Learning to Quantize Deep Neural Networks: A Competitive-Collaborative Approach

Md Fahim Faysal Khan[§]
Department of EE
Pennsylvania State University
mzk591@psu.edu

Mohammad Mahdi Kamani[§]
College of IST
Pennsylvania State University
mqk5591@psu.edu

Mehrdad Mahdavi
Department of CSE
Pennsylvania State University
mzm616@psu.edu

Vijaykrishnan Narayanan
Department of CSE
Pennsylvania State University
vxn9@psu.edu

***Abstract*—Reducing the model size and computation costs for dedicated AI accelerator designs, neural network quantization methods have attracted momentous attention recently. Unfortunately, merely minimizing quantization loss using constant discretization causes accuracy deterioration. In this paper, we propose an iterative accuracy-driven learning framework of competitive-collaborative quantization (CCQ) to gradually adapt the bit-precision of each individual layer. Orthogonal to prior quantization policies working with full precision for the first and last layers of the network, CCQ offers layer-wise competition for any target quantization policy with holistic layer fine-tuning to recover accuracy, where the state-of-the-art networks can be entirely quantized without any significant accuracy degradation.**

***Index Terms*—Quantization, Mixed Precision, Deep Learning, Online Learning**

I. Introduction

Deep neural networks (DNNs) are tremendously revolutionizing numerous technologies and industries in the past few years, by achieving superb performance in a variety of tasks such as prediction and object detection. Consequently, there have been many works in design automation for accelerating the DNNs on FPGAs and ASICs [1]–[3] to run them on edge nodes. However, as prediction tasks are getting more complicated, the model complexity of DNNs is also growing dramatically, which requires more storage and computational power for storing, training, and inference. These issues pose a significant challenge to the applicability of DNNs on edge devices such as smartphones, or training them in a distributed fashion [4], [5] such as federated learning [6]. A solution to overcome the aforementioned challenges is to reduce the size of DNNs by quantization and make them computationally efficient by conducting arithmetic operations in a lower bit precision.

Quantized Neural Networks (QNNs) offer higher energy efficiency at lower inference time and computation cost which made them an active research area. Numerous quantization policies developed to train QNNs recently [7]–[12]. Notwithstanding their success, the existing methods mostly suffer from following key issues:

- **Quantization error driven.** While looking for the best quantization policies, the earlier methods focus mainly on the distribution of weights, activations and the quantization error [13]–[15]. Their main assumption arises from the fact that if we can minimize the perturbation in weight and activation distributions due to discretization, then the accuracy can be maintained. However, their objective is not accuracy-driven, hence, they are prone to degradation in accuracy of the resulted network.
- **Uniform precision.** Most of the works to date assume a uniform precision for weights and activations for different layers [7], [9]–[12], [16], [17]. Yet, from the recent studies [18], it has been asserted that different layers in a DNN have distinct representational capabilities and should be treated differently, according to their properties.
- **Quantizing first and last layers.** This fact that different layers should be treated differently, has been observed practically in various quantization-aware policies as well; where they avoid to quantize the first and last layers of DNNs since they might have devastating effects on the overall accuracy [10]–[12]. On the other hand, quantizing these two layers can greatly reduce the model size and computational power required for training and inference of these networks.

In pursuit of maintaining the overall accuracy for the quantization, we propose an *accuracy-driven* and *policy-agnostic* framework with variable bit precision for different layers dubbed as **competitive-collaborative** quantization. The framework is policy-agnostic, meaning that any quantization policy can be employed, and it can improve the accuracy of the network using that policy, compared to the best it can achieve separately. Also, it is accuracy-driven, which implies that we aim at deciding when and to what extent quantize each layer to have the minimum degradation on test accuracy. The proposed framework employs a gradual quantization from a higher bit precision to a lower one, instead of quantizing the parameters to the lower bit precision instantly. This approach will avoid a huge drop in accuracy, and it would be easy to recover between quantization steps. Specifically, we gradually quantize the layers, where at each quantization step, we run a competition between all layers, and based on the effect of their quantization to the next level of bit precision on the overall accuracy, one layer wins and gets quantized to the next level. After the competition stage and quantizing a single layer, since the overall accuracy drops, we run a fine-tuning step, in which all the layers collaborate with each other to recover the drop in the accuracy caused by the quantized layer. Using this framework, we also show that the first and last layers can be quantized to lower bit precision without having a drastic effect on the accuracy. Examining this framework on common datasets of CIFAR10 [19] and ImageNet [20] with ResNet [21] architectures, we empirically show the efficacy of the proposed framework over state-of-the-art methods.

II. Related Work

Being inspired by the efficiency and effectiveness of QNNs, there have been a quite number of studies on bit discretization of neural networks. These efforts can roughly fall into two categories, namely, post-training quantization, where we train with full precision and then quantize the trained model [14], [15]; and training with the quantized model, where quantization is done during training with the effect of quantization getting accounted within the main training loop [7], [9]–[12], [22]. Notwithstanding their differences, both approaches aim at having a compressed version of the model with reduced bit precision to have a faster inference procedure.

a) Static quantization: The first approach comprises of methods which take a pre-trained model and apply a fixed quantization formula. The key idea here is to minimize the quantization loss. For example, ACIQ [15] analytically determines the best clipping value

[§]Both student authors contributed equally by comparing the original weight distribution with standard distributions such as a Gaussian or a Laplacian. Nvidia's TensorRT [14] uses KL divergence to measure the relative entropy between full precision and quantized weights in order to find optimal clipping value. Static quantization schemes are the simplest ones in terms of implementation. However, it is hard to achieve accuracy at par with the baseline without enough retraining.

b) Quantization error driven: Suffering from the degradation issue, the next set of approaches takes the effect of quantization inside the main training loop, which is referred to as quantization-aware training. The approach was first introduced in BNN [7], where they quantize the parameters of the network to binary values of +1 and -1. Their idea was extended further in XNOR-net [9], where they quantize both the network parameters and their inputs to binary values so that a basic convolution operation becomes an XNOR operation. Later DoReFa [10] and WRPN [11] implement the idea for multiple bits assigned to both weights and activations while they clipped the activations in the range of $[0, 1]$. PACT [12] points out that clipping all activations uniformly to 1 may not be a good idea. Rather, they suggest learning the clipping value for each layer during the training. Later, they extend their earlier idea towards weight clipping as well [23]. While the aforementioned frameworks use uniform quantization step size for a given range, QIL [24] and LSQ [17] suggest that non-uniform step-size learned along the training provides better accuracy. In order to prevent the quantization loss during execution, Jain et al. [25] introduce a new fixed-point number format. The key focus is on keeping the quantization error as minimum as possible.

c) Learning-based: Very recently, there have been a few approaches targeting mixed-precision quantization for different layers and how to find them. HAQ [26] and ReLeQ [27] use reinforcement learning to learn different bit precision for different layers under various size, energy, latency, and compute constraints. Although the results are auspicious, the exploration phase for the agent is vast and can take a significantly long time. Another learning-based work, HAWQ [28], approximates the hessian for quantization of each of the layers/blocks and decides which layers/blocks to quantize. Rusci et al. [29] also explored the mixed-precision quantization domain where they did static quantization of weights and activations based on the memory constraints for micro-controllers. As they do not include any fine-tuning, their accuracy is not the best. Nonetheless, they show that having mixed-precision across different layers is very useful, especially for small devices.

III. Competitive-Collaborative Quantization

In this section, we discuss our training and quantization methodology with the implementation of our competitive-collaborative algorithm. As we have stated earlier, our algorithm is policy-agnostic, meaning, we can choose any quantization policy and implement our framework on top of it. As the name of the framework suggests, it consists of two stages, namely competition and collaboration. In the following subsections, we first briefly discuss the quantization aware training methodology and then elaborate on our algorithm in detail.

A. Problem formulation

In order to investigate the problem of quantization in deep neural networks, we first need to discuss the learning process of these machine learning models. Consider we have a neural network model $\boldsymbol{w} \in \mathbb{R}^d$, where it contains M layers (e.g., convolutional or fully-connected) each of which has a number of parameters to be learned. Formally, we denote each layer's parameter tensor by $\boldsymbol{W}_m,\ m \in [M]$. For instance, if the mth layer is a convolution layer, $\boldsymbol{W}_m$ is a 4-dimensional tensor. We represent the set of all layers' parameters as $\mathcal{W} = \{\boldsymbol{W}_1, \ldots, \boldsymbol{W}_M\}$, where the concatenation of all these parameters in a vector format is $\boldsymbol{w}$. In a supervised learning task, we want to perform a prediction mapping on a training dataset $\mathcal{T}$, with n training samples. This mapping is from a feature space $\mathcal{X}$ to a label space $\mathcal{Y}$, where each sample point is denoted by $(\boldsymbol{x}_i, y_i) \in \mathcal{X} \times \mathcal{Y}$. Then, in this task, the goal is to learn a classifier, $\boldsymbol{w}$, that has the minimum empirical loss on the training data for this mapping:

$$\mathcal{L}\left(\boldsymbol{w}; \mathcal{T}\right) = \frac{1}{n} \sum_{i=1}^{n} \ell\left(\boldsymbol{w}; \boldsymbol{x}_i, y_i\right), \tag{1}$$

where $\ell(.;.,.)$ is the loss function for each sample data. One main challenge is that the size of parameters, d, for each network is huge, which is a burden, both in terms of memory and computation. Moreover, these parameters and operations for feed-forward and feed-backward parts are done in full precision of 32 bits, while the parameter vector is heavily sparse. Considering these facts, quantization of these parameters to a lower bit precision seems necessary. Hence, practitioners use a quantization mapping function $\mathsf{Q}(.;.,.)$ to map values from a higher bit precision to the lower one:

$$\mathsf{Q}\left(\boldsymbol{z}; N, \alpha\right) = \boldsymbol{z}_q \in \mathcal{C}_{\alpha}^{N}, \tag{2}$$

where $\mathcal{C}_{\alpha}^{N} = \left\{0, \pm 2\alpha, \ldots, \pm 2^N \alpha\right\}$ is the N-bit representation domain with α as its scale. Based on this representation, $\boldsymbol{z} \in \mathcal{C}_{\beta}^{N_z}$, where $N_z > N$. The idea behind quite number of different quantization policies [10]–[12], [15], [23] is to find the optimum $\mathcal{C}_{\alpha}^{N}$, where for the parameter vector, $\boldsymbol{w}$, we can minimize the following quantization error:

$$\mathcal{L}_q(\boldsymbol{w}, \mathsf{Q}) = \left\|\boldsymbol{w} - \mathsf{Q}(\boldsymbol{w})\right\|_2^2. \tag{3}$$

On the other hand, considering the primary goal of the learning, which is a high accuracy, recently, it has been asserted that different layers in a neural network need different levels of bit precision for achieving the best accuracy [18]. Motivated by this observation, we propose a framework that takes into consideration the effect of each layer's quantization on the accuracy of the validation set $\mathcal{V}$; and then, decide which layer to pick and quantize. After the quantization of the picked layer, we will probably have degradation in the overall accuracy, which we can recover by fine-tuning all the layers simultaneously. Then, we will continue this cycle until we reach the desired compression rate and accuracy level. This cycle suggests a two-stage framework, in which in the first stage, each layer competes with other layers to be picked and quantized, and then all the layers collaborate with each other to recover the degradation from the quantization step. Despite other accuracy-driven approaches [28], we do not need any second-order information, which is a huge burden from a computational standpoint. We will elaborate on the details of this competitive-collaborative framework in what follows.

B. The proposed framework

As mentioned before, the main idea behind the proposed framework is that different parts of the model (e.g., layers) should have different levels of quantization since they are inherently different. This idea has been corroborated, even in schemes with the mere objective of quantization error [10]–[12], [23], where they do not quantize first and last layer, because it might heavily affect the accuracy of the network. Hence, in our framework we start each layer parameters $\mathcal{W}$ with the same level of quantization, $\boldsymbol{Q}_m^{(0)} = \mathsf{Q}\left(\boldsymbol{W}_m^{(0)}; N^{(0)}, \alpha_m^{(0)}\right),\ \forall m \in [M]$. Note that, in the first step, we quantize all the layers to the same level of bit precision $N^{(0)}$ (e.g., $N^{(0)} = 8$). In order to avoid a huge degradation in accuracy for each quantization step, we consider K different levels of bit precision from $N^{(0)}$ to $N^{(K-1)}$ (e.g., $N^{(K-1)} = 2$ or binary). This will help the framework to gradually decreases the bit precision of different layers without a huge drop in the accuracy. Then, the proposed algorithm iteratively alternates between competition and collaboration stages as discussed below:

a) Competition: Based on the discussion above, the goal is to find the best quantization level for each layer in the network. Searching through all possible choices, we need to train K^M different networks and choose the best one in terms of the overall accuracy on the validation set and the compression rate. The reason is that we do not know how different layers with different levels of quantization would affect accuracy. Hence, we advocate the use of the online learning framework [30], where at each step, we will have access to the full information from all layers. The information in this setting is the accuracy of validation based on the model where all layers' parameters are kept at the same level of quantization as before, except for one that is quantized to the next level of bit precision. Using the resulted network, we calculate the validation loss. Then, we will repeat this step for all layers in the network several times.

This is a cheap operation in terms of computation since it is a simple feed-forward on a small validation set, in contrast to the large training dataset. Using this information, we know the effect of the quantization of each layer at that time on the overall accuracy. The ultimate goal of this stage is to gradually learn a probability distribution $\boldsymbol{p} = [p_1, \dots, p_M]$, based on the effect of quantization of each layer to the next level of bit precision on the validation accuracy. Hence, having a higher validation accuracy (or the least degradation) is reflected in a high probability of picking that layer for quantization at this step. We start this probability distribution from a uniform distribution, $p_m = 1/M,\ \forall m \in [M]$, and then update these probabilities by accessing to more information about the layers. Formally, if at step $t > 0$ of the quantization, we have the quantized weight set of the previous step as $\mathcal{Q}^{(t-1)} = \left\{\boldsymbol{Q}_1^{(t-1)}, \dots, \boldsymbol{Q}_M^{(t-1)}\right\}$, each of which with possibly different quantization level from $N^{(0)}$ to $N^{(K-1)}$. Then, we randomly choose one layer based on the probability distribution $\boldsymbol{p}$, say i, that has the bit precision of $N^{(k)}, 0 \le k < K-1$, and quantize it to the next level of bit precision, $N^{(k+1)}$. Note that, if a layer already has the minimum bit precision of $N^{(K-1)}$, we will not use it for quantization and consider it as a sleeping expert. Then we evaluate the resulting network using the loss on the validation set:

$$\xi_m^{(t)} = \mathcal{L}\left(\mathcal{Q}_m^{(t)}; \mathcal{V}\right) = \frac{1}{n_v}\sum_{i=1}^{n_v} \ell\left(\boldsymbol{x}_i, y_i; \mathcal{Q}_m^{(t)}\right), \tag{4}$$

where n_v is the number of validation samples and:

$$\mathcal{Q}_m^{(t)} = \left\{\boldsymbol{Q}_1^{(t-1)}, \dots, \boldsymbol{Q}_m^{(t)}, \dots, \boldsymbol{Q}_M^{(t-1)}\right\}, \quad m \in [M] \tag{5}$$

which is the previous step quantization set, except for the mth layer that has been quantized to the next level of bit precision. Note that for this set we are calculating the mth layer quantization as $\boldsymbol{Q}_m^{(t)} = \mathsf{Q}\left(\boldsymbol{Q}_m^{(t-1)}, N^{(k+1)}, \alpha_i^{(t)}\right)$ using any chosen quantization operator. We run this operation of competition for U steps, and at each step u based on this observation, we will update the weights and probability distribution:

$$p_m^{(u)} = \frac{\pi_m^{(u)}}{\sum_{i=1}^{M} \pi_i^{(u)}}, \tag{6}$$

where $p_m^{(u)}$ is the probability of choosing the mth expert (i.e. layer). Also, $\pi_i^{(u)}$ is the weight of each layer calculated based on the validation loss. After calculating every layer's probability, based on their distribution we will choose a layer and quantize it to the next level. Hence, if the m_tth layer at the tth quantization step is chosen, the parameter set will be updated as $\mathcal{Q}^{(t)} = \mathcal{Q}_{m_t}^{(t)}$.

b) Collaboration: After running a competition between all layers and have a winner m_t, most probably, we will have a decrease in the overall accuracy of the network. Hence, we will run a fine-tuning step and update parameters from all the layers together, in order to recover the overall accuracy. We call this stage collaboration since all the layers will adapt their parameters to the recent changes

Algorithm 1 Competitive-Collaborative Framework

```
1: input $\mathcal{W}^{(0)} = \left\{\boldsymbol{W}_1^{(0)}, \dots, \boldsymbol{W}_M^{(0)}\right\}$, $p_m^{(0)} = 1/M\ \forall m \in [M]$,
   $\pi_m^{(0)} = 1\ \forall m \in [M]$, Quantization Policy $\mathsf{Q}(.;.,.)$
2: procedure CCQ( $\mathcal{W}^{(0)}$, $\boldsymbol{p}^{(0)}$, $\boldsymbol{\pi}^{(0)}$,$\mathsf{Q}(.;.,.)$ )
3:   Initialize layers $\mathcal{Q}^{(0)} = \left\{\boldsymbol{Q}_1^{(0)}, \boldsymbol{Q}_2^{(0)}, \dots, \boldsymbol{Q}_M^{(0)}\right\}$ to the same
     $N^{(0)}$ bit precision.
4:   for $t = 1, \dots, T$ do
5:     Calculate $\mathcal{Q}_1^{(t)}, \dots, \mathcal{Q}_M^{(t)}$ with $\mathsf{Q}(.;.,.)$ using (5)
6:     for $u = 1, \dots, U$ do                                    ▷ Competition
7:       Pick a layer $m_u \propto \boldsymbol{p}^{(u-1)}$
8:       Incur the validation loss $\xi_{m_u}^{(u)} = \mathcal{L}\left(\mathcal{Q}_{m_u}^{(t)}; \mathcal{V}\right)$
9:       Update weights as $\pi_{m_u}^{(u)} = \pi_{m_u}^{(u-1)} \exp\left(-\gamma \xi_{m_u}^{(u)}\right)$
10:      Update probabilities as $p_{m_u}^{(u)} = \frac{\pi_{m_u}^{(u-1)}}{\sum_{i=1}^{M} \pi_i^{(u-1)}}$
11:    Randomly draw $m_t \propto \left\{p_1^{(U)}, \dots, p_M^{(U)}\right\}$.
12:    set $\mathcal{Q}_f^{(1)} \longleftarrow \mathcal{Q}_{m_t}^{(t)}$
13:    set $\mathcal{W}_f^{(1)} \longleftarrow \mathcal{W}^{(t-1)}$
14:    for $s = 1, \dots, S_t - 1$ do                              ▷ Collaboration
15:      Calculate the training loss $\mathcal{L}\left(\mathcal{Q}_f^{(s)}; \mathcal{T}\right)$
16:      Calculate the gradients $\mathcal{G}^{(s)} = \nabla \mathcal{L}\left(\mathcal{Q}_f^{(s)}; \mathcal{T}\right)$
17:      Update weights $\mathcal{W}_f^{(s+1)} = \mathcal{W}_f^{(s)} - \eta_s \mathcal{G}^{(s)}$
18:      Quantize the weights $\mathcal{Q}_f^{(s+1)} = \mathsf{Q}\left(\mathcal{W}_f^{(s+1)}\right)$
19:    set $\mathcal{Q}^{(t)} \longleftarrow \mathcal{Q}_f^{(S_t)}$
20:    set $\mathcal{W}^{(t)} \longleftarrow \mathcal{W}_f^{(S_t)}$
   return $\mathcal{Q}^{(T)} = \left\{\boldsymbol{Q}_1^{(T)}, \boldsymbol{Q}_2^{(T)}, \dots, \boldsymbol{Q}_m^{(T)}\right\}$
```

(i.e., quantization of m_tth layer to the next bit precision) in the network and recover the accuracy. This stage can be done in two different modes, that is, manual or adaptive recovery. For the manual recovery, we can set the number of epochs S_t for fine-tuning after each quantization step beforehand. The rule of thumb is that, as we quantize more layers to the lower bit precision, we should expect a higher degradation in the accuracy, and hence, need more epochs of fine-tuning to recover. However, in practice, it can be inferred that this is not quite right, and the degradation does not necessarily worsen as we quantize more layers. Thus, we use the second approach, adaptive recovery, in which we set a threshold for accuracy, and continue the fine-tuning until we reach to that predefined threshold. This means that the number of epochs S_t will be set adaptively in training. We observed that some quantization steps need only one epoch to recover, while some others need more than several epochs to fully recover the threshold. For more details on the collaboration part, the reader is referred to the next section.

IV. Experimental Results

In this section, we first describe our experimental settings and then move onto discussing and comparing our results and findings with other concurrent works. Next, we investigate the learning behavior of our proposed algorithm, followed by the details of the retraining phase of the networks. We also discuss the power requirements of fully quantized versus partially quantized networks to further demonstrate the efficacy of our proposed mixed precision framework.

Quantization Scheme	DoReFa [10] fp-3b-fp	WRPN [11] fp-3b-fp	PACT [12] fp-3b-fp
One-shot Quantization	89.9	87.9	91.1
Ours (Gradual)	91.8	89.33	91.94

TABLE I: Comparison between one-shot quantization versus gradual quantization. We reach the same bit configuration gradually using our framework and find that our accuracy driven approach gives the best result in all of the three cases. We run this experiment using ResNet20 on CIFAR10. For WRPN, the accuracy for one-shot quantization is calculated by ourselves.

a) Experimental settings: For our experiments, we use CIFAR10 [19] and ImageNet [20] datasets. CIFAR10 is an image classification dataset of 10 classes with 50000 training and 10000 validation images of size $32\times32\times3$. ImageNet contains 1000 classes consisting of 1.2 million training samples and 50000 validation samples. We mostly use ResNet models as they are vastly being used by other approaches as well. For the fine-tuning part, all the standard data augmentations such as random cropping ($224\times224\times3$ for ImageNet) and flipping have been used. For fine-tuning, we choose stochastic gradient descent as the optimizer with the quantization aware training methodology.

b) Policy-agnostic framework: We first demonstrate the efficacy of our algorithm as a standalone framework regardless of any of the quantization policies. In order to do that, we choose three already available and popular quantization policies: DoReFa, WRPN, and PACT. We call the typical learning using these policies as *one-shot quantization*. To prove the versatility of our proposed framework, we take a fixed bit pattern reported by them and force our framework to reach the same bit configuration by using the very same quantization policy. Table I shows that our framework achieves better results in all of the three cases. Even though both of the networks reach the same bit configuration, the accuracy improves when we follow a layer-wise gradual quantization scheme with an accuracy-driven goal. For quantization policy, we find PACT to be the most effective in our case. As we change the bit precision layer by layer gradually, the optimal clipping values for those layers change. Hence, PACT learns this clipping parameter throughout the training and can adapt well with the sudden change in bit-width. In our subsequent experiments, we mainly use PACT as our quantization policy and report the numbers based on that.

c) Model compression: Quantization not only reduces the compute cost but also leads to a higher model compression ratio. In order to have the memory footprint in the loop, we introduce a parameter λ, which determines how aggressively we want to reduce the memory. The idea is that, in addition to the effect of each layer's quantization on the validation loss, we want to quantize bigger layers (in terms of memory) sooner in the pipeline. As mentioned in Section III-B, the layers are randomly picked according to a corresponding probability distribution we learned during the competition steps. We modify this probability distribution to include the effect of memory each layer with respect to the total network size:

$$p_{m_{\text{new}}}^{(t)} = (1-\lambda)p_m^{(t)} + \lambda\frac{|\boldsymbol{Q}_m^{(t)}|}{\sum_{i=1}^{M}|\boldsymbol{Q}_i^{(t)}|}, \tag{7}$$

where $\lambda \in [0,1]$ and $|\boldsymbol{Q}_m^{(t)}|$ is the size of the mth layer at step t. Having a very high λ will compress the model fast, whereas a low λ would be more accuracy-driven. We use a linear decay to decrease the value of λ over the time since, in the beginning, most of the layers are at high precision, making it comparatively easier to recover. On the other hand, as the number of quantization steps increases, the recovery is harder; thus, we need to be more accuracy-driven by lowering the λ. To choose the best policy for λ, we need to tune it as a hyperparameter. Fig. 1 shows the overall accuracy trend for different average λ for training CIFAR10 on ResNet20.

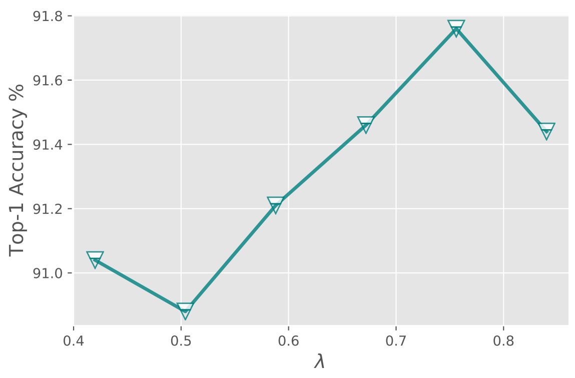

Fig. 1: Accuracy for different λ values to take the model size into account according to (7). The average λ value within the vicinity of $\sim 0.6-0.7$ gives the best results.

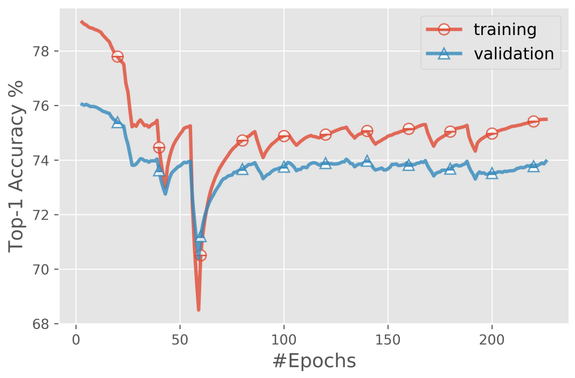

Fig. 2: Learning curves for Competitive-Collaborative scheme. The valleys indicate the accuracy degradation after a competition for a quantization step. The peaks denote the recovery due to collaborative efforts from all the layers.

d) Quantization results: Next, we present the results on CIFAR10 and ImageNet datasets summarized in Table II. As all the other reported works have different baselines, we decide to use the degradation in accuracy as our primary point of comparison rather than using absolute accuracy numbers. For CIFAR10, we get almost as same accuracy as the baseline leading towards the least accuracy degradation($\sim$0.06) with a very high model compression ratio ($\sim$10x). For ResNet18 on ImageNet we achieved the highest compression ratio($\sim$9.7x) with the least amount of accuracy degradation($\sim$2.6%). For ResNet50, we get the least accuracy degradation number($\sim$1.45%) with a very high model compression rate.

e) Learning curve: According to our competitive-collaborative strategy, all the layers compete against each other at each competition step to get selected for quantization. After quantization, we incur a degradation in the accuracy, hence, all the other layers collaborate to compensate for the loss due to the quantization. The learning curves in Fig. 2 very well illustrates these steps and the fact that competition hurts while collaboration helps. The valleys indicate the degradation in accuracy after a quantization step, followed by a gradual recovery where all the other layers collaboratively participate.

f) Adaptive recovery: The recovery phase basically consists of quantization-aware-retraining for few epochs. As the number of

Dataset & Arch	Framework	Baseline Top-1%	Bits (W/A)	first/last	Quantized Top-1%	Model Compression	Degradation
ResNet20 on CIFAR10	DoReFa [10]	91.8	3/3	32/32	89.90	10.27x	1.9
	PACT [12]	91.6	4/4	32/32	91.3	7.78x	0.3
	PACT-SAWB [23]	91.8	2/2	32/32	90.65	<15.1x	1.15
	LQ-Nets [16]	92.1	3/3	32/32	91.6	10.27x	0.5
	HAWQ [28]	92.37	MP	MP	92.22	13.11x	0.15
	PACT+CCQ(Ours)	92.34	MP	MP	92.28	10.1x	0.06
ResNet18 on ImageNet	DoReFa [10]	70.2	2/2	32/32	62.6	9.52x	7.6
	PACT [12]	70.2	2/2	32/32	64.4	9.52x	5.8
	PACT-SAWB [23]	70.4	2/2	32/32	67.0	9.52x	3.4
	LQ-Nets [16]	70.3	2/2	32/32	64.9	<9.52x	5.4
	QIL [24]	70.5	2/2	32/32	65.7	9.52x	4.8
	PACT+CCQ(Ours)	70.1	MP	MP	67.5	9.75x	2.6
ResNet50 on ImageNet	DoReFa [12]	76.9	3/3	32/32	67.10	6x	9.8
	PACT [12]	76.9	3/3	32/32	72.2	6x	4.7
	PACT-SAWB [23]	76.9	2/2	32/32	74.2	<7.24x	2.7
	LQ-Nets [16]	76.4	2/2	32/32	74.2	7.57x	2.4
	HAWQ [28]	77.39	MP	MP	75.48	12.28x	1.91
	PACT+CCQ(Ours)	77.09	MP	MP	75.64	8.47x	1.45

TABLE II: Comparison of our approach with other related ones. We report the top-1 accuracy and compare the degradation from baseline after quantization. The *"first/last"* column denotes the bit precision for the first & last layers respectively. *MP* stands for mixed-precision architectures where each layer has different bits. For CIFAR10, we have the least accuracy degradation with $\sim$ 10.1x model compression. For ImageNet, our quantized ResNet18 gives the best results for both model compression($\sim$ 9.75x) and accuracy($\sim$ 2.6% degradation from baseline). With ResNet50 on ImageNet, we are able to attain the least accuracy degradation($\sim$ 1.45%) with $\sim$ 8.5x compression ratio.

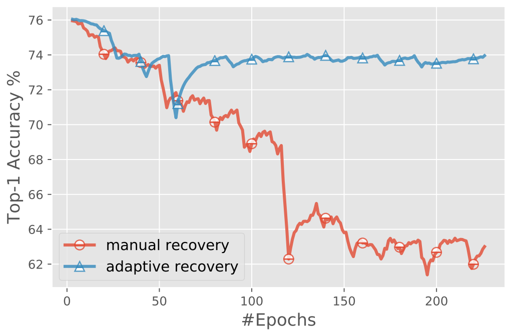

Fig. 3: Manual versus adaptive recovery for our framework. Having a predefined number of recovery epochs does not guarantee a full recovery. On the other hand, with adaptive recovery, the length of fine-tuning steps can be controlled more efficiently.

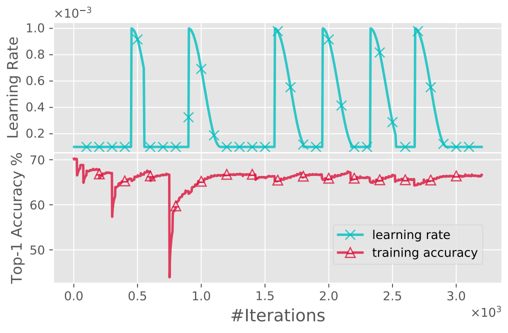

Fig. 4: Illustration of our hybrid learning rate schedule with the corresponding learning curve. Whenever the learning plateaus, a slight increase in learning rate can expedite the learning.

quantization steps increases, the model becomes more and more quantized. Intuitively, it should require more number of epochs to recover. Keeping this in mind, we first manually choose the number of recovery epochs in an increasing manner. This seems to work reasonably well for a few quantization steps. However, if at some point, the network fails to converge, then it becomes extremely difficult to recover from the subsequent quantization steps. In order to mitigate this issue, we introduce an adaptive recovery scheme where we keep retraining until the network reaches an accuracy threshold. We also observe that in case of certain quantization steps, the network can recover within one or two epochs, and hence, do not need so many epochs for recovery. Fig. 3 illustrates the behavior of these two recovery schemes, which is for the quantization of ResNet50 on ImageNet. In order to expedite the recovery process, motivated by [31], we use a hybrid learning rate where we switch to the cosine learning rate decay schedule upon reaching a plateau.

g) Adaptive learning rate schema: We start fine-tuning with a constant learning rate (1e-4 for ImageNet) and retrain the model for a few epochs. If the learning plateaus, i.e., the model fails to recover, we slightly increase the learning rate and follow a cosine decay rule to get back to the previous value. Repetitive quantization and fine-tuning can lead the network to a local optimum. By slightly increasing the learning rate, we introduce a perturbation and help the network come out of the local saddle point. Fig. 4 provides one such example for ResNet18 on ImageNet.

h) Reduction in power consumption: Quantization of all the layers leads to a significant reduction in power requirement as well. We synthesized the RTL module for a MAC (Multiply and Accumulate) unit from the DesignWare Libary [32] under 32nm technology node. Fig. 5 depicts the power consumption profile when we compare partially quantized networks having full precision in the first and last layers to our fully quantized ones in mixed precision. Note that, the reported power numbers are estimated for an iso-throughput scenario. *fp-4b-fp* or *fp-2b-fp* indicate the first and last layers being in full precision and other layers in uniform 4 or 2 bits, respectively. Even after having comparatively less number of parameters, the full precision MAC calculation for the first and last layers requires 4 $\sim$ 56$\times$ more power compared to the total power of all the other layers operating at quantized state. The fully quantized networks explored here, ResNet20_CIFAR, ResNet18, and ResNet50 have 6/2, 6/6, and 8/3 bits in the first/last layers, respectively. Consequently, our proposed fully quantized networks have the least power budget in the order of magnitude for all design exploration.

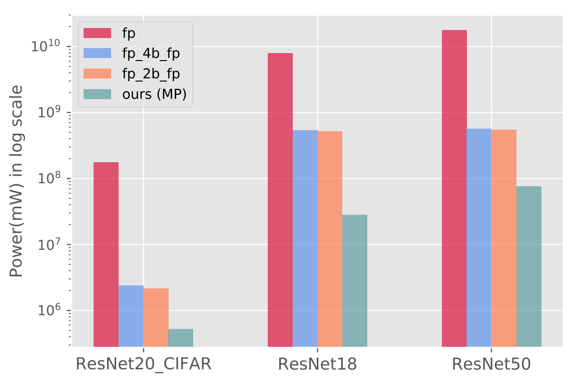

Fig. 5: The power consumed by the unquantized, partially quantized, and fully quantized models at the iso-throughput condition. The partially quantized networks where the first and last layers are operating at full precision (*fp-4b-fp* or *fp-2b-fp*) consume significantly more power compared to the fully quantized networks.

ACKNOWLEDGMENT

This work was supported in part by C-BRIC (Center for Brain-inspired Computing), one of six centers in JUMP, a Semiconductor Research Corporation (SRC) program sponsored by DARPA.

V. CONCLUSION AND FUTURE WORK

We propose a novel quantization framework compatible with any quantization policy and show how our framework is able to learn different precision levels for different layers inside a neural network. This framework runs a competition between layers to be selected and quantized to the next level, and then all layers collaborate to recover the degradation of accuracy from quantization. Hence, it has mixed-precision levels for different layers, that is learned gradually through the competition and collaboration stages. Moreover, the accuracy-driven approach helps maintain the accuracy at par with the baseline. Finally, we show that the first and last layers can also be quantized to significantly low precision levels by gradual quantization while preserving the accuracy. The overall results prove the efficacy of our proposed framework in design automation and show promises for the mixed-precision networks.

REFERENCES

[1] K. Kwon, A. Amid, A. Gholami, B. Wu, K. Asanovic, and K. Keutzer, "Co-design of deep neural nets and neural net accelerators for embedded vision applications," in *2018 55th ACM/ESDA/IEEE Design Automation Conference (DAC)*. IEEE, 2018, pp. 1–6.

[2] D. Wang, K. Xu, Q. Jia, and S. Ghiasi, "Abm-spconv: A novel approach to fpga-based acceleration of convolutional neural network inference," in *Proceedings of the 56th Annual Design Automation Conference 2019*. ACM, 2019, p. 87.

[3] C. Hao, X. Zhang, Y. Li, S. Huang, J. Xiong, K. Rupnow, W.-m. Hwu, and D. Chen, "FPGA/DNN Co-Design: An Efficient Design Methodology for IoT Intelligence on the Edge," in *Proceedings of the 56th Annual Design Automation Conference (DAC)*, 2019, pp. 206:1–206:6.

[4] F. Haddadpour, M. M. Kamani, M. Mahdavi, and V. Cadambe, "Trading redundancy for communication: Speeding up distributed sgd for non-convex optimization," in *International Conference on Machine Learning*, 2019, pp. 2545–2554.

[5] ——, "Local sgd with periodic averaging: Tighter analysis and adaptive synchronization," in *Advances in Neural Information Processing Systems*, 2019, pp. 11 080–11 092.

[6] F. Haddadpour and M. Mahdavi, "On the convergence of local descent methods in federated learning," *arXiv preprint arXiv:1910.14425*, 2019.

[7] M. Courbariaux, I. Hubara, D. Soudry, R. El-Yaniv, and Y. Bengio, "Binarized neural networks: Training deep neural networks with weights and activations constrained to+ 1 or-1," *arXiv preprint arXiv:1602.02830*, 2016.

[8] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio, "Binarized neural networks," in *Advances in neural information processing systems*, 2016, pp. 4107–4115.

[9] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, "Xnor-net: Imagenet classification using binary convolutional neural networks," in *European Conference on Computer Vision*. Springer, 2016, pp. 525–542.

[10] 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.

[11] A. Mishra, E. Nurvitadhi, J. J. Cook, and D. Marr, "Wrpn: wide reduced-precision networks," *arXiv preprint arXiv:1709.01134*, 2017.

[12] 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.

[13] L. Hou, Q. Yao, and J. T. Kwok, "Loss-aware binarization of deep networks," *arXiv preprint arXiv:1611.01600*, 2016.

[14] S. Migacz, "8-bit inference with tensorrt," in *GPU Technology Conference*, vol. 2, 2017, p. 7.

[15] R. Banner, Y. Nahshan, E. Hoffer, and D. Soudry, "Aciq: Analytical clipping for integer quantization of neural networks," *arXiv preprint arXiv:1810.05723*, 2018.

[16] 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.

[17] S. K. Esser, J. L. McKinstry, D. Bablani, R. Appuswamy, and D. S. Modha, "Learned step size quantization," *arXiv preprint arXiv:1902.08153*, 2019.

[18] C. Zhang, S. Bengio, and Y. Singer, "Are all layers created equal?" *arXiv preprint arXiv:1902.01996*, 2019.

[19] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," Citeseer, Tech. Rep., 2009.

[20] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A Large-Scale Hierarchical Image Database," in *CVPR09*, 2009.

[21] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[22] 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.

[23] J. Choi, P. I.-J. Chuang, Z. Wang, S. Venkataramani, V. Srinivasan, and K. Gopalakrishnan, "Bridging the accuracy gap for 2-bit quantized neural networks (qnn)," *arXiv preprint arXiv:1807.06964*, 2018.

[24] 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.

[25] S. Jain, S. Venkataramani, V. Srinivasan, J. Choi, P. Chuang, and L. Chang, "Compensated-dnn: energy efficient low-precision deep neural networks by compensating quantization errors," in *2018 55th ACM/ESDA/IEEE Design Automation Conference (DAC)*. IEEE, 2018, pp. 1–6.

[26] K. Wang, Z. Liu, Y. Lin, J. Lin, and S. Han, "Haq: Hardware-aware automated quantization with mixed precision," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 8612–8620.

[27] A. Yazdanbakhsh, A. T. Elthakeb, P. Pilligundla, F. Mireshghallah, and H. Esmaeilzadeh, "Releq: An automatic reinforcement learning approach for deep quantization of neural networks," *arXiv preprint arXiv:1811.01704*, 2018.

[28] Z. Dong, Z. Yao, A. Gholami, M. Mahoney, and K. Keutzer, "Hawq: Hessian aware quantization of neural networks with mixed-precision," *arXiv preprint arXiv:1905.03696*, 2019.

[29] M. Rusci, A. Capotondi, and L. Benini, "Memory-driven mixed low precision quantization for enabling deep network inference on microcontrollers," *arXiv preprint arXiv:1905.13082*, 2019.

[30] S. Shalev-Shwartz *et al.*, "Online learning and online convex optimization," *Foundations and Trends® in Machine Learning*, vol. 4, no. 2, pp. 107–194, 2012.

[31] I. Loshchilov and F. Hutter, "Sgdr: Stochastic gradient descent with warm restarts," *arXiv preprint arXiv:1608.03983*, 2016.

[32] Synopsys, Inc. , [online]. Available: https://www.synopsys.com/designware-ip/soc-infrastructure-ip/designware-library.html, [Accessed: Nov-27-2019].