Least squares binary quantization of neural networks

Hadi Pouransari, Zhucheng Tu, Oncel Tuzel
Apple Inc.
Cupertino, CA 95014, USA
{mpouransari,zhucheng_tu,ctuzel}@apple.com

Abstract

Quantizing weights and activations of deep neural networks results in significant improvement in inference efficiency at the cost of lower accuracy. A source of the accuracy gap between full precision and quantized models is the quantization error. In this work, we focus on the binary quantization, in which values are mapped to -1 and 1. We provide a unified framework to analyze different scaling strategies. Inspired by the pareto-optimality of 2-bits versus 1-bit quantization, we introduce a novel 2-bits quantization with provably least squares error. Our quantization algorithms can be implemented efficiently on the hardware using bitwise operations. We present proofs to show that our proposed methods are optimal, and also provide empirical error analysis. We conduct experiments on the ImageNet dataset and show a reduced accuracy gap when using the proposed least squares quantization algorithms.[1]

1. Introduction

A major challenge in the deployment of Deep Neural Networks (DNNs) is their high computational cost. Finding effective methods to improve run-time efficiency is still an area of research. We can group various approaches taken by researchers into the following three categories.

Hardware optimization: Specifically designed hardwares are deployed to efficiently perform computations in ML tasks. **Compiler optimization**: Compression and fusion techniques coupled with efficient hardware-aware implementations, such as dense and sparse matrix-vector multiplication, are used. **Model optimization**: Run-time performance can also be gained by modifying the model structure and the underlying arithmetic operations. While hardware and compiler optimization are typically lossless, model optimization trades-off computational cost (memory, runtime, or power) for model accuracy. For example, by scaling the width of the network [53]. The goal of model optimization is to improve the trade-off between computational cost and model accuracy. This work falls into this category. We briefly explain different model optimization techniques here.

Architecture optimization One strategy to construct efficient DNNs is to define a template from which efficient computational blocks can be generated. SqueezeNet [29], MobileNets [27, 45], ShuffleNets [39, 55], and ESPNets [40, 41] fall into this category. Complementary to these methods, NASNet [58] and EfficientNet [49] search for an optimal composition of blocks restricted to a computational budget (e.g., FLOPS) by changing the resolution, depth, width, or other parameters of each layer.

Pruning and Compression: Several methods have been proposed to improve runtime performance by detecting and removing computational redundancies. Examples of methods in this category include low-rank acceleration [32], the use of depth-wise convolution in Inception [48], sparsification of kernels in deep compression [20], re-training redundant neurons in DSD [21], depth-wise separable convolution in Xception [10], pruning redundant filters in PFA [47], finding an optimal sub-network in lottery ticket hypothesis [16], and separating channels based on the features resolution in octave convolution [8].

Low-precision arithmetic and quantization: Another avenue to improve runtime performance, and the focus of this work, is to use low-precision arithmetic. The idea is to use fewer bits to represent weights and activations. Some instances of these strategies already exist in AI compilers, where it is common to cast weights of a trained model from 32 bits to 16 or 8 bits. However, in general, post-training quantization reduces the model accuracy. This can be addressed by incorporating lower-precision arithmetic into the training process (during-training quantization), allowing the resulting model to better adapt to the lower precision [18, 31]. Additionally, to improve utilization, many works considered mixed-precision quantization, where different number of bits are allowed at different layers of the network [14, 50, 52].

Using fewer bits results in dramatic memory savings.

[1] Code will be available.

This has motivated research into methods that use a single bit to represent a scalar weight: In [11] the authors train models with weights quantized to the values in $\{-1, 1\}$. While this results in a high level of compression, model accuracy can drop significantly. [35] and [57] reduce the accuracy gap between full precision and quantized models by considering ternary quantization (using the values in $\{-1, 0, 1\}$), at the cost of slightly less compression.

To further improve the computational efficiency, the intermediate activation tensors (feature maps) can also be quantized. When this is the case, an implementation can use high-performance operators that act on quantized inputs, for example a convolutional block depicted in Figure 1. This idea has been explored in [7, 12, 28, 36, 42, 43, 44, 54, 56], and many other works.

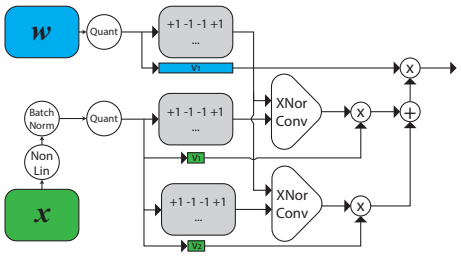

Figure 1. When both weights and activations are quantized using binary quantization, the convolution can be implemented efficiently using bitwise XNor and bit-counting operations. See Section 3.2 for more details.

We call a mapping from a tensor with full precision entries to a tensor with the same shape but with values in $\{-1, 1\}$ a **binary quantization**. When both weights and activations of a DNN are quantized using binary quantization, called Binary Neural Network (BNN), fast and power-efficient kernels which use bitwise operations can be implemented. Observe that the inner-product between two vectors with entries in $\{-1, 1\}$ can be written as bitwise XNor operations followed by bit-counting [12]. However, the quantization of both weights and activations further reduces the model accuracy [28, 44].

The accuracy drop due to quantization can be compensated by increasing model capacity through architecture modifications, for example by increasing the number of filters as in [53]. In Figure 2 we show the trade-off between computational cost (memory and flops) and classification error for different quantization schemes by uniformly scaling the number of filters in ResNet18 architecture trained on CIFAR100 dataset [34]. We followed methodology in [51] to approximate equivalent flops of BNNs. We use the same training setup for quantized models without any tuning. This empirical result suggests that for a given computational budget using 2-bits quantizations is pareto-optimal when compared to the original BNN and the full-precision network. [33] also observed much larger accuracy degradation when the bit precision is reduced from 2-bits to 1-bit than other cases with $>$2-bits. This interesting observation, although not necessarily a universal conclusion, motivates deriving the optimal 2-bits quantization.

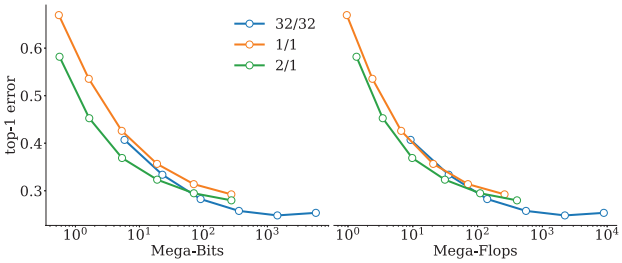

Figure 2. The trade-off between computational cost model error for ResNet18 trained on CIFAR100. Left plot show memory footprint, and right plot shows flop counts. k^a/k^w refers to using k^a bits for activations and k^w bits for weights.

Note that there are many other directions to improve the accuracy of BNNs. For example, using learned clipping in PACT [9], double skip-connection in BiRealNet [37], parametric ReLU in [6], multi-stage knowledge distillation in [2, 6], and tailored binary optimization in [24]. These are all orthogonal improvements to the proposed provably least squares error 2-bits quantization, and can be used together to further improve the model accuracy.

1.1. Main contributions

In this work, we analyze the accuracy of binary quantization when applied to both weights and activations of a DNN, and propose methods to improve the quantization accuracy:

- We present a unified framework to analyze different scaling strategies for BNNs, and show that scaled binary quantization is a good approximation (Section 2).
- We derive 2-bits (Section 3.2.2) and ternary (Section 3.2.3) scaled binary quantization algorithms with least squares error. We also propose greedy variants of these algorithms (Section 3.2.4).
- Experiments on the ImageNet dataset show that the optimal algorithms have reduced quantization error, and lead to improved classification accuracy (Section 5).

2. Low-rank binary quantization

Binary quantization (that maps entries of a tensor to $\{-1, 1\}$) of weights and activation tensors of a neural network can significantly reduce the model accuracy. A remedy to retrieve this accuracy loss is to scale the binarized tensors with few full precision values. For example, [28] learn a scaling for each channel from the parameters of

batch-normalization, and [44] scale the quantized activation tensors using the channel-wise average of pixel values.

In this section, using low-rank matrix analysis, we analyze different scaling strategies. We conclude that multiplying the quantized tensor by a single scalar, which is computationally the most efficient option, has approximately the same accuracy as the more expensive alternatives.

We introduce the rank-1 binary quantization– an approximation to a matrix $\boldsymbol{X} \in \mathbb{R}^{m\times n}$:

$$\boldsymbol{X} \simeq \boldsymbol{X}_1 \odot \boldsymbol{S}, \tag{1}$$

where $\boldsymbol{X}_1 \in \mathbb{R}^{m\times n}$ is a rank-1 matrix, $\boldsymbol{S} \in \{-1,1\}^{m\times n}$, and $\odot$ is element-wise multiplication (Hadamard product). Note that this approximation is also defined for tensors, after appropriate reshaping. For example, for an image classification task, we can reshape the output of a layer of a DNN with shape $h \times w \times n$, where h, w, and n are height, width, and number of channels, respectively, into an $m \times n$ matrix with $m = hw$ rows and one column per channel.

We define the error of a rank-1 binary quantization as $\|\boldsymbol{X} - \boldsymbol{X}_1 \odot \boldsymbol{S}\|_F$, where $\|\ \|_F$ is the Frobenius norm. Entries of $\boldsymbol{S}$ are in $\{-1,1\}$, therefore, the quantization error is equal to $\|\boldsymbol{X} \odot \boldsymbol{S} - \boldsymbol{X}_1\|_F$. Note that $\|\boldsymbol{X} \odot \boldsymbol{S}\|_F^2$ (the total energy), which is equal to sum of the squared singular values, is the same for any $\boldsymbol{S}$. Different choices of $\boldsymbol{S}$ change the distribution of the total energy among components of the Singular Value Decomposition (SVD) of $\boldsymbol{X} \odot \boldsymbol{S}$. The optimal rank-1 binary quantization is achieved when most of the energy of $\boldsymbol{X} \odot \boldsymbol{S}$ is in its first component.

In [44], the authors proposed to quantize the activations by applying the sign function and scale them by their channel-wise average. We can formulate this scaling strategy as a special rank-1 binary quantization $\boldsymbol{X} \simeq \boldsymbol{a}\mathbf{1}^\top \odot \text{ sign}(\boldsymbol{X})$, where

$$\begin{aligned} a_i &= \frac{\sum_{j=1}^{n} |\boldsymbol{X}_{i,j}|}{n} \quad \text{for} \quad 1 \le i \le m, \\ \text{sign}(x) &= \begin{cases} -1 & \text{if } x < 0 \\ 1 & \text{if } x \ge 0 \end{cases}, \end{aligned} \tag{2}$$

and $\mathbf{1}$ is an n-dimensional vector with all entries 1.

In Appendix A we show that the optimal rank-1 binary quantization of an arbitrary $\boldsymbol{X} \in \mathbb{R}^{m\times n}$ is given by $\boldsymbol{S} = \text{sign}(\boldsymbol{X})$ and $\boldsymbol{X}_1 = \text{truncated}_1\text{-SVD}(|\boldsymbol{X}|)$, where $\text{sign}(\boldsymbol{X})$ is the element-wise sign of $\boldsymbol{X}$, and $\text{truncated}_1\text{-SVD}(|\boldsymbol{X}|) = \sigma_1 \boldsymbol{u}_1 \boldsymbol{v}_1^\top$ is the first component of the SVD of $\boldsymbol{X} \odot \text{sign}(\boldsymbol{X}) = |\boldsymbol{X}|$. Moreover, if entries of $\boldsymbol{X}$ are i.i.d. $\sim \mathcal{N}(0,1)$, the first singular value of $|\boldsymbol{X}|$ captures most of the energy $\sigma_1^2(|\boldsymbol{X}|)/\|\boldsymbol{X}\|_F^2 \simeq 0.64$, and the first left and right singular vectors are almost constant vectors. Normal distribution assumption for entries of $\boldsymbol{X}$ is relevant due to application of Batch Normalization (BN) [30]. Therefore, a scalar multiple of $\text{sign}(\boldsymbol{X})$ approximates $\boldsymbol{X}$ well: $\boldsymbol{X} \simeq \sigma_1 \boldsymbol{u}_1 \boldsymbol{v}_1^\top \odot \text{sign}(\boldsymbol{X}) \simeq v\ \mathbf{1}\mathbf{1}^\top \odot \text{sign}(\boldsymbol{X}) = v\,\text{sign}(\boldsymbol{X})$, where $v \in \mathbb{R}_{\ge 0}$. We call this computationally efficient approximation **scaled binary quantization**.

3. Scaled binary quantization

In Section 2 we showed that scaled binary quantization is a good approximation to activation and weight tensors of a DNN. Next we show how we can further improve the accuracy of scaled binary quantization using more bits. To simplify the presentation (1) we flatten matrix $\boldsymbol{X} \in \mathbb{R}^{m\times n}$ in to a vector $\boldsymbol{x} \in \mathbb{R}^N$ with $N = mn$, and (2) we assume the entries of $\boldsymbol{x}$ are different realizations of a random variable x with an underlying probability distribution $p(x)$. In practice, we compute all statistics using their unbiased estimators from vector $\boldsymbol{x}$ (e.g., $\sum_i x_i/N$ is an unbiased estimator of $\mathbb{E}_{\text{x}\sim p}[\text{x}]$). Furthermore, for $f : \mathbb{R} \to \mathbb{R}$, we denote entrywise application of f to $\boldsymbol{x}$ by $f(\boldsymbol{x})$. The quantized approximation of $\boldsymbol{x}$ is denoted by $\boldsymbol{x}^q$, and the error (loss) of quantization is $\|\boldsymbol{x} - \boldsymbol{x}^q\|_2$. All optimal solutions refers to the least squares error and hold for an arbitrary distribution $p(x)$.

3.1. 1-Bit quantization

A 1-bit scaled binary quantization of $\boldsymbol{x}$ is:

$$\boldsymbol{x} \simeq \boldsymbol{x}^q = v s(\boldsymbol{x}), \tag{3}$$

which is determined by a scalar $v \in \mathbb{R}_{\ge 0}$ and a function $s : \mathbb{R} \to \{-1,1\}$. Finding the least squares 1-bit scaled binary quantization can be formulated as the following optimization problem:

$$\begin{aligned} \underset{v,s}{\text{minimize}} \quad & \int_{-\infty}^{+\infty} p(x)(v\,s(x) - x)^2 dx \\ \text{s.t.} \quad & s : \mathbb{R} \to \{-1,1\}, v \in \mathbb{R}_{\ge 0} \end{aligned} \tag{4}$$

3.1.1 Least squares 1-Bit algorithm

The solution of problem (4) is given by $v = \mathbb{E}_{\text{x}\sim p}[|\text{x}|]$ and $s(x) = \text{sign}(x)$ (for the proofs see Appendix B). Therefore, for a vector $\boldsymbol{x}$ the optimal scaled binary quantization is given by

$$\boldsymbol{x} \simeq \boldsymbol{x}^q = \frac{\sum_i |x_i|}{N} \text{sign}(\boldsymbol{x}), \tag{5}$$

where $\frac{\sum_i |x_i|}{N}$ is an unbiased estimator of $\mathbb{E}_{\text{x}\sim p}[|\text{x}|]$.

3.2. k-Bits quantization

We can further improve the accuracy of scaled binary quantization by adding more terms to the approximation (3).

A k-bits scaled binary quantization of $\boldsymbol{x}$ is

$$\boldsymbol{x} \simeq \boldsymbol{x}^q = \sum_{i=1}^{k} v_i s_i(\boldsymbol{x}), \tag{6}$$

which is determined by a set of k pairs of scalars v_i's and functions $s_i : \mathbb{R} \to \{-1, 1\}$. Observe that any permutation of (v_i, s_i)'s results in the same quantization. To remove ambiguity, we assume $v_1 \geq \ldots \geq v_k \geq 0$.

When both weights, $\boldsymbol{w}$, and activations, $\boldsymbol{x}$, are quantized using (6), their inner-product can be written as:

$$\langle \boldsymbol{x}^q, \boldsymbol{w}^q \rangle = \sum_{i=1}^{k^a} \sum_{j=1}^{k^w} v_i^a v_j^w \langle \boldsymbol{s}_i^a, \boldsymbol{s}_j^w \rangle, \tag{7}$$

where $\boldsymbol{x}^q = \sum_{i=1}^{k^a} v_i^a \boldsymbol{s}_i^a$ and $\boldsymbol{w}^q = \sum_{j=1}^{k^w} v_j^w \boldsymbol{s}_i^w$ are quantized activations and weights with k^a and k^w bits, respectively, $\boldsymbol{s}_i^a = s_i^a(\boldsymbol{x})$, and $\boldsymbol{s}_j^w = s_j^w(\boldsymbol{w})$. This inner-product can be computed efficiently using bitwise XNors followed by bit-counting (see Figure 1 with $k^a = 2$ and $k^w = 1$).

Finding the least squares k-bits scaled binary quantization can be formulated as:

$$\begin{aligned} \underset{s_i, v_i}{\text{minimize}} \quad & \int_{-\infty}^{+\infty} p(x) \left(\left(\sum_{i=1}^{k} v_i s_i(x) \right) - x \right)^2 dx \\ \text{s.t.} \quad & \forall\, 1 \leq i \leq k \quad s_i : \mathbb{R} \to \{-1, 1\}, \\ & v_1 \geq v_2 \geq \ldots \geq v_k \geq 0 \end{aligned} \tag{8}$$

This is an optimization problem with a non-convex domain for all $k \geq 1$. We solve the optimization for $k = 1$ in Section 3.1 and $k = 2$ in Section 3.2.2 for arbitrary distribution $p(x)$. We also provide an approximate solution to (8) in Section 3.2.4 using a greedy algorithm.

Discussion: A general k-bits quantizer maps full precision values to an arbitrary set of 2^k numbers, not necessarily in the form of (6). The optimal quantization in this case can be computed using the Lloyd's algorithm [38]. While a general k-bits quantization has more representation power compared to k-bits scaled binary quantization, it does not allow an efficient implementation based on bitwise operations. Fixed-point representation (as opposed to floating point) is also in the form of (6) with an additional constant term. However, fixed-point quantization uniformly quantizes the space, therefore, it can be significantly inaccurate for small values of k.

3.2.1 Foldable quantization

In this section, we introduce a special family of k-bits scaled binary quantizations that allow fast computation of the quantized values. We name this family of quantizations **foldable**. A k-bits scaled binary quantization given by (v_i, s_i)'s is foldable if the following conditions are satisfied:

$$s_i(x) = \text{sign}(x - \sum_{j=1}^{i-1} v_j s_j(x)) \quad \text{for } 1 \leq i \leq k \tag{9}$$

When the foldable condition is satisfied, given v_i's, we can compute the $s_i(\boldsymbol{x})$'s in (6) efficiently by applying the sign function.

3.2.2 Least squares 2-bits algorithm

In this section, we present the least squares 2-bits binary quantization algorithm, the solution of (8) for $k = 2$. In Appendix C we show that the least squares 2-bits binary quantization is foldable and the scalars v_1 and v_2 should satisfy the following optimality conditions:

$$v_1 = \frac{1}{2} \left(\mathbb{E}_{\text{x} \sim p}[|\text{x}| \mid |\text{x}| > v_1] + \mathbb{E}_{\text{x} \sim p}[|\text{x}| \mid |\text{x}| \leq v_1] \right) \tag{10}$$

$$v_2 = \frac{1}{2} \left(\mathbb{E}_{\text{x} \sim p}[|\text{x}| \mid |\text{x}| > v_1] - \mathbb{E}_{\text{x} \sim p}[|\text{x}| \mid |\text{x}| \leq v_1] \right) \tag{11}$$

In Figure 3 we visualize the conditional expectations that show up in (10) for a random variable x with standard normal distribution. The optimal v_1 lies on the intersection of the identity line and average of the conditional expectations in (10).

For a given vector $\boldsymbol{x} \in \mathbb{R}^N$ we can solve for v_1 in (10) efficiently. We substitute the conditional expectations in (10) by conditional average operators as their unbiased estimators. (10) implies that for the optimal v_1, the average of the entries in $|\boldsymbol{x}|$ smaller than v_1 (an estimator of $\mathbb{E}_{\text{x} \sim p}[|\text{x}| \mid |\text{x}| \leq v_1]$) and the average of the entries greater than v_1 (an estimator of $\mathbb{E}_{\text{x} \sim p}[|\text{x}| \mid |\text{x}| > v_1]$) should be equidistant form v_1. Note that (10) may have more than one solution, which are local minima of the objective function in (8). We find all the values that satisfy this condition in $\mathcal{O}(N \log N)$ time. We first sort entries of $\boldsymbol{x}$ based on their absolute value and compute their cumulative sum. Then with one pass we can check whether (10) is satisfied for each element of $\boldsymbol{x}$. We evaluate the objective function in (8) for each local minima, and retain the best. After v_1 is calculated v_2 is simply computed from (11). As explained in Section 4, this process is only done during the training. In our experiments, finding the least squares 2-bits quantization was as fast as the 2-bits greedy algorithm (see Section 3.2.4). Since the least squares 2-bits binary quantization is foldable, after recovering v_1 and v_2, we have $s_1(\boldsymbol{x}) = \text{sign}(\boldsymbol{x})$ and $s_2(\boldsymbol{x}) = \text{sign}(\boldsymbol{x} - v_1 \text{sign}(\boldsymbol{x}))$.

3.2.3 Least squares ternary algorithm

The boundaries of the optimization domain in (8) for $k = 2$, $v_2 = 0$ and $v_1 = v_2 = v$, correspond to 1-bit binary

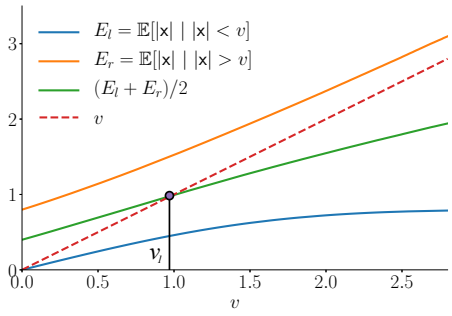

Figure 3. The conditional expectations in (10) for a random variable x with standard normal distribution. The optimal value for 2-bits quantization is shown with a solid dot.

and ternary [35] quantizations, respectively. The scaled ternary quantization maps each full precision value x to $\{-2v, 0, 2v\}$. Ternary quantization needs 2-bits for representation. However, when a hardware with sparse calculation support is available, for example as in EIE [19], using ternary quantization can be more efficient compared to a general 2-bits quantization. Setting $v_1 = v_2 = v$ in (10) and (11) we get the ternary optimality condition:

$$v = \frac{1}{2}\,\mathbb{E}_{\mathrm{x}\sim p}[|\mathrm{x}| \mid |\mathrm{x}| > v] \tag{12}$$

The process of solving for v in (12) is similar to that of solving for v_1 in (10) as described above. The optimality condition in (12) has been also obtained in [26] using a different approach.

3.2.4 k-bits greedy algorithm

In this section, we propose a greedy algorithm to compute k-bits scaled binary quantization, which we call Greedy Foldable (GF). It is given in Algorithm 1.

Algorithm 1: k-bits Greedy Foldable (GF) binary quantization: compute $\boldsymbol{x}^q$ given $\boldsymbol{x}$

$\boldsymbol{r} \leftarrow \boldsymbol{x}$
for $i \leftarrow 1$ **to** k **do**
  $v_i \leftarrow \text{mean}(\text{abs}(\boldsymbol{r}))$
  $\boldsymbol{s}_i \leftarrow \text{sign}(\boldsymbol{r})$ `// element-wise sign. For gradient of sign use STE.`
  $\boldsymbol{r} \leftarrow \boldsymbol{r} - v_i \boldsymbol{s}_i$ `// compute new residual.`
end
return $\boldsymbol{x} - \boldsymbol{r}$

In GF algorithm we compute a sequence of residuals. At each step, we greedily find the best $\boldsymbol{s}_i$ and v_i for the current residual using the least squares 1-bit binary quantization (5). Note that for $k = 1$ the GF is the same as the least squares 1-bit binary quantization.

Few of the other papers that have tackled the k-bits binary quantization to train quantized DNNs are as follows. In ReBNet [17], the authors proposed an algorithm similar to Algorithm 1, but considered v_i's as trainable parameters to be learned by back-propagation. [36] and [54] find k-bits binary quantization via alternating optimization for s_i's and v_i's. Note that, all these methods produce sub-optimal solutions.

4. Training binary networks

The loss functions in our quantized neural networks are non-differentiable due to the sign function in the quantizers. To address this challenge we use the training algorithm proposed in [11]. To compute the gradient of the sign function we use the Straight Through Estimator (STE) [3]: $d/dx\ \text{sign}(x) = \mathbf{1}_{|x|\leq 1}$. During the training we keep the full precision weights and use Stochastic Gradient Descent (SGD) to gradually update them in back-propagation. In the forward-pass, only the quantized weights are used.

During the training we compute quantizers (for both weights and activations) using the online statistics, i.e., the scalars in a k-bits scaled binary quantization (6) are computed based on the observed values. During the training we also store the running average of these scalars. During inference we use the stored quantized scalars to improve the efficiency. This procedure is similar to the update of the batch normalization parameters in a standard DNN training [30].

5. Experiments

We conduct experiments on the ImageNet dataset [13] using the ResNet-18 architecture [23]. The details of the architecture and training are provided in Appendix D.

We conduct three sets of experiments: (1) evaluate quantization error of activations of a pre-trained DNN, (2) evaluate the quantization error based on the classification accuracy of a post-training quantized network, and (3) evaluate the classification accuracy of during-training quantized networks. We report the quantization errors of the proposed binary quantization algorithms (least squares 1-bit, 2-bits, ternary, and the greedy foldable quantizations) and compare with the state-of-the-art algorithms BWN-Net [44], XNor-Net [44], TWN-Net [35], DoReFa-Net [56], ABC-Net [36], and LQ-Net [54].

5.1. Quantization error of activations

To quantify the errors of the introduced binary quantization algorithms we adopt the analysis performed by [1]. They show that the angle between $\boldsymbol{x}$ and $\boldsymbol{x}^q$ can be used as a measure of accuracy of a quantization scheme. They prove that when $\boldsymbol{x}^q = \text{sign}(\boldsymbol{x})$ and elements of $\boldsymbol{x}$ are i.i.d. $\sim \mathcal{N}(0, 1)$, $\angle(\boldsymbol{x}, \boldsymbol{x}^q)$ converges to $\sim 37^\circ$ for large N.

Method	k^a	k^w	Top-1	Top-5
Post Greedy Foldable	32	1	0.1	0.5
Post Greedy Foldable	32	2	0.3	1.1
Post Greedy Foldable	32	3	1.4	4.6
Post Greedy Foldable	32	4	5.3	14.1
Post Least Squares	32	2	5.3	13.9

Table 1. Validation accuracy of a quantized ResNet-18 trained on ImageNet. k^a and k^w are number of bits to quantize activations and weights, respectively.

Here we use the real data distribution. We trained a full precision network, and computed the activation tensors at each layer for a set of 128 images. In Figure 4 we show the angle between the full precision and quantized activations for different layers. When the least squares quantization is used, a significant reduction in the angle is observed compared to the greedy algorithm. The least squares 2-bits quantization is even better than the greedy 4-bits quantization for later layers of the network, for which activation tensors have more skewed distribution, make it harder for quantization in form of (6). Furthermore, the accuracy of the least squares quantization has less variance with respect to different input images and different layers of the network.

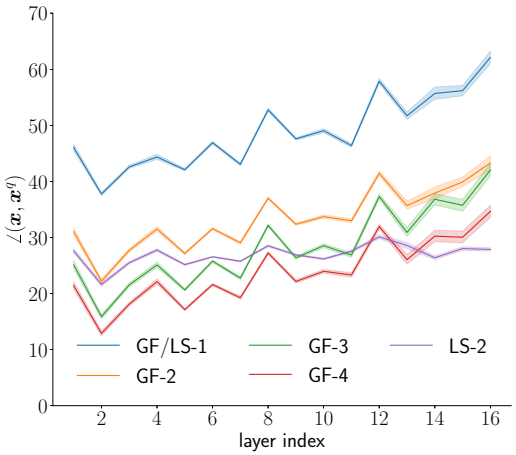

Figure 4. The angle between the full precision and the quantized activations for different layers of a trained full precision ResNet-18 architecture on ImageNet. The 95% confidence interval over different input images is shown. LS and GF refer to Least Squares and Greedy Foldable, respectively.

5.2. Post-training quantization

In this section we apply post-training quantization to the weights of a pre-trained full precision network. We then use the quantized network for inference and report the classification accuracy. This procedure can result in an acceptable accuracy for a moderate number of bits (e.g., 16 or 8). However, the error significantly grows with a lower number of bits, which is the case in this experiment. Therefore, we only care about the relative differences between different quantization strategies. This experiment demonstrates the effect of quantization errors on the accuracy of the quantized DNNs. The results are shown in Table 1. When the least squares 2-bits quantization is used, significant accuracy improvement (more than one order of magnitude) is observed compared to the greedy 2-bits quantization, which illustrate the effectiveness of the optimal quantization.

Method	k^a	k^w	top-1	top-5
XNor-Net [44]	1	1	51.2	73.2
Bi-Real Net [37]	1	1	56.4	79.5
XNor-Net++ [5]	1	1	57.1	79.9
Least Squares (ours)	1	1	**58.9**	**81.4**
Least Squares (ours)	T	1	**62.0**	**83.6**
HWGQ-Net [7]	2	1	59.6	82.2
LQ-Net [54]	2	1	62.6	84.3
Greedy Foldable (ours)	2	1	62.6	84.0
Least Squares (ours)	2	1	**63.4**	**84.6**
DoReFa-Net [56]	4	1	59.2	81.5
SYQ [15]	8	1	62.9	84.6
DoReFa-Net [56]	2	2	62.6	84.4
ABC-Net [36]	3	3	61.0	83.2
BWN-Net [44]	32	1	60.8	83.0
Least Squares (ours)	32	1	**66.1**	**86.5**
TWN-Net [35]	32	T	61.8	84.2
Full precision baseline	32	32	69.6	89.2

Table 2. Validation accuracy of ResNet-18 architecture on the ImageNet dataset. T refers to ternary quantization.

5.3. During-training quantization

To achieve higher accuracy we apply quantization during the training, so that the model can adapt to the quantized weights and activations. In Table 2 we report results from the related works in which ResNet-18 architecture with quantized weights and/or activations is trained on the ImageNet dataset for the classification task. The proposed least-squares quantization algorithms improve the classification accuracies when compared with the state-of-the-art (with even more number of bits) significantly. For all of our results we use some of the suggested training setups discussed in the literature (see Appendix D) to train the BNNs. These changes improved the performance of the baseline 1-bit XNor-Net [44] from 51.2% to 59.0%. With 2-bits the least squares quantization algorithm achieves a significantly better accuracy compared to the 2-bits greedy and 1-bit, with an identical training setup.

6. Conclusion

In this work, we analyze the accuracy of binary quantization to train DNNs with quantized weights and activations. We discuss methods to improve the accuracy of quantization, namely scaling and using more bits.

We introduce the rank-1 binary quantization, as a general scaling scheme. Based on a singular value analysis we motivate using the scaled binary quantization, a computationally efficient scaling strategy. We define a general k-bits scaled binary quantization. We provide provably least squares 1-bit, 2-bits, and ternary quantizations. In addition, we propose a greedy k-bits quantization algorithm. We show results for post and during-training quantization, and demonstrate significant improvement in accuracy when least squares quantization is used. We compare the proposed quantization algorithms with state-of-the-art BNNs on the ImageNet dataset and show improved classification accuracies.

A. Optimal rank-1 binary quantization

In this section, we find the optimal rank-1 binary quantization of an m by n matrix $\boldsymbol{X}$ discussed in Section 2:

$$\begin{aligned} \underset{\boldsymbol{X}_1,\boldsymbol{S}}{\text{minimize}} \quad & \|\boldsymbol{X}-\boldsymbol{X}_1\odot\boldsymbol{S}\|_F \\ \text{s.t.} \quad & \boldsymbol{S}\in\{-1,1\}^{m\times n} \\ & \boldsymbol{X}_1\in\mathbb{R}^{m\times n}, \quad \text{rank}(\boldsymbol{X}_1)=1 \end{aligned} \tag{13}$$

First, observe that the element-wise multiplication by -1 and $+1$ does not change the Frobenius norm. Therefore:

$$\begin{aligned} \min_{\boldsymbol{S},\boldsymbol{X}_1}\|\boldsymbol{X}-\boldsymbol{X}_1\odot\boldsymbol{S}\|_F &= \min_{\boldsymbol{S},\boldsymbol{X}_1}\|(\boldsymbol{X}-\boldsymbol{X}_1\odot\boldsymbol{S})\odot\boldsymbol{S}\|_F \\ &= \min_{\boldsymbol{S},\boldsymbol{X}_1}\|\boldsymbol{X}\odot\boldsymbol{S}-\boldsymbol{X}_1\|_F \end{aligned} \tag{14}$$

Furthermore, note that

$$\min_{\boldsymbol{S},\boldsymbol{X}_1}\|\boldsymbol{X}\odot\boldsymbol{S}-\boldsymbol{X}_1\|_F^2=\sigma_2^2(\boldsymbol{X}\odot\boldsymbol{S})+\ldots+\sigma_r^2(\boldsymbol{X}\odot\boldsymbol{S}) \tag{15}$$

Here $\sigma_i(\boldsymbol{X}\odot\boldsymbol{S})$ is the i'th singular value of $\boldsymbol{X}\odot\boldsymbol{S}$ and r is its rank. In addition for any $\boldsymbol{S}$:

$$\sum_{i=1}^{r}\sigma_i^2(\boldsymbol{X}\odot\boldsymbol{S})=\|\boldsymbol{X}\odot\boldsymbol{S}\|_F^2=\|\boldsymbol{X}\|_F^2 \tag{16}$$

Hence, to minimize the sum in (15) we need to find an $\boldsymbol{S}$ for which $\sigma_1^2(\boldsymbol{X}\odot\boldsymbol{S})$ is maximized:

$$\min_{\boldsymbol{S},\boldsymbol{X}_1}\|\boldsymbol{X}\odot\boldsymbol{S}-\boldsymbol{X}_1\|_F^2=\|\boldsymbol{X}\|_F^2-\max_{\boldsymbol{S}}\sigma_1^2(\boldsymbol{X}\odot\boldsymbol{S}) \tag{17}$$

$\sigma_1(\boldsymbol{X}\odot\boldsymbol{S})=\|\boldsymbol{X}\odot\boldsymbol{S}\|_2$ is the 2-norm of $\boldsymbol{X}\odot\boldsymbol{S}$. Therefore:

$$\max_{\boldsymbol{S}}\sigma_1^2(\boldsymbol{X}\odot\boldsymbol{S})=\max_{\boldsymbol{S}}\max_{\|\boldsymbol{r}\|_2=1}\|(\boldsymbol{X}\odot\boldsymbol{S})\boldsymbol{r}\|_2^2 \tag{18}$$

For any $\boldsymbol{S}$ and $\boldsymbol{r}\in\mathbb{R}^n$ we have $\|(\boldsymbol{X}\odot\boldsymbol{S})\boldsymbol{r}\|_2^2\le\||\boldsymbol{X}||\boldsymbol{r}|\|_2^2$ since for $1\le i\le m$ we have $|\sum_j S_{i,j}X_{i,j}r_j|\le\sum_j|X_{i,j}||r_j|$. Here $|\boldsymbol{X}|=\boldsymbol{X}\odot\text{sign}(\boldsymbol{X})$ is the element-wise absolute value of $\boldsymbol{X}$. Note that for $\boldsymbol{S}=\text{sign}(\boldsymbol{X})$ and $\boldsymbol{r}$ with positive values the inequality becomes an equality. Therefore:

$$\max_{\boldsymbol{S}}\max_{\|\boldsymbol{r}\|_2=1}\|(\boldsymbol{X}\odot\boldsymbol{S})\boldsymbol{r}\|_2^2=\max_{\|\boldsymbol{r}\|_2=1}\||\boldsymbol{X}||\boldsymbol{r}|\|_2^2 \tag{19}$$

Observe that the element-wise absolute value does not change the vector norm, i.e. $\|\,|\boldsymbol{r}|\,\|_2=\|\boldsymbol{r}\|_2$, and hence $|\boldsymbol{r}|$ is a unit vector when $\boldsymbol{r}$ is. Also for any $\boldsymbol{r}$ we have $\|\,|\boldsymbol{X}|\,\boldsymbol{r}\,\|_2^2\le\|\,|\boldsymbol{X}|\,|\boldsymbol{r}|\,\|_2^2$ since for $1\le i\le m$ we have $|\sum_j|X_{i,j}|r_j|\le\sum_j|X_{i,j}||r_j|$. So we have

$$\max_{\|\boldsymbol{r}\|_2=1}\||\boldsymbol{X}||\boldsymbol{r}|\|_2^2=\max_{\|\boldsymbol{r}\|_2=1}\||\boldsymbol{X}|\boldsymbol{r}\|_2^2=\sigma_1^2(|\boldsymbol{X}|) \tag{20}$$

Therefore, we showed that $\boldsymbol{S}=\text{sign}(\boldsymbol{X})$ and X_1 equal to the best rank-1 approximation of $|\boldsymbol{X}|$ (i.e. the first term in its SVD) is a solution of (13). ■

Now consider the case that entries of $\boldsymbol{X}$ are i.i.d. $\sim\mathcal{N}(0,1)$. In [4, 46] the authors show that the largest singular value of a random matrix with i.i.d. entries from a distribution with mean μ and bounded 4th moment (which is the case for standard folded normal distribution) asymptotes to $\sqrt{mn}\mu$ as m and n are increased (with $m/n\to$constant). Note that $\mathbb{E}[\frac{\mathbf{1}^T}{\sqrt{m}}|\boldsymbol{X}|\,|\boldsymbol{X}|^T\frac{\mathbf{1}}{\sqrt{m}}]=\mathbb{E}[\frac{\mathbf{1}^T}{\sqrt{n}}|\boldsymbol{X}|^T\,|\boldsymbol{X}|\frac{\mathbf{1}}{\sqrt{n}}]=mn\mu^2$ (with convergence given by the central limit theorem), and therefore, for large matrices the first left and right singular vectors are expected to be almost constant, that is: truncated-SVD-1$(|\boldsymbol{X}|)\approx\mu\mathbf{1}\mathbf{1}^T$ where $\mu=\sqrt{2/\pi}$. This is shown empirically in Figure 5. Therefore, the optimal rank-1 binary quantization captures $2/\pi\simeq0.64$ of the total energy of $\boldsymbol{X}$, making it a good approximation, and can be written as a scalar times a binary matrix.

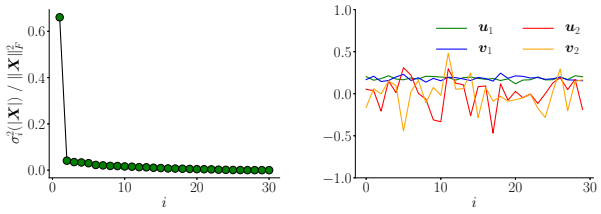

Figure 5. **Left**: Distribution of energy for $|\boldsymbol{X}|$, where $\boldsymbol{X}\in\mathbb{R}^{30\times30}$ is a standard normal random matrix. **Right**: Entries of the first left and right singular vectors of $|\boldsymbol{X}|$ (shown in green and blue) are almost constant.

B. Least squares 1-bit binary quantization

In this section, we solve (4). First, observe that:

$$\forall x\in\mathbb{R}:(v-x)^2<(-v-x)^2 \quad \text{iff} \quad x>0 \tag{21}$$

Therefore, the optimal choice for function s in (4) is $s(x) = \text{sign}(x)$. We can rewrite (4) as follows:

$$\begin{aligned} \underset{v}{\text{minimize}} \quad & \int_{-\infty}^{+\infty} p(x)(v - |x|)^2 dx \\ \text{s.t.} \quad & v \in \mathbb{R}_{\geq 0} \end{aligned} \tag{22}$$

Setting the gradient of the objective function in (22) with respect to v to zero, we get:

$$v = \frac{\int_{-\infty}^{+\infty} p(x)|x|dx}{\int_{-\infty}^{+\infty} p(x)dx} = \mathbb{E}_{\text{x}\sim p}[|\text{x}|] \qquad \blacksquare \tag{23}$$

C. Least squares 2-bits binary quantization

In this section, we solve the following optimization problem corresponding to the least squares 2-bits binary quantization as discussed in Section 3.2.2:

$$\begin{aligned} \underset{v_1,v_2,s_1,s_2}{\text{minimize}} \quad & \int_{-\infty}^{+\infty} p(x) \left(v_1 s_1(x) + v_2 s_2(x) - x\right)^2 dx \\ \text{s.t.} \quad & s_1, s_2 : \mathbb{R} \to \{-1, 1\}, \quad v_1 \geq v_2 \geq 0 \end{aligned} \tag{24}$$

First, we show that the optimal 2-bits binary quantization is foldable, i.e., $\forall x \in \mathbb{R}\ s_1(x) = \text{sign}(x)$ and $s_2(x) = \text{sign}(x - v_1 s_1(x))$. Observe that

$$\begin{aligned} f(x) &= \left(v_1 s_1(x) + v_2 s_2(x) - x\right)^2 \\ &= v_1^2 \left(1 + \frac{v_2}{v_1} s_1(x) s_2(x) - \frac{s_1(x)x}{v_1}\right)^2 \\ &\geq v_1^2 \left(1 + \frac{v_2}{v_1} s_1(x) s_2(x) - \frac{|s_1(x)x|}{v_1}\right)^2 = g(x) \end{aligned} \tag{25}$$

The inequality in (25) holds because $v_1 \geq v_2$, and therefore, $1 + \frac{v_2}{v_1} s_1(x) s_2(x) \geq 0$. The objective function in (24) is a weighted average of $f(x)$ with non-negative weights. For $x \in \mathbb{R}$ the inequality is strict if $s_1(x) \neq \text{sign}(x)$. In that case, flipping the value of both $s_1(x)$ and $s_2(x)$ reduces $f(x)$ to a strictly smaller $g(x)$. Hence, the optimal solution of (24) should satisfy $s_1(x) = \text{sign}(x)$ for all $x \in \mathbb{R}$.

For any v_1 and s_1 if we consider $y = x - v_1 s_1(x)$, the problem reduces to the 1-bit binary quantization for y. Based on the result showed in Appendix B for the optimal solution we have $s_2(x) = \text{sign}(y) = \text{sign}(x - v_1 s_1(x))$. This completes the proof to show that the optimal 2-bits binary quantization is foldable.

Next, we find the optimal values for v_1 and v_2. Substitute $s_1(x) = \text{sign}(x)$ and $s_2(x) = \text{sign}(x - v_1 s_1(x))$ in (24):

$$e(v_1, v_2) = \int_{-\infty}^{+\infty} p(x) \left(v_1 s_1(x) + v_2 s_2(x) - x\right)^2 dx = \int_0^{v_1} q(x)(x - v_1 + v_2)^2 dx + \int_{v_1}^{+\infty} q(x)(x - v_1 - v_2)^2 dx \tag{26}$$

Here $e(v_1, v_2)$ is the error as a function of v_1 and v_2, and $q(x) = p(-x) + p(x)$ is the folded distribution function. Assuming the optimal point occurs in the interior of the domain, it should satisfy the zero gradient condition: $\partial e/\partial v_1 = \partial e/\partial v_2 = 0$. Taking derivative from (26) with respect to v_1 and v_2 and set it to zero we get:

$$\begin{aligned} v_1 &= \int_0^{v_1} xq(x)dx + \int_{v_1}^{+\infty} xq(x)dx \\ &\quad + v_2 \left(\int_0^{v_1} q(x)dx - \int_{v_1}^{+\infty} q(x)dx\right) \\ v_2 &= -\int_0^{v_1} xq(x)dx + \int_{v_1}^{+\infty} xq(x)dx \\ &\quad + v_1 \left(\int_0^{v_1} q(x)dx - \int_{v_1}^{+\infty} q(x)dx\right) \end{aligned} \tag{27}$$

Simplifying (27) results in (10) and (11). $\blacksquare$

D. Details of training ResNet on ImageNet

In this section, we explain the details of how the DNN results reported in this paper are produced using some of the stable good practices from the literature. We use the standard training and validation splits of the ImageNet dataset, without hyper parameters search. We followed a similar architecture as XNor-Net [44]. The convolutional block that we use is depicted in Figure 1. We use PReLU [22, 6] non-linearity before the batch normalization as suggested by [44]. Also, we find it important to use bounded dynamic range, and therefore clip the values to $[-d, d]$. Without tuning, we picked $d = 2$, 3, 5, and 8 for $k = 1$, 2/T, 3, and 4 bits quantizations, respectively. Similar to the other BNNs for the first and last layers we use full precision. Also, as suggested by [37] we use full precision double shortcuts in ResNet architecture, which adds a small computational/memory overhead. We quantize weights per filter and activations per layer. As [7] we use first-order polynomial learning-rate annealing schedule (from $2.0e-4$ to $2.0e-7$) and train for 240 epochs with Adam optimizer. When use 120 epochs, with identical setup, we get top-1 accuracies for different activations quantizations (with ls-1 binary weights) as follows: ls-1: 58.1, T:61.1, gf-2:61.1, ls-2:62.4, and fp:65.2. We use smooth labels from a pre-trained full precision teacher [25] with unit temperature. Quantized networks are initialized randomly. We do not use weight decay. For the data augmentation we use the standard methods used to train full precision ResNet architecture. For training we apply random resize and crop to 224×224, followed by random horizontal flipping, color jittering, and lightening. For test we resize the images to 256×256 followed by a center cropping to 224×224.

References

[1] Alexander G Anderson and Cory P Berg. The high-dimensional geometry of binary neural networks. *arXiv preprint arXiv:1705.07199*, 2017.

[2] Hessam Bagherinezhad, Maxwell Horton, Mohammad Rastegari, and Ali Farhadi. Label refinery: Improving imagenet classification through label progression. *arXiv preprint arXiv:1805.02641*, 2018.

[3] Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013.

[4] Włodek Bryc and Jack W Silverstein. Singular values of large non-central random matrices. *Random Matrices: Theory and Applications*, page 2050012, 2019.

[5] Adrian Bulat and Georgios Tzimiropoulos. Xnor-net++: Improved binary neural networks. *arXiv preprint arXiv:1909.13863*, 2019.

[6] Adrian Bulat, Georgios Tzimiropoulos, Jean Kossaifi, and Maja Pantic. Improved training of binary networks for human pose estimation and image recognition. *arXiv preprint arXiv:1904.05868*, 2019.

[7] Zhaowei Cai, Xiaodong He, Jian Sun, and Nuno Vasconcelos. Deep learning with low precision by half-wave gaussian quantization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5918–5926, 2017.

[8] Yunpeng Chen, Haoqi Fang, Bing Xu, Zhicheng Yan, Yannis Kalantidis, Marcus Rohrbach, Shuicheng Yan, and Jiashi Feng. Drop an octave: Reducing spatial redundancy in convolutional neural networks with octave convolution. *arXiv preprint arXiv:1904.05049*, 2019.

[9] Jungwook Choi, Zhuo Wang, Swagath Venkataramani, Pierce I-Jen Chuang, Vijayalakshmi Srinivasan, and Kailash Gopalakrishnan. Pact: Parameterized clipping activation for quantized neural networks. *arXiv preprint arXiv:1805.06085*, 2018.

[10] François Chollet. Xception: Deep learning with depthwise separable convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1251–1258, 2017.

[11] Matthieu Courbariaux, Yoshua Bengio, and Jean-Pierre David. Binaryconnect: Training deep neural networks with binary weights during propagations. In *Advances in neural information processing systems*, pages 3123–3131, 2015.

[12] Matthieu Courbariaux, Itay Hubara, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Binarized neural networks: Training deep neural networks with weights and activations constrained to+ 1 or-1. *arXiv preprint arXiv:1602.02830*, 2016.

[13] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[14] Zhen Dong, Zhewei Yao, Amir Gholami, Michael W Mahoney, and Kurt Keutzer. Hawq: Hessian aware quantization of neural networks with mixed-precision. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 293–302, 2019.

[15] Julian Faraone, Nicholas Fraser, Michaela Blott, and Philip HW Leong. Syq: Learning symmetric quantization for efficient deep neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4300–4309, 2018.

[16] Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. *arXiv preprint arXiv:1803.03635*, 2018.

[17] Mohammad Ghasemzadeh, Mohammad Samragh, and Farinaz Koushanfar. Rebnet: Residual binarized neural network. In *2018 IEEE 26th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*, pages 57–64. IEEE, 2018.

[18] Suyog Gupta, Ankur Agrawal, Kailash Gopalakrishnan, and Pritish Narayanan. Deep learning with limited numerical precision. In *International Conference on Machine Learning*, pages 1737–1746, 2015.

[19] Song Han, Xingyu Liu, Huizi Mao, Jing Pu, Ardavan Pedram, Mark A Horowitz, and William J Dally. Eie: efficient inference engine on compressed deep neural network. In *2016 ACM/IEEE 43rd Annual International Symposium on Computer Architecture (ISCA)*, pages 243–254. IEEE, 2016.

[20] Song Han, Huizi Mao, and William J Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*, 2015.

[21] Song Han, Jeff Pool, Sharan Narang, Huizi Mao, Enhao Gong, Shijian Tang, Erich Elsen, Peter Vajda, Manohar Paluri, John Tran, et al. Dsd: Dense-sparse-dense training for deep neural networks. *arXiv preprint arXiv:1607.04381*, 2016.

[22] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015.

[23] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[24] Koen Helwegen, James Widdicombe, Lukas Geiger, Zechun Liu, Kwang-Ting Cheng, and Roeland Nusselder. Latent weights do not exist: Rethinking binarized neural network optimization. In *Advances in neural information processing systems*, pages 7531–7542, 2019.

[25] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

[26] Lu Hou and James T Kwok. Loss-aware weight quantization of deep networks. *arXiv preprint arXiv:1802.08635*, 2018.

[27] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

[28] Itay Hubara, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Quantized neural networks: Training neural networks with low precision weights and activations. *The Journal of Machine Learning Research*, 18(1):6869–6898, 2017.

[29] Forrest N Iandola, Song Han, Matthew W Moskewicz, Khalid Ashraf, William J Dally, and Kurt Keutzer. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and¡ 0.5 mb model size. *arXiv preprint arXiv:1602.07360*, 2016.

[30] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[31] Benoit Jacob, Skirmantas Kligys, Bo Chen, Menglong Zhu, Matthew Tang, Andrew Howard, Hartwig Adam, and Dmitry Kalenichenko. Quantization and training of neural networks for efficient integer-arithmetic-only inference. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2704–2713, 2018.

[32] Max Jaderberg, Andrea Vedaldi, and Andrew Zisserman. Speeding up convolutional neural networks with low rank expansions. *arXiv preprint arXiv:1405.3866*, 2014.

[33] Hyungjun Kim, Kyungsu Kim, Jinseok Kim, and Jae-Joon Kim. Binaryduo: Reducing gradient mismatch in binary activation network by coupling binary activations. In *International Conference on Learning Representations*, 2020.

[34] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[35] Fengfu Li, Bo Zhang, and Bin Liu. Ternary weight networks. *arXiv preprint arXiv:1605.04711*, 2016.

[36] Xiaofan Lin, Cong Zhao, and Wei Pan. Towards accurate binary convolutional neural network. In *Advances in Neural Information Processing Systems*, pages 345–353, 2017.

[37] Zechun Liu, Baoyuan Wu, Wenhan Luo, Xin Yang, Wei Liu, and Kwang-Ting Cheng. Bi-real net: Enhancing the performance of 1-bit cnns with improved representational capability and advanced training algorithm. In *Proceedings of the European conference on computer vision (ECCV)*, pages 722–737, 2018.

[38] Stuart Lloyd. Least squares quantization in pcm. *IEEE transactions on information theory*, 28(2):129–137, 1982.

[39] Ningning Ma, Xiangyu Zhang, Hai-Tao Zheng, and Jian Sun. Shufflenet v2: Practical guidelines for efficient cnn architecture design. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 116–131, 2018.

[40] Sachin Mehta, Mohammad Rastegari, Anat Caspi, Linda Shapiro, and Hannaneh Hajishirzi. Espnet: Efficient spatial pyramid of dilated convolutions for semantic segmentation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 552–568, 2018.

[41] Sachin Mehta, Mohammad Rastegari, Linda Shapiro, and Hannaneh Hajishirzi. Espnetv2: A light-weight, power efficient, and general purpose convolutional neural network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9190–9200, 2019.

[42] Asit Mishra, Eriko Nurvitadhi, Jeffrey J Cook, and Debbie Marr. Wrpn: wide reduced-precision networks. *arXiv preprint arXiv:1709.01134*, 2017.

[43] Eunhyeok Park, Sungjoo Yoo, and Peter Vajda. Value-aware quantization for training and inference of neural networks. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 580–595, 2018.

[44] Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi. Xnor-net: Imagenet classification using binary convolutional neural networks. In *European Conference on Computer Vision*, pages 525–542. Springer, 2016.

[45] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4510–4520, 2018.

[46] Jack W Silverstein. The spectral radii and norms of large dimensional non-central random matrices. *Stochastic Models*, 10(3):525–532, 1994.

[47] Xavier Suau, Luca Zappella, and Nicholas Apostoloff. Network compression using correlation analysis of layer responses. 2018.

[48] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

[49] Mingxing Tan and Quoc V Le. Efficientnet: Rethinking model scaling for convolutional neural networks. *arXiv preprint arXiv:1905.11946*, 2019.

[50] Kuan Wang, Zhijian Liu, Yujun Lin, Ji Lin, and Song Han. Haq: Hardware-aware automated quantization with mixed precision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8612–8620, 2019.

[51] Ziwei Wang, Jiwen Lu, Chenxin Tao, Jie Zhou, and Qi Tian. Learning channel-wise interactions for binary convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 568–577, 2019.

[52] Bichen Wu, Yanghan Wang, Peizhao Zhang, Yuandong Tian, Peter Vajda, and Kurt Keutzer. Mixed precision quantization of convnets via differentiable neural architecture search. *arXiv preprint arXiv:1812.00090*, 2018.

[53] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016.

[54] Dongqing Zhang, Jiaolong Yang, Dongqiangzi Ye, and Gang Hua. Lq-nets: Learned quantization for highly accurate and compact deep neural networks. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 365–382, 2018.

[55] Xiangyu Zhang, Xinyu Zhou, Mengxiao Lin, and Jian Sun. Shufflenet: An extremely efficient convolutional neural network for mobile devices. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6848–6856, 2018.

[56] Shuchang Zhou, Yuxin Wu, Zekun Ni, Xinyu Zhou, He Wen, and Yuheng Zou. Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients. *arXiv preprint arXiv:1606.06160*, 2016.

[57] Chenzhuo Zhu, Song Han, Huizi Mao, and William J Dally. Trained ternary quantization. *arXiv preprint arXiv:1612.01064*, 2016.

[58] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V Le. Learning transferable architectures for scalable image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8697–8710, 2018.