.

Hauswald, J., Laurenzano, M., Zhang, Y., Li, C., Rovinski, A., Khurana, A., Dreslinski, R. G., Mudge, T. N., Petrucci, V., Tang, L., and Mars, J. Sirius: An open end-to-end voice and vision personal assistant and its implications for future warehouse scale computers. In *ASPLOS*, 2015.

Hinton, G. E., Vinyals, O., and Dean, J. Distilling the knowledge in a neural network. *CoRR*, abs/1503.02531, 2015. URL http://arxiv.org/abs/1503.02531.

Hubara, I., Courbariaux, M., Soudry, D., El-Yaniv, R., and Bengio, Y. Quantized neural networks: Training neural networks with low precision weights and activations. *Journal of Machine Learning Research*, 18:187:1–187:30, 2017a. URL http://jmlr.org/papers/v18/16-456.html.

Hubara, I., Courbariaux, M., Soudry, D., El-Yaniv, R., and Bengio, Y. Quantized Neural Networks: Training Neural Networks with Low Precision Weights and Activations. *J. Mach. Learn. Res.*, 2017b.

Jaderberg, M., Czarnecki, W. M., Osindero, S., Vinyals, O., Graves, A., Silver, D., and Kavukcuoglu, K. Decoupled neural interfaces using synthetic gradients. In Precup, D. and Teh, Y. W. (eds.), *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, pp. 1627–1635. PMLR, 2017. URL http://proceedings.mlr.press/v70/jaderberg17a.html.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. Imagenet classification with deep convolutional neural networks. In Bartlett, P. L., Pereira, F. C. N., Burges, C. J. C., Bottou, L., and Weinberger, K. Q. (eds.), *Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems 2012. Proceedings of a meeting held December 3-6, 2012, Lake Tahoe, Nevada, United States.*, pp. 1106–1114, 2012.

LeCun, Y., Boser, B. E., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W. E., and Jackel, L. D. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1:541–551, 1989.

LeCun, Y., Bengio, Y., and Hinton, G. E. Deep learning. *Nature*, 521(7553):436–444, 2015. doi: 10.1038/nature14539. URL https://doi.org/10.1038/nature14539.

Li, F. and Liu, B. Ternary Weight Networks. *CoRR*, abs/1605.04711, 2016.

Mishra, A. and Marr, D. Apprentice: Using Knowledge Distillation Techniques To Improve Low-Precision Network Accuracy. In *International Conference on Learning Representations*, 2018.

Mishra, A. K., Nurvitadhi, E., Cook, J. J., and Marr, D. WRPN: Wide Reduced-Precision Networks. In *ICLR*, 2018.

Polino, A., Pascanu, R., and Alistarh, D. Model compression via distillation and quantization. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018. URL https://openreview.net/forum?id=S1XolQbRW.

Rastegari, M., Ordonez, V., Redmon, J., and Farhadi, A. XNOR-Net: ImageNet Classification Using Binary Convolutional Neural Networks. In *ECCV*, 2016a.

Rastegari, M., Ordonez, V., Redmon, J., and Farhadi, A. XNOR-Net: ImageNet Classification Using Binary Convolutional Neural Networks. In *European Conference on Computer Vision*, pp. 525–542, 2016b.

Romero, A., Ballas, N., Kahou, S. E., Chassang, A., Gatta, C., and Bengio, Y. Fitnets: Hints for thin deep nets. In Bengio, Y. and LeCun, Y. (eds.), *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. URL http://arxiv.org/abs/1412.6550.

Sakr, C., Kim, Y., and Shanbhag, N. Analytical guarantees on numerical precision of deep neural networks. In Precup, D. and Teh, Y. W. (eds.), *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 3007–3016, International Convention Centre, Sydney, Australia, 06–11 Aug 2017. PMLR. URL http://proceedings.mlr.press/v70/sakr17a.html.

Samragh, M., javaheripi, m., and Koushanfar, F. Encodeep: Realizing bit-flexible encoding for deep neural networks. *ACM Transactions on Embedded Computing Systems (TECS)*, 2020.

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., and Fergus, R. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

Virmaux, A. and Scaman, K. Lipschitz regularity of deep neural networks: analysis and efficient estimation. In *Advances in Neural Information Processing Systems*, pp. 3835–3844, 2018.

Wang, J., Bao, W., Sun, L., Zhu, X., Cao, B., and Yu, P. S. Private model compression via knowledge distillation. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019.*, pp. 1190–1197. AAAI Press, 2019. ISBN 978-1-57735-809-1. URL https://aaai.org/ojs/index.php/AAAI/article/view/3913.

Yim, J., Joo, D., Bae, J., and Kim, J. A gift from knowledge distillation: Fast optimization, network minimization and transfer learning. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pp. 7130–7138. IEEE Computer Society, 2017. ISBN 978-1-5386-0457-1. doi: 10.1109/CVPR.2017.754. URL https://doi.org/10.1109/CVPR.2017.754.

Zhang, D., Yang, J., Ye, D., and Hua, G. Lq-nets: Learned quantization for highly accurate and compact deep neural networks. In Ferrari, V., Hebert, M., Sminchisescu, C., and Weiss, Y. (eds.), *Computer Vision - ECCV 2018 - 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part VIII*, volume 11212 of *Lecture Notes in Computer Science*, pp. 373–390. Springer, 2018. ISBN

978-3-030-01236-6. doi: 10.1007/978-3-030-01237-3_23. URL https://doi.org/10.1007/978-3-030-01237-3_23.

Zhou, A., Yao, A., Guo, Y., Xu, L., and Chen, Y. Incremental network quantization: Towards lossless cnns with low-precision weights. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017. URL https://openreview.net/forum?id=HyQJ-mclg.

Zhou, S., Ni, Z., Zhou, X., Wen, H., Wu, Y., and Zou, Y. Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients. *CoRR*, abs/1606.06160, 2016. URL http://arxiv.org/abs/1606.06160.

Zhu, C., Han, S., Mao, H., and Dally, W. J. Trained Ternary Quantization. In *ICLR*, 2017.

Zmora, N., Jacob, G., and Novik, G. Neural network distiller, June 2018. URL https://doi.org/10.5281/zenodo.1297430.

Zou, D., Balan, R., and Singh, M. On lipschitz bounds of general convolutional neural networks. *IEEE Transactions on Information Theory*, 2019.