Explicit Loss-Error-Aware Quantization for Low-Bit Deep Neural Networks

Aojun Zhou[1]* Anbang Yao[1]* Kuan Wang[2] Yurong Chen[1]
[1]Intel Labs China
[2]Department of Automation, Tsinghua University
[1]{aojun.zhou, anbang.yao, yurong.chen}@intel.com [2]wangkuan15@mails.tsinghua.edu.cn

Abstract

Benefiting from tens of millions of hierarchically stacked learnable parameters, Deep Neural Networks (DNNs) have demonstrated overwhelming accuracy on a variety of artificial intelligence tasks. However reversely, the large size of DNN models lays a heavy burden on storage, computation and power consumption, which prohibits their deployments on the embedded and mobile systems. In this paper, we propose Explicit Loss-error-aware Quantization (ELQ), a new method that can train DNN models with very low-bit parameter values such as ternary and binary ones to approximate 32-bit floating-point counterparts without noticeable loss of predication accuracy. Unlike existing methods that usually pose the problem as a straightforward approximation of the layer-wise weights or outputs of the original full-precision model (specifically, minimizing the error of the layer-wise weights or inner products of the weights and the inputs between the original and respective quantized models), our ELQ elaborately bridges the loss perturbation from the weight quantization and an incremental quantization strategy to address DNN quantization. Through explicitly regularizing the loss perturbation and the weight approximation error in an incremental way, we show that such a new optimization method is theoretically reasonable and practically effective. As validated with two mainstream convolutional neural network families (i.e., fully convolutional and non-fully convolutional), our ELQ shows better results than state-of-the-art quantization methods on the large scale ImageNet classification dataset. Code will be made publicly available.

1. Introduction

In the past half a decade, we have witnessed tremendous success of Deep Neural Networks (DNNs) in many artificial intelligence domains such as computer vision [22], speech recognition [12], natural language processing [2], Go games [31] and so forth. The pioneering AlexNet [22] proposed by Krizhevsky et al. ignites the resurgence of the popularity of DNNs. From then on, there is a clear trend that the architecture of mainstream DNNs is evolved to be significantly deeper and more complex than the 8-layer AlexNet. This is vividly demonstrated with the development of Convolutional Neural Networks (CNNs) [32] [35] [10] [41] [37] [18] which are now prevailing in the computer vision community. There are also some works [1] [16] [42] that make special efforts on designing more efficient network architectures from the perspective of reshaping dense convolutions into depth-wise separable convolutions. Despite these great advances on network design and accuracy improvement, the intensive storage and computational costs of these top-performing DNN models make it difficult to deploy them on the mobile and embedded systems for real-time applications.

With the drive to make DNN solutions be applicable on low-power devices, substantial research efforts have been invested in DNN compression and acceleration both in academia and industry. In this paper, our focus is restricted to DNN quantization. Specifically, we intend to address the problem of how to train DNN models whose weights are forced to be very low-bit values such as ternary and binary ones without noticeable loss of model accuracy when compared with full-precision (i.e., 32-bit floating-point) counterparts. By representing DNN models with very low-bit parameter values such as $\{-1, 0, 1\}$ and $\{-1, 1\}$ multiplied with layer-wise scaling factors, it would bring great benefits to the applications of DNN solutions, especially on the specialized deep learning hardware where originally time-intensive multiplication operations can be replaced by simple bit-shift and accumulation operations. Our method differs greatly with the known methods from the perspective of optimization formulation. Existing methods [4] [23] [28] [45] usually achieve ternary or binary quantization goal by a straightforward approximation of the layer-wise weights or inner product outputs of the full-precision model. In the

*This work was done when Aojun Zhou was an intern at Intel Labs China supervised by Anbang Yao who is responsible for correspondence. Intern Kuan Wang contributed to the experiments. The first two authors jointly wrote the paper with equal contribution.

approximation, they try to minimize the error of the layer-wise weights or inner products of the weights and the inputs between the original and respective quantized models. However, a critical fact is that replacing 32-bit floating-point weight values with very low-bit equivalents will introduce fluctuations to weight and output magnitudes, thus regularizing respective approximation error is necessarily important. Beyond that, it will bring perturbation on the classification loss which would influence the predication accuracy of the quantized DNN models, therefore a careful handling of such a loss perturbation is also critical to suppress common model accuracy loss. To alleviate this problem, Hou et al. [15] propose a proximal Newton algorithm based quantization method that directly minimizes the loss w.r.t. the binarized weights. In [14], they further extend this idea to handle ternary quantization task. With proximal Newton algorithm, a close-form solution can be derived, but it needs to estimate the second order Hessian matrix of the loss function w.r.t. the quantized weights and the input activations, bringing unacceptable computational complexity which prohibits its using in the training with a large-scale dataset such as ImageNet. Unlike these methods, in this paper, we present a new and efficient quantization scheme that considers aforementioned two critical factors jointly during the optimization. Our main contributions are summarized as follows.

1. We present Explicit Loss-error-aware Quantization (ELQ), a new DNN quantization method that jointly regularizes the weight approximation error and the accompanying loss perturbation in an explicit manner. To train lossless quantized models, we further bridge our ELQ with an incremental quantization strategy.

2. We show that our approach is theoretically reasonable and practically effective. It achieves state-of-the-art results on the large scale ImageNet classification dataset, as validated with two mainstream CNN architecture families (i.e., fully convolutional and non-fully convolutional).

2. Related Works

In the literature, there exists numerous methods for DNN compression and acceleration, and the reader is referred to [34] for a recent survey. Here, we just briefly summarize three major solution families that are related to our method.

Knowledge Distillation. One way to train a smaller student network while maintaining the similar accuracy of the large teacher is knowledge distillation. A pioneering work is [13] in which Hinton et al. take the soft predication outputs from a large and powerful teacher model or an ensemble of the trained models as the hints to jointly regularize the optimization objective when training a smaller neural network with the given data and the corresponding one-hot labels. Later on, several works [29] [40] attempt to use both final predications and intermediate representations from the ensemble teachers to enhance the training of target student model. Recently, [26] and [27] show that the knowledge of large teacher networks can be well applied to improve model accuracy during the training and quantization of smaller student networks.

Network Pruning. The basic goal of network pruning is to compress any dense network architecture into a sparse version without loss of model accuracy. This line of research can be dated back to the late 1980s. Two representative works are brain damage [39] and brain surgeon [9] in which the authors derive optimal sparse network architecture from the perspective of information-theoretic optimization. Both methods suffer from the high computational complexity since they need to compute second order derivatives of the loss function, thus they cannot handle large and deep neural networks. Recently, Han et al. [8] revisit this topic and introduce a simple yet effective pruning strategy which gets impressive network compression and inference performance as tested with a specialized hardware design [7]. In Han et al.'s method, network parameters whose absolute values are below given thresholds are directly removed from the network layer-by-layer first, then the re-training is adopted to recover pruned network accuracy. Guo et al. [5] further present a more efficient method that can enable the recovery of wrongly pruned connections appeared in [8], yielding better compression performance. In the latest works [17] [11] [25], the authors propose to either trim convolutional filters or prune feature map channels.

Low-Bit Network Quantization. Most of the computational cost of a DNN model, especially during inference phase, comes from the intensive multiplication operations of the 32-bit floating-point weights and inputs. Naturally, network quantization aims to eliminate the need of these dominant multiplications by replacing any full-precision DNN model with a low-precision version. In [36] [6], the authors use 8-bit fixed-point or 16-bit fixed-point representation to substitute 32-bit floating-point representation. Expectation BackPropagation (EBP) [33] constrains the network weights to $+1$ and -1 during feed-forward test in a probabilistic way. BinaryConnect [3] extends the idea behind EBP to directly binarize network weights during training phase. By keeping a copy of the full-precision network weights and taking it as the reference in weight binarization, BinaryConnect achieves competitive accuracy on small datasets (e.g., MNIST and CIFAR-10) using shallow CNNs. From then on, a lot of methods [4] [23] [28] [45] [44] [19] are proposed for training binary or ternary DNNs, but most of them are based on a straightforward approximation of either layer-wise network weights or layer-wise network outputs through minimizing the error of the layer-wise weights or inner products of the weights and the inputs between the original and respective quantized models. Our method differs with them both in the optimization formulation and quantization strategy.

3. The Proposed Method

In this section, we give a detailed view of our Explicit Loss-error-aware Quantization (ELQ), show how to formulate the optimization, how to bridge our basic quantization algorithm with an incremental strategy, and how to train very low-bit DNNs from the full-precision reference models with our ELQ.

3.1. Very Low-Bit Neural Networks

We begin with the notations in this paper. Denote $M = \{(W_l, X_l)|1 \leq l \leq L\}$ as a full-precision (i.e., 32-bit floating-point) DNN model, where W_l is the weight set of the l^{th} layer, X_l is the input set of the l^{th} layer, and L is the number of layers in the DNN model M. To simplify the explanation, here we only consider convolutional layers and fully connected layers of CNNs, and we also omit the dimension difference in the expression. Usually, for the convolutional layers, W_l is a $4D$ tensor and X_l is a $3D$ tensor. For the fully connected layers, W_l is a $2D$ matrix and X_l is either a $2D$ matrix (obtained by reshaping a $3D$ tensor) or a $1D$ vector. In this paper, we aim to constrain DNN model M to only have very low-bit weight set $\widehat{W_l}$ whose entries are composed of $Q_l = \{\alpha_l c_k|1 \leq k \leq K\}$. Here for the l^{th} layer, α_l is a corresponding positive scaling factor that needs to be determined during training, c_k is an integer value, and K is the number of the quantized weight centers. Specifically, for binary networks, $c_k \in \{-1, 1\}$, while for ternary networks, $c_k \in \{-1, 0, 1\}$.

Numerous methods, including but not limited to [4] [23] [28] [45] [44] [19] [15] [14], are proposed to train binary or ternary neural networks. Basically, most of them can be grouped into two solution families. The first solution family directly approximates full-precision weight sets with binary or ternary versions in a layer-by-layer manner, as defined below

$$\begin{aligned} \min_{\widehat{W_l}} \quad & ||W_l - \widehat{W_l}||^2 \\ \text{s.t.} \quad & \widehat{W_l} \in \{\alpha_l c_k|1 \leq k \leq K\},\ 1 \leq l \leq L, \end{aligned} \tag{1}$$

BinaryConnect [3] and BinaryNet [4] apply stochastic binarization functions to transform trained full-precision weights into binary equivalents. Inspired by them, Zhu et al. [45] use similar probabilistic trinization functions as the bounds to clip full-precision weights into ternary values. The other solution family considers the approximation of the inner products of the layer-wise weight sets and input sets, such as XNOR-Net [28] and Ternary Weight Networks (TWNs) [23], as defined below

$$\begin{aligned} \min_{\widehat{W_l}} \quad & ||W_l X_l - \widehat{W_l} X_l||^2 \\ \text{s.t.} \quad & \widehat{W_l} \in \{\alpha_l c_k|1 \leq k \leq K\},\ 1 \leq l \leq L. \end{aligned} \tag{2}$$

From an optimization perspective, aforementioned approximation based DNN quantization methods have two main drawbacks. First, regularizing respective approximation error will bring noticeable perturbation on the classification loss which would impact the predication accuracy of the quantized DNN model. However, loss perturbation has not been considered in these methods because they usually assume that the derivatives of the loss function w.r.t. the full-precision and quantized weights are exactly the same. Second, for these methods, there still exist noticeable accuracy gaps between the full-precision and very low-bit models.

3.2. Explicit Loss-Error-Aware Quantization

One of our goals is to encode the quantization effect to the loss function into the process of training very low-bit DNNs. To the best of our knowledge, only two attempts using the same optimization framework are available. Hou et al. [15] propose a proximal Newton algorithm based quantization method that directly minimizes the loss w.r.t. the binarized weights. In [14], they further extend this idea to handle ternary quantization task. Their optimization problem is defined as

$$\begin{aligned} \min_{\widehat{W_l}} \quad & L(\widehat{W_l}) \\ \text{s.t.} \quad & \widehat{W_l} \in \{\alpha_l c_k|1 \leq k \leq K\},\ 1 \leq l \leq L. \end{aligned} \tag{3}$$

Here, the loss function L w.r.t. the quantized weights at iteration t is defined as

$$\begin{aligned} L(\widehat{W}_l^{(t)}) = L(\widehat{W}_l^{(t-1)}) + J^{(t-1)}(\widehat{W}_l^{(t)} - \widehat{W}_l^{(t-1)}) + \\ 1/2(\widehat{W}_l^{(t)} - \widehat{W}_l^{(t-1)})^{\mathrm{T}} H^{(t-1)}(\widehat{W}_l^{(t)} - \widehat{W}_l^{(t-1)}), \end{aligned} \tag{4}$$

where $J^{(t-1)}$ and $H^{(t-1)}$ are the Jocobian and Hessian matrices w.r.t. the quantized weights at iteration $t-1$. With proximal Newton algorithm, a close-form solution can be derived (details on theoretical proofs are referred to [15] and [14]). However, proximal Newton algorithm needs to estimate the Hessian matrix of the loss function w.r.t. the quantized weights and the inputs, bringing unacceptable computational complexity which prohibits its using in the training with a large-scale dataset such as ImageNet. Besides, the loss difference between the quantized model and the full-precision counterpart is not well considered.

Unlike the optimization methods analyzed above, in our ELQ, the optimization problem is defined as

$$\begin{aligned} \min_{\widehat{W_l}} \quad & L(W_l) + a_1 L_p(W_l, \widehat{W_l}) + a_2 E(W_l, \widehat{W_l}) \\ \text{s.t.} \quad & \widehat{W_l} \in \{\alpha_l c_k|1 \leq k \leq K\},\ 1 \leq l \leq L. \end{aligned} \tag{5}$$

Here, L is the basic loss function w.r.t. the original full-precision model (It shall be noticed that this is different from the existing methods that only consider the loss function w.r.t. the quantized model at feed-forward stage during training. Retaining the loss function w.r.t. the full-precision model is critical for our ELQ, which will be clarified in the following paragraphs), L_p encodes the loss difference between the quantized and full-precision models, E represents the approximation error between the quantized weight sets and the full-precision counterparts, and a_1 and a_2 are two positive coefficients balancing the regularization. Specifically, $E = ||W_l - \widehat{W_l}||^2$, thus we only need to determine L_p. Here, we define it as

$$L_p(W_l, \widehat{W_l}) = |L(W_l) - L(\widehat{W_l})|. \tag{6}$$

Supposing that W_l is quantized into $\widehat{W_l}$ where $\widehat{W_l} \in \{\alpha_l c_k | 1 \leq k \leq K\}$, we can denote the absolute approximation difference as

$$\delta = |W_l - \widehat{W_l}|. \tag{7}$$

Now we consider first order Taylor expansion of the loss function perturbation L_p by flattening $L(\widehat{W_l})$ w.r.t. W_l, and we can derive

$$\begin{aligned} L_p(W_l, \widehat{W_l}) &= |L(W_l) - L(W_l) - \frac{\partial L}{\partial (W_l)}(\widehat{W_l} - W_l)| \\ &= |\frac{\partial L}{\partial (W_l)}(W_l - \widehat{W_l})| \\ &= |\frac{\partial L}{\partial (W_l)}|\delta. \end{aligned} \tag{8}$$

Akin to [38], for easy implementation, we also use a linear assumption $\frac{\partial L}{\partial (W_l)} \propto \delta$, then the loss difference term L_p and the approximation error term E can be easily reshaped into a uniform expression. Accordingly, for the optimization problem defined in (5), we can derive following weight update scheme

$$W_l^t = W_l^{t-1} - \gamma \frac{\partial L}{\partial (W_l^{t-1})} - \lambda sign(W_l^{t-1} - \widehat{W_l}^{t-1}), \tag{9}$$

where γ is a positive learning rate, λ is a positive coefficient, and $sign(x)$ is the sign function. Now, the difference between the optimization formulations of our ELQ and existing methods is apparent. Existing methods assume $L(W_l) == L(\widehat{W_l})$ as can be easily seen from the optimization (1), (2) and (3). In our formulation defined as (5), besides the weight approximation error (i.e., the third term), we use $L(W_l)$ instead of $L(\widehat{W_l})$ as the first term to emphasize their difference and use the loss difference between the quantized and full-precision models as the second term to encode the loss perturbation from the weight quantization, this is critical to obtain Equation (9) for weight update. On the one hand, the full-precision version of network weights is retained during training and updated at backward propagation stage. From this perspective, W_l is the variable to be optimized. On the other hand, the updated weights are quantized at feed-forward stage. From this perspective, $\widehat{W_l}$ is the final variable to be optimized.

After the weight sets are updated at iteration t, the respective ternary or binary equivalents can be obtained if the corresponding optimal scaling factor set $\{\alpha_l\}$ can be determined. Specifically, α_l is computed as

$$\alpha_l = mean(W_l) + \beta max(W_l), \tag{10}$$

where β is a positive coefficient, and we empirically set $\beta = 0.05$ in this paper. Now, for the ternary quantization of a weight set W_l,

$$\widehat{W_l} = \begin{cases} \alpha_l & \text{if } W_l > 0.5\alpha_l \\ -\alpha_l & \text{if } W_l < -0.5\alpha_l \\ 0 & \text{otherwise,} \end{cases} \tag{11}$$

and the binary quantization of a weight set W_l can be directly determined by taking the sign of the full-precision weight values

$$\widehat{W_l} = \begin{cases} \alpha_l & \text{if } W_l \geq 0 \\ -\alpha_l & \text{otherwise.} \end{cases} \tag{12}$$

3.3. Association of ELQ with Incremental Strategy

Since our ELQ jointly considers the weight approximation error and the accompanying quantization impact to the loss function for very low-bit DNN quantization problem, it has advantages in suppressing potential accuracy loss of the quantized DNN models in comparison to popular quantization methods. We can directly apply ELQ to handle ternary or binary quantization task. However, similar to state-of-the-art methods [4] [23] [28] [45] [15] [14] [44] [19], if we also adopt popular global strategy for ELQ in the quantization process, there may still exist accuracy gaps for achieving lossless quantization since simultaneously transforming all full-precision network weights into ternary or binary equivalents leaves little room to recover model accuracy. To address this problem, we combine our ELQ with a new extension of the incremental strategy [43] proposed recently. Its basic idea is to first split the weights of each layer of a DNN model into two disjoint groups, then the weights in one group are directly quantized and fixed, and the weights of the other group retaining 32-bit floating-point values are re-trained to compensate for model accuracy loss resulted from the quantization. The operations of weight partition, group-wise quantization and re-training are repeated until all network weights are quantized. In what follows, we first

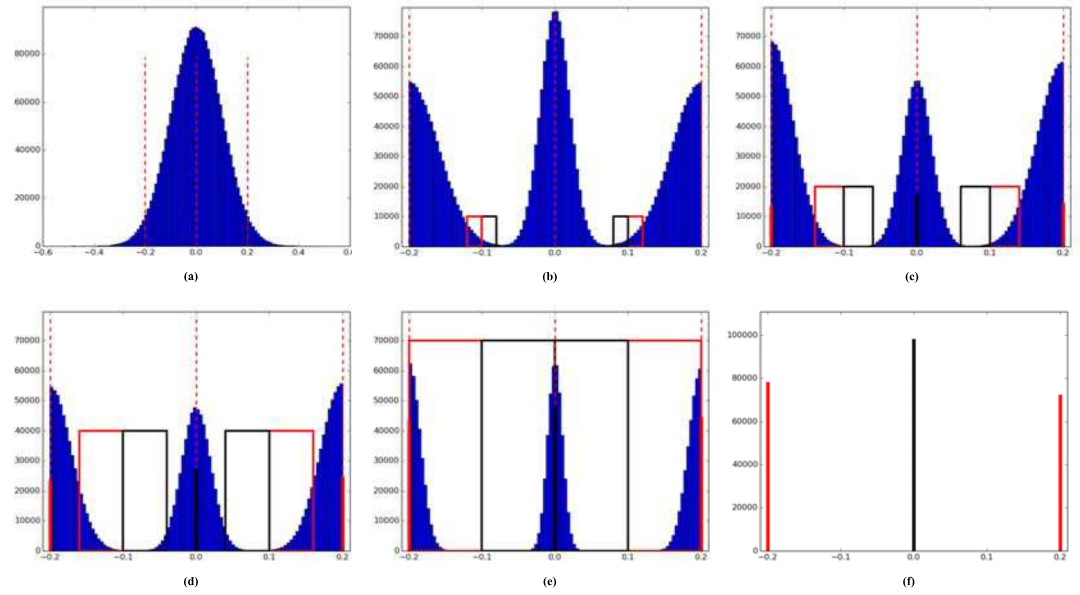

Figure 1: Illustration of the quantization procedure of our ELQ with the incremental strategy, taking ternary quantization task as an example and setting interval bound factors at successive partition steps as $\{\sigma_1 = 0.5, \sigma_2 = 0.4, \sigma_3 = 0.3, \sigma_4 = 0.2, \sigma_5 = 0.15, \sigma_6 = 0.1, \sigma_7 = 0.05, \sigma_8 = 0\}$. (a) Step 1: determine layer-wise 3 quantized centers $\{\alpha_l c_k | 1 \leq k \leq 3\}$ by applying Equation (10) over respective weight distribution, and perform weight clip and re-training. The resulting centers of $\{-0.2, 0, 0.2\}$ are indicated with 3 red dashed lines. (b) Step 2: calculate two pairs of $(\sigma_1 \alpha_l c_1, \sigma_2 \alpha_l c_1)$ and $(\sigma_1 \alpha_l c_3, \sigma_2 \alpha_l c_3)$ first, and then the weights that fall into the intervals of $[(2\sigma_1 - \sigma_2)\alpha_l c_1, \sigma_1 \alpha_l c_1)$, $[\sigma_1 \alpha_l c_1, \sigma_2 \alpha_l c_1) \cup [\sigma_2 \alpha_l c_3, \sigma_1 \alpha_l c_3)$, $[\sigma_1 \alpha_l c_3, (2\sigma_1 - \sigma_2)\alpha_l c_3)$ need to be quantized into 3 respective center values. Here, $\sigma_1 = 0.5$, $\sigma_2 = 0.4$, $\alpha_l = 0.2$, $c_1 = -1$ and $c_3 = 1$, thus the weights fall into the ranges of $[-0.12, -0.1)$, $[-0.1, -0.08) \cup [0.08, 0.1)$, $[0.1, 0.12)$ (shown as black and red rectangles) are quantized into $\{-0.2, 0, 0.2\}$ respectively and fixed, and the training is only performed on the remained full-precision weights with value clips. (c) Step 3: let $\sigma_2 = \sigma_3$, repeat the operations described in (b). (d), (e) and (f) show some successive steps until all network weights are quantized, reaching final convergence. In each figure, the horizontal axis denotes weight values, and the vertical axis denotes the accumulated number of weights with the same values, and only one particular network layer is considered for illustration.

show how to bridge our ELQ with this incremental quantization strategy, and then illustrate how to perform weight partition and train very low-bit DNNs.

Let T_l be a binary matrix having the same dimension to W_l, W_a be the weight group that needs to be quantized, W_b be the weight group that needs to be re-trained, we have

$$W_a \cup W_b = W_l, \text{and } W_a \cap W_b = \emptyset. \tag{13}$$

Now, for the l^{th} layer, weight partition can be defined as

$$T_l = \begin{cases} 0 & \text{if } W_l \odot T_l \in W_a \\ 1 & \text{if } W_l \odot T_l \in W_b, \end{cases} \tag{14}$$

where $\odot$ denotes the Hadamard product operator. By combing Equation (14) and Equation (9), we can obtain the weight update scheme for our ELQ as

$$W_l^t = W_l^{t-1} - \gamma \frac{\partial L}{\partial (W_l^{t-1})} \odot T_l - \lambda sign(W_l^{t-1} - \widehat{W_l}^{t-1}) \odot T_l. \tag{15}$$

Here, it can be seen that binary matrix T_l forces quantized weights to be fixed. In other words, only weights still have 32-bit floating-point values are re-trained to enhance network model accuracy. Now, a critical problem is how to perform weight partition. [43] proposes a magnitude-based weight partition method in which weights with larger magnitudes are grouped into the set to be quantized, while the other weights are considered to be re-trained. However, this magnitude-based weight partition method is somewhat naive, and the algorithm in [43] is also empirically studied, lacking theoretical analysis. In our ELQ, we introduce a new weight partition strategy. Considering that in ternary or binary DNN quantization task, the problem becomes easier to get a better solution if the weights are trained to be close to a number of target weight centers (i.e., $\{\alpha_l c_k | 1 \leq k \leq K\}$), in comparison to the case that network weights are scattered with a flat distribution curve. Inspired by this, for each layer, we define an interval bound factor set $\{\sigma_n | 1 \leq n \leq N\}$ where $0 \leq \sigma_n \leq 1$, guiding succes-

Algorithm 1 Explicit Loss-error-aware Quantization for training a ternary or binary DNN.

```
Require: $X$: the training data, $M = \{W_l : 1 \leq l \leq L\}$: the full-precision DNN model
Ensure: $\widehat{M} = \{\widehat{W_l} : 1 \leq l \leq L\}$: the final low-precision model with weight set $W_l$ constrained to be ternary set $\{-\alpha_l, 0, \alpha_l\}$ or binary set $\{-\alpha_l, \alpha_l\}$
 1: for $l = 1, 2, \dots, L$ do
 2:     Initialize $W_a \leftarrow \emptyset$, $W_b \leftarrow W_l$, $T_l \leftarrow 1$
 3:     Calculate ternary or binary scaling factor $\alpha_l$ by Equation (10)
 4:     Set interval bound factors at successive partition steps as $\{\sigma_1 = a, \sigma_2 = b, \cdots, \sigma_N = 0\}$
 5:     for $n = 1, 2, \dots, N$ do
 6:         Reset the base learning rate and the learning policy
 7:         Start optimization and quantization
 8:         Optimize the neural network w.r.t. the loss function
 9:         Quantize weights into ternary or binary equivalents by Equation (11) or Equation (12)
10:         Calculate feed-forward loss w.r.t. the current model
11:         Update weights by Equation (15)
12:     end for
13: end for
```

sive weight partition, quantization and re-training steps. An illustration of our method for ternary DNN quantization is shown in Figure 1. Algorithm 1 summarizes the procedure of our ELQ to train a ternary or binary DNN for approximating the full-precision reference model.

4. Experiments

In this section, we evaluate our method by conducting extensive CNN quantization experiments on two well-known image classification benchmarks, and compare our quantized model accuracies with the latest state-of-the-art results reported from XNOR-Net [28], DoReFa-Net [44], Ternary Weight Networks (TWNs) [23], Trained Ternary Quantization (TTQ) [45], Incremental Network Quantization (INQ) [43] and the BinaryConnect-ADAM (BC-ADAM) based quantization method [24]. For fair comparison and easy reproduction, we implement our ELQ algorithm by Caffe package [20] with standard CNN model files, and all experiments are conducted with the exactly same settings including data augmentation, network architecture configuration, bit-width used for quantization and so forth in comparison to [28] [44] [23] [45] [43] [24].

Model	Top-1 error	Decrease in top-1 error	Top-5 error	Decrease in top-5 error
Ref[45]	42.80		19.70	
TWNs[23]	45.50	-2.7	23.20	-3.5
TTQ[45]	42.50	0.3	20.30	-0.6
Our Ref	42.76		19.77	
Our ELQ	42.12	**0.64**	19.78	**-0.01**

Table 1: Accuracy (%) comparison of the ternary AlexNet models trained on the ImageNet classification dataset.

Model	Top-1 error	Decrease in top-1 error	Top-5 error	Decrease in top-5 error
Ref[44]	44.10		NA	
DoReFa-Net[44]	47.00	-2.9	NA	NA
Ref[28]	43.40		19.80	
XNOR-Net[28]	43.20	0.2	20.60	-0.8
Our Ref	42.76		19.77	
Our ELQ	43.05	**-0.29**	20.23	**-0.46**

Table 2: Accuracy (%) comparison of the binary AlexNet models trained on the ImageNet classification dataset.

4.1. Experiments on ImageNet

With ELQ, our main experiments are performed to train both ternary and binary CNNs for ImageNet classification task [30]. So far, in the deep learning community, ImageNet is known as the most famous image classification benchmark. It has about 1.2 million training images and 50 thousand validation images. The images in the dataset are natural images, and each image is annotated as one of 1000 object classes. We apply our ELQ to AlexNet [22] and ResNet-18 [10], covering both non-fully convolutional and fully convolutional CNN architecture families. In the experiments, all images are resized to have 256×256 pixels, and then the random crops of 224×224 pixels are used for training. At feed-forward testing phase, only 224×224 center crops of validation images are used. We report our results with two standard measures: Top-1 error rate and Top-5 error rate.

First, we consider AlexNet which has 5 convolutional layers and 3 fully connected layers. Following TWNs [23], DoReFa-Net [44], XNOR-Net [28] and TTQ [45], we also remove dropout operations and add batch normalization operations to original AlexNet architecture [22]. Taking full-precision AlexNet model as the reference, we separately train corresponding ternary and binary models with our ELQ using the same experimental settings. Specifically, we run our ELQ training algorithm with SGD for 80 epochs with the batch size of 256, the weights decay of 0.0003 and the momentum of 0.9. At each re-training step, the learning rate starts at 0.001 and is divided by 2 every 2 epochs. For weight partition in our ELQ, we set interval bound factors at successive partition steps as $\{\sigma_1 = 0.5, \sigma_2 = 0.4, \sigma_3 = 0.3, \sigma_4 = 0.2, \sigma_5 = 0.15, \sigma_6 = 0.1, \sigma_7 = 0.05, \sigma_8 = 0\}$. Table 1 compares the ternary AlexNet model accuracies of our method with so far best reported results. First, we can find that the accuracy difference between our full-precision AlexNet reference and the comparative reference is very small (specifically, 0.04% difference in Top-1 error rate and 0.07% difference in Top-5 error rate). Second, it can be seen that our ternary AlexNet model reaches a Top-1 error rate of 42.12% which outperforms its full-precision reference by 0.64%, and its Top-5 accuracy is almost the same as the reference. Comparatively, TWNs [23] brings serious accuracy drops: 2.7% to Top-1 accuracy and 3.5% to Top-5 accuracy. TTQ [45] exhibits much better accuracy than TWNs, but it still introduces 0.6% decrease in Top-5 accuracy. Generally, our ternary model achieves the best accuracies with obvious margins over all compared methods. Table 2 further compares the binary AlexNet model accuracies of our method with DoReFa-Net [44] and XNOR-Net [28]. Again, we can see that our full-precision reference model has slightly better accuracy than those used in DoReFa-Net and XNOR-Net. Comparatively, for binary AlexNet models, DoReFa-Net shows noticeable accuracy drop: 2.9% to Top-1 accuracy. Although XNOR-Net achieves 0.2% gain to Top-1 accuracy, it brings 0.8% drop in Top-5 accuracy. Generally, our binary model has more consistent accuracy in comparison to the reference model. Furthermore, both our ternary and binary AlexNet models show the best accuracies compared with the state-of-the-art results.

Next, we consider full convolutional ResNet-18 that is popularly used in DNN quantization tasks. ResNets [10] have batch normalization layers and relief the vanishing gradient problem by using shortcut connections. Taking full-precision ResNet-18 model as the reference, we separately train corresponding ternary and binary models with our ELQ using the same experimental settings. Specifically, we run our ELQ training algorithm with SGD for 96 epochs with the batch size of 80, the weights decay of 0.0003 and the momentum of 0.9. At each re-training step, the learning rate starts at 0.001 and is divided by 2 every 2 epochs. For weight partition in our ELQ, we set interval bound factors at successive partition steps as $\{\sigma_1 = 0.5, \sigma_2 = 0.4, \sigma_3 = 0.3, \sigma_4 = 0.2, \sigma_5 = 0.15, \sigma_6 = 0.1, \sigma_7 = 0.05, \sigma_8 = 0\}$. Accuracy comparison of ternary ResNet-18 models is summarized in Table 3. Partially owing to the increased depth and the significantly decreased number of parameters, unlike the results on AlexNet, the ternary ResNet-18 models from all methods have different level accuracy losses when compared with the corresponding full-precision references. However, only our method shows minor accuracy

Model	Top-1 error	Decrease in top-1 error	Top-5 error	Decrease in top-5 error
Ref[45]	30.4		10.80	
TWNs[23]	34.70	-4.3	13.80	-3.0
TTQ[45]	33.40	-3.0	12.80	-2.0
Our Ref	31.73		11.31	
Our ELQ	32.48	**-0.75**	11.95	**-0.64**

Table 3: Accuracy (%) comparison of the ternary ResNet-18 models trained on the ImageNet classification dataset.

Model	Top-1 error	Decrease in top-1 error	Top-5 error	Decrease in top-5 error
Ref[28]	30.70		10.80	
XNOR-Net[28]	39.20	-8.5	17.00	-6.2
Our Ref	31.73		11.31	
Our ELQ	35.28	**-3.55**	13.96	**-2.65**

Table 4: Accuracy (%) comparison of the binary ResNet-18 models trained on the ImageNet classification dataset.

drops: 0.75% to Top-1 accuracy and 0.64% to Top-5 accuracy, which are much lower than the accuracy drops reported for TWNs [23] and TTQ [45]. Even for the latest ternary ResNet-18 model distilled by a much deeper and more powerful ResNet-152 model [26], it still has -1.4% decrease in Top-1 error rate (31.8% vs 30.4%). Accuracy comparison of binary ResNet-18 models is summarized in Table 4. Compared with ternary ResNet-18 models, as the bit width goes from 2 bits to 1 bit, binary ResNet-18 models show much larger accuracy drops. The binary ResNet-18 model from XNOR-Net [28] has 8.5% drop to Top-1 accuracy and 6.2% drop to Top-5 accuracy, while our binary ResNet-18 model outperforms it with significant margins.

4.2. Quantization with Increased Network Depth

We also perform ablation study on the CIFAR-10 dataset [21] to explore the quantization performance of our ELQ with increased network depth. CIFAR-10 is an image classification dataset which has 60000 32×32 color images grouped into 10 object categories. We present experiments trained on the training set of 50000 images and evaluated on the test set of 10000 images. Following [23] [45] [24], we also consider ResNet-20 and ResNet-56, and use the same data augmentation for training. Here, we intentionally use the same parameter settings to train very low-bit ResNet-20 and ResNet-56 separately. Specifically, we run our ELQ training algorithm with SGD for 150 epochs with the batch size of 256, the weights decay of 0.0003 and the momentum of 0.9. At each re-training step, the learning rate starts

Model	Top-1 error	Decrease in top-1 error
Ref[45]	8.23	
TWNs (Ternary)[23]	9.12	-0.89
TTQ (Ternary)[45]	8.87	-0.64
Our Ref	8.75	
Our ELQ (Ternary)	8.55	**0.2**
Our ELQ (Binary)	8.85	**-0.1**

Table 5: Accuracy (%) comparison of the ternary/binary ResNet-20 models trained on the CIFAR-10 dataset.

Model	Top-1 error	Decrease in top-1 error
Ref[45]	6.80	
TTQ (Ternary)[45]	6.44	0.36
Ref[24]	8.10	
BC-ADAM (Binary)[24]	8.83	-0.73
Our Ref	6.97	
Our ELQ (Ternary)	6.30	**0.67**
Our ELQ (Binary)	7.18	**-0.21**

Table 6: Accuracy (%) comparison of the ternary/binary ResNet-56 models trained on the CIFAR-10 dataset.

at 0.001 and is divided by 2 every 3 epochs. For weight partition in our ELQ, we set interval bound factors at successive partition steps as $\{\sigma_1 = 0.5, \sigma_2 = 0.4, \sigma_3 = 0.3, \sigma_4 = 0.2, \sigma_5 = 0.15, \sigma_6 = 0.1, \sigma_7 = 0.05, \sigma_8 = 0\}$.

For ResNet-20, we first train a ternary model, and then we train a binary version as well. Accuracy comparison is presented in Table 5. Similar to the results on the ImageNet classification dataset, our method gets the best ternary ResNet-20 model when compared with TWNs [23] and TTQ [45]. For CIFAR-10 task, our ternary model is even better than the full-precision reference, and our binary model also shows slightly improved accuracy than the ternary models from TWNs and TTQ. It only brings 0.1% drop in Top-1 accuracy in comparison to the full-precision reference. With much deeper ResNet-56, we further explore ternary and binary quantization performance of our ELQ. According to the results shown in Table 6, we can find that our ELQ obtains much better results than the latest BC-ADAM method [24]. Even though our ternary ResNet-56 model has 0.88% accuracy improvement over our binary model, the accuracy difference between our binary model and the full-precision reference is only 0.21%. Above experiments clearly show that our ELQ can well address ternary and binary quantization of DNNs with increased network depth.

4.3. Comparison of Weight Partition Strategies

Recall that our ELQ uses an incremental quantization implementation based on a new weight partition strategy. Original magnitude-based weight partition method proposed in [43] is an empirical solution, lacking theoretical analysis. Although our weight partition strategy is different from this magnitude-based partition, it is necessary to compare them. To this end, we perform an experiment on the ImageNet classification dataset. In the experiment, we also consider ternary quantization task for ResNet-18, as in [43]. The parameter settings for the batch size, the weight decay, the momentum and so forth are exactly the same. The result summary is given in Table 7. It can be seen that our model accuracy is consistently better than the results from INQ, yielding the improvements of 1.5% and 0.92% w.r.t. Top-1 and Top-5 accuracy. Although INQ is inferior to our method, its model accuracy is much better than TWNs [23] and TTQ [45], demonstrating the advantage of the way to perform network quantization incrementally.

Model	Top-1 error	Decrease in top-1 error	Top-5 error	Decrease in top-5 error
Ref	31.73		11.31	
INQ[43]	33.98	-2.25	12.87	-1.56
Our ELQ	32.48	**-0.75**	11.95	**-0.64**

Table 7: Accuracy (%) comparison of the ternary ResNet-18 models trained on the ImageNet classification dataset.

5. Conclusion

This paper presents ELQ, a new quantization method, for training very low-bit DNNs with negligible loss of predication accuracy compared with the full-precision counterparts. Taking a full-precision DNN as the reference, existing methods usually pose the optimization problem as the approximation of either layer-wise weights or layer-wise inner products of the weights and the respective inputs, without considering the quantization effect to the loss function. Our ELQ jointly considers the loss perturbation and the weight approximation error in the optimization, and we further bridge it with an incremental quantization strategy for lossless DNN quantization. One appealing merit of our ELQ is that the negative quantization impact on the predication accuracy can be well suppressed. Moreover, it avoids the drawbacks of second order optimization methods [15] [14], such as the expensive computational cost of the Jacobian and Hessian matrices when training very low-bit DNNs with large scale datasets. We have shown that our approach is theoretically reasonable and practically effective. As validated with two mainstream convolutional neural network families (i.e., fully convolutional and non-fully convolutional), our ELQ shows promising binary and ternary quantization results on the large scale ImageNet classification dataset. As for future works, we may extend the idea behind ELQ to train very low-bit DNNs from scratch.

References

[1] F. Chollet. Xception: Deep learning with depthwise separable convolutions. In *CVPR*, 2017.

[2] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 2011.

[3] M. Courbariaux, Y. Bengio, and J.-P. David. Binaryconnect: Training deep neural networks with binary weights during propagations. In *NIPS*, 2015.

[4] M. Courbariaux, I. Hubara, D. Soudry, R. El-Yaniv, and Y. Bengio. Binarized neural networks: Training neural networks with weights and activations constrained to +1 or -1. In *NIPS*, 2016.

[5] Y. Guo, A. Yao, and Y. Chen. Dynamic network surgery for efficient dnns. In *NIPS*, 2016.

[6] S. Gupta, A. Agrawal, and K. Gopalakrishnan. Deep learning with limited numerical precision. In *ICML*, 2015.

[7] S. Han, X. Liu, H. Mao, J. Pu, A. Pedram, M. A. Horowitz, and W. J. Dally. Eie: Efficient inference engine on compressed deep neural network. In *ISCA*, 2016.

[8] S. Han, J. Pool, J. Tran, and W. J. Dally. Learning both weights and connections for efficient neural networks. In *NIPS*, 2015.

[9] B. Hassibi and D. G. Stork. Second order derivatives for network pruning: Optimal brain surgeon. In *NIPS*, 1993.

[10] K. He, X. Zhang, S. Ren, and S. Jian. Deep residual learning for image recognition. In *CVPR*, 2016.

[11] Y. He, X. Zhang, and J. Sun. Channel pruning for accelerating very deep neural networks. In *ICCV*, 2017.

[12] G. Hinton, L. Deng, D. Yu, G. Dahl, A.-r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. Sainath, and B. Kingsbury. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Processing Magazine*, 2012.

[13] G. Hinton, O. Vinyals, and J. Dean. Distilling the knowledge in a neural network. In *NIPS*, 2014.

[14] L. Hou and J. T. Kwok. Loss-aware weight quantization of deep networks. *ICLR*, 2018.

[15] L. Hou, Q. Yao, and J. T. Kwok. Loss-aware binarization of deep networks. In *ICLR*, 2017.

[16] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv:1704.04861v1*, 2017.

[17] H. Hu, R. Peng, Y.-W. Tai, and C.-K. Tang. A data-driven neuron pruning approach towards efficient deep architectures. *arXiv:1607.03250v1*, 2016.

[18] G. Huang, Z. Liu, and K. Q. Weinberger. Densely connected convolutional networks. In *CVPR*, 2017.

[19] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio. Quantized neural networks: Training neural networks with low precision weights and activations. *arXiv:1609.07061v1*, 2016.

[20] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. In *ACM MM*, 2014.

[21] A. Krizhevsky and G. Hinton. Learning multiple layers of features from tiny images. 2009.

[22] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012.

[23] F. Li and B. Liu. Ternary weight networks. *arXiv:1605.04711v1*, 2016.

[24] H. Li, S. De, Z. Xu, C. Studer, H. Samet, and T. Goldstein. Training quantized nets: A deeper understanding. In *NIPS*, 2017.

[25] H. Li, A. Kadav, I. Durdanovic, H. Samet, and H. P. Graf. Pruning filters for efficient convnets. In *ICLR*, 2017.

[26] A. Mishra and D. Marr. Apprentice: Using knowledge distillation techniques to improve low-precision network accuracy. *ICLR*, 2018.

[27] A. Polino, R. Pascanu, and D. Alistarh. Model compression via distillation and quantization. *ICLR*, 2018.

[28] M. Rastegari, V. Ordonez, J. Redmon, and A. Joseph. Xnor-net: Imagenet classification using binary convolutional neural networks. In *ECCV*, 2016.

[29] A. Romero, N. Ballas, S. E. Kahou, A. Chassang, C. Gatta, and Y. Bengio. Fitnets: Hints for thin deep nets. In *ICLR*, 2015.

[30] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.

[31] D. Silver, A. Huang, C. Maddison, A. Guez, L. Sifre, G. v. d. Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis. Mastering the game of go with deep neural networks and tree search. *Nature*, 2016.

[32] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015.

[33] D. Soudry, I. Hubara, and R. Meir. Expectation backpropagation: Parameter-free training of multilayer neural networks with continuous or discrete weights. In *NIPS*, 2014.

[34] V. Sze, Y.-H. Chen, T.-J. Yang, and J. Emer. Efficient processing of deep neural networks:a tutorial and survey. *arXiv:1703.09039v2*, 2017.

[35] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *CVPR*, 2015.

[36] V. Vanhoucke, A. Senior, and M. Z. Mao. Improving the speed of neural networks on cpus. In *NIPS Workshop*, 2011.

[37] S. Xie, R. Girshick, P. Dollar, Z. Tu, and K. He. Aggregated residual transformations for deep neural networks. In *CVPR*, 2017.

[38] X. Xiong and F. D. l. Torre. Supervised descent method and its applications to face alignment. In *CVPR*, 2013.

[39] L. Yann, J. S. Denker, I. Solla, sara A., and G. E. Hinton. Optimal brain damage. In *NIPS*, 1990.

[40] J. Yim, D. Joo, J. Bae, and J. Kim. A gift from knowledge distillation: Fast optimization, network minimization and transfer learning. In *CVPR*, 2017.

[41] S. Zagoruyko and N. Komodakis. Wide residual networks. In *BMVC*, 2016.

[42] X. Zhang, X. Zhou, M. Lin, and J. Sun. Shufflenet: An extremely efficient convolutional neural network for mobile devices. *arXiv:1707.01083v1*, 2017.

[43] A. Zhou, A. Yao, Y. Guo, L. Xu, and Y. Chen. Incremental network quantization: Towards lossless cnns with low-precision weights. In *ICLR*, 2017.

[44] S. Zhou, W. Yuxin, Z. Ni, X. Zhou, H. Wen, and Y. Zou. Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients. *arXiv:1606.06160v1*, 2016.

[45] C. Zhu, S. Han, and H. Mao. Trained ternary quantization. In *ICLR*, 2017.