Journal of Computational Mathematics
Vol.37, No.3, 2019, 349–359.

http://www.global-sci.org/jcm
doi:10.4208/jcm.1803-m2017-0301

QUANTIZATION AND TRAINING OF LOW BIT-WIDTH CONVOLUTIONAL NEURAL NETWORKS FOR OBJECT DETECTION*

Penghang Yin
Department of Mathematics, University of California, Los Angeles, CA 90095, USA
Email: yph@ucla.edu
Shuai Zhang,[1)] Yingyong Qi and Jack Xin
Department of Mathematics, University of California, Irvine, CA 92697, USA
Email: szhang3@uci.edu, yqi@uci.edu, jack.xin@uci.edu

Abstract

We present LBW-Net, an efficient optimization based method for quantization and training of the low bit-width convolutional neural networks (CNNs). Specifically, we quantize the weights to zero or powers of 2 by minimizing the Euclidean distance between full-precision weights and quantized weights during backpropagation (weight learning). We characterize the combinatorial nature of the low bit-width quantization problem. For 2-bit (ternary) CNNs, the quantization of N weights can be done by an exact formula in $O(N \log N)$ complexity. When the bit-width is 3 and above, we further propose a semi-analytical thresholding scheme with a single free parameter for quantization that is computationally inexpensive. The free parameter is further determined by network retraining and object detection tests. The LBW-Net has several desirable advantages over full-precision CNNs, including considerable memory savings, energy efficiency, and faster deployment. Our experiments on PASCAL VOC dataset show that compared with its 32-bit floating-point counterpart, the performance of the 6-bit LBW-Net is nearly lossless in the object detection tasks, and can even do better in real world visual scenes, while empirically enjoying more than 4× faster deployment.

Mathematics subject classification: 90C26, 90C10, 90C90.
Key words: Quantization, Low bit width deep neural networks, Exact and approximate analytical formulas, Network training, Object detection.

1. Introduction

Deep convolutional neural networks (CNNs) have demonstrated superior performance in various computer vision tasks [3,13–16,18,22–24]. However deep CNNs typically have hundreds of millions of trainable parameters which easily take up hundreds of megabytes of memory, and billions of FLOPs for a single inference. This poses a significant challenge for the deployment of deep CNNs on small devices with limited memory storage and computing power such as mobile phones. To address this issue, recent efforts have been made to compress the model size [7,9] and train neural networks with heavily quantized weights, activations, and gradients [1, 2, 6, 7, 9, 17, 20, 21, 26–28], which demand less storage and fewer FLOPs for deployment. These models include BinaryConnect [1], BinaryNet [2], XNOR-Net [21], TWN [17], TTQ [28],

* Received December 24, 2017 / Accepted March 13, 2018 /
Published online August 16, 2018 /
[1)] P. Yin and S. Zhang contributed equally to this work.

DoReFa-Net [27] and QNN [9], to name a few. In particular, binary (1-bit) and ternary (2-bit) weight models not only enable high model compression rate, but also eliminate the need of most floating-point multiplications during forward and backward propagations, which shows promise to resolve the problem. Compared with binary models, ternary weight networks such as TWN strike a better balance between model size and accuracy. It has been shown that ternary weight CNNs [17] can achieve nearly lossless accuracy on MNIST [16] and CIFAR-10 [12] benchmark datasets. Yet with fully ternarized weights, there is still noticeable drop in performance on larger datasets like ImageNet [4], which suggests the necessity of relatively wider bit-width models with stronger performance for challenging tasks.

An incremental network quantization strategy (INQ) is proposed in [26] for converting pre-trained full-precision CNNs into low bit-width versions whose weights are either zero or powers of two. A b bit-width model can have $2^{b-1}+1$ distinct candidate values, in which 2 bits are used for representing the zero and the signs, while the remaining $b-2$ bits for the powers. More precisely, the parameters are constrained to $2^s \times \{0, \pm 2^{1-2^{b-2}}, \pm 2^{2-2^{b-2}}, \ldots, \pm 1\}$ associated with a layerwise scaling factor 2^s, s an integer depending only on the weight maximum in the layer. At inference time, the original floating-point multiplication operations can be replaced by faster and cheaper binary bit shifting. The quantization scheme of [26] is however heuristic.

In this paper, we present the exact solution of the general b-bit approximation problem of a real weight vector W^f in the least squares sense. If $b = 2$ and the dimension of W^f is N, the computational complexity of the 2 bit solution is $O(N \log N)$. At $b \geq 3$, the combinatorial nature of the solution renders direct computation too expensive for large scale tasks. We shall develop a semi-analytical quantization scheme involving a single adjustable parameter μ to set up the quantization levels. The exponent s in the scaling factor can be calculated analytically from μ and the numbers of the downward sorted weight components between quantization levels. If the weight vector comes from a Gaussian ensemble, the parameter μ can be estimated analytically. However, we found that the weight vectors in CNNs (in particular ResNet) are strongly non-Gaussian. In this paper, μ is determined based on the object detection performance after retraining the network. This seems to be a natural choice in general as quantization is often part of a larger computer vision problem as is here. Therefore, the optimal parameter μ should not be decided by approximation (the least squares problem) errors alone. Indeed, we found that at $b \geq 4$, $\mu = \frac{3}{4}\|W^f\|_\infty$ gives the best detection performance, which suggests that a percentage of the large weights plays a key role in representing the image features and should be encoded during quantization.

Network retraining is necessary after quantization as a way for the system to adjust and absorb the resulting errors. Besides warm start, INQ [24] requires a careful layerwise partitioning and grouping of the weights which are then quantized and re-trained incrementally group by group rather than having all weights updated at once. Due to both classification and detection networks involved in this work, we opted for a simpler retraining method, a variant of the projected stochastic gradient descent (SGD) method (see [1,17,21] and references therein). As a result, our LBW-Net can be trained either from scratch or a partial warm start. During each iteration, besides forward and backward propagations, only an additional low cost thresholding (projection) step is needed to quantize the full-precision parameters to zero or powers of two. We train LBW-Net with randomly initialized weights in the detection network (R-FCN [3]), and pre-trained weights in ResNet [8]. We conduct object detection experiments on PASCAL VOC data sets [5] as in [3,22]. We found that at bit-width $b = 6$, the accuracies of the quantized networks are well within 1% of those of their 32-bit floating-point counterparts on both

ResNet-50 and ResNet-101 backbone architectures. In some complex real world visual scenes, the 6-bit network even detects persons missed by the full-precision network.

The rest of the paper is organized as follows. In section 2, we construct the exact solution of the general low bit-width approximation problem and present our semi-analytical quantization scheme with a single adjustable parameter μ. We also outline the training algorithm and the choice of μ. In section 3, we describe our experiments, the datasets, the object detection results, the non-Gaussian and sparsity properties of the floating weights in training. In section 4, we conclude with remarks on future work.

2. Training Low Bit-width Convolutional Neural Networks

2.1. Weight quantization at low bit-width

For general quantization problem, we seek to minimize the Euclidean distance between the given full-precision weight vector W^f and quantized weight vector W^q, which is formulated as the following optimization problem:

$$\min_{W^q} \|W^q - W^f\|^2 \quad \text{subject to} \quad W^q \in \mathcal{Q},$$

where $\mathcal{Q}$ is the set of quantized states.

To quantize the full-precision weights into low-precision ones of b bits ($b \geq 2$), we constrain the quantized weights to the value set of $2^s \times \{0, \pm 2^{1-n}, \pm 2^{2-n}, \ldots, \pm 1\}$ for some integer $s \in \mathbb{Z}$, where $n = 2^{b-2}$ and 2^s serves as the scaling factor. The minimal distance problem becomes:

$$(s^*, Q^*) = \arg\min_{s \in \mathbb{Z}, Q} \|2^s Q - W^f\|^2 \quad \text{subject to} \quad Q_i \in \{0, \pm 2^{1-n}, \ldots, \pm 1\}. \tag{2.1}$$

Then the optimal quantized weight vector is given by $2^{s^*} Q^*$. A precise characterization of (2.1) is as follows.

Theorem 2.1. *Let $b \geq 2$, $n = 2^{b-2}$, and $k_0, \ldots, k_{n-1} \in \mathbb{N}$. Suppose that $W^f_{[k_0]}$ keeps the k_0 largest components in magnitude of W^f and zeros out the other components; $W^f_{[k_1]}$ extracts the next k_1 largest components and zeros out the other components, and so on. The solution Q^* to (2.1) is:*

$$Q^* = \sum_{t=0}^{n-1} \operatorname{sign}(W^f_{[k_t^*]}) 2^{-t},$$

where

$$(k_0^*, \ldots, k_{n-1}^*) = \arg\min_{k_0, \ldots, k_{n-1} \in \mathbb{N}} g\left(\sum_{t=0}^{n-1} \|W^f_{[k_t]}\|_1 2^{-t}, \sum_{t=0}^{n-1} k_t 2^{-2t} \right) \tag{2.2}$$

with

$$g(u, v) := v \left(2^{\lfloor \log_2 \frac{4u}{3v} \rfloor} - \frac{u}{v} \right)^2 - \frac{u^2}{v}.$$

The bracket $\lfloor \cdot \rfloor$ in $g(u, v)$ is the floor operation or the closest integer on the left. Moreover, the optimal power of scaling is:

$$s^* = \left\lfloor \log_2 \frac{4 \sum_{t=0}^{n-1} 2^{-t} \|W^f_{[k_t^*]}\|_1}{3 \sum_{t=0}^{n-1} k_t^* 2^{-2t}} \right\rfloor.$$

Proof. Let k_t be the number of entries in Q quantized to $\pm 2^{-t}$, $t = 0, \ldots, n-1$. It follows that

$$\|Q\|^2 = \sum_{t=0}^{n-1} k_t 2^{-2t} \quad \text{and} \quad |\langle Q, W^f \rangle| \le \sum_{t=0}^{n-1} \|W^f_{[k_t]}\|_1 2^{-t}. \tag{2.3}$$

Therefore, for any $s \in \mathbb{Z}$,

$$\begin{aligned}
&\|2^s Q - W^f\|^2 = 2^{2s}\|Q\|^2 - 2^{s+1}\langle Q, W^f\rangle + \|W^f\|^2 \\
\ge& 2^{2s}\sum_{t=0}^{n-1} k_t 2^{-2t} - 2^{s+1}\sum_{t=0}^{n-1}\|W^f_{[k_t]}\|_1 2^{-t} + \|W^f\|^2 \qquad \text{(by (2.3))} \\
=& \left(\sum_{t=0}^{n-1} k_t 2^{-2t}\right)\left(2^s - \frac{\sum_{t=0}^{n-1}\|W^f_{[k_t]}\|_1 2^{-t}}{\sum_{t=0}^{n-1} k_t 2^{-2t}}\right)^2 - \frac{(\sum_{t=0}^{n-1}\|W^f_{[k_t]}\|_1 2^{-t})^2}{\sum_{t=0}^{n-1} k_t 2^{-2t}} + \|W^f\|^2.
\end{aligned} \tag{2.4}$$

Since $s \in \mathbb{Z}$, by symmetry of the parabola, it suffices to find the nearest power of 2 to $\sum_{t=0}^{n-1}\|W^f_{[k_t]}\|_1 2^{-t} / \sum_{t=0}^{n-1} k_t 2^{-2t}$ to achieve the lower bound in (2.4). The nearest power 2^s satisfies

$$\frac{2^{s-1}+2^s}{2} \le \frac{\sum_{t=0}^{n-1}\|W^f_{[k_t]}\|_1 2^{-t}}{\sum_{t=0}^{n-1} k_t 2^{-2t}} < \frac{2^s + 2^{s+1}}{2},$$

or equivalently,

$$\log_2 \frac{4\sum_{t=0}^{n-1}\|W^f_{[k_t]}\|_1 2^{-t}}{3\sum_{t=0}^{n-1} k_t 2^{-2t}} - 1 < s \le \log_2 \frac{4\sum_{t=0}^{n-1}\|W^f_{[k_t]}\|_1 2^{-t}}{3\sum_{t=0}^{n-1} k_t 2^{-2t}}.$$

Therefore,

$$s^* = \lfloor \log_2 \frac{4\sum_{t=0}^{n-1}\|W^f_{[k_t]}\|_1 2^{-t}}{3\sum_{t=0}^{n-1} k_t 2^{-2t}} \rfloor. \tag{2.5}$$

Let us define

$$g(u,v) := v\left(2^{\log_2 \lfloor \frac{4u}{3v} \rfloor} - \frac{u}{v}\right)^2 - \frac{u^2}{v}.$$

Then we examine the minimum value of $g(\sum_{t=0}^{n-1}\|W^f_{[k_t]}\|_1 2^{-t}, \sum_{t=0}^{n-1} k_t 2^{-2t})$ over all possible combinations of natural numbers $k_0, \ldots, k_{n-1}$, i.e., the optimal numbers of quantized weights at the n levels are given by

$$(k_0^*, \ldots, k_{n-1}^*) = \underset{k_0,\ldots,k_{n-1}\in\mathbb{N}}{\operatorname{argmin}} \; g\left(\sum_{t=0}^{n-1}\|W^f_{[k_t]}\|_1 2^{-t}, \sum_{t=0}^{n-1} k_t 2^{-2t}\right).$$

Finally, to achieve the minimum in (2.4) with respect to $(k_0^*, \ldots, k_{n-1}^*)$, we must have

$$Q^* = \sum_{t=0}^{n-1} \operatorname{sign}(W^f_{[k_t^*]}) 2^{-t}$$

so that $\langle Q^*, W^f\rangle = \sum_{t=0}^{n-1}\|W^f_{[k_t^*]}\|_1 2^{-t}$, and choose $s^* = \lfloor \log_2 \frac{4\sum_{t=0}^{n-1}\|W^f_{[k_t^*]}\|_1 2^{-t}}{3\sum_{t=0}^{n-1} k_t^* 2^{-2t}} \rfloor$. □

In Theorem 2.1, we have assumed that the components of W^f have no ties in magnitudes, as such situation occurs with zero probability for random floating vectors from continuous

distributions. To solve the problem (2.1) by Theorem 2.1, we need to sort the elements of W^f in magnitude, and find the optimal numbers of weights $k_0^*, \ldots, k_{n-1}^*$ at n quantization levels by solving (2.2). We can then obtain the optimal scaling factor 2^{s^*}. The largest k_0^* weights (in magnitude) are quantized to $\pm 2^{s^*}$, and the next largest k_1^* weights to $\pm 2^{s^*-1}$, and so on. Finally, all the remaining small weights are pruned to 0.

The subproblem (2.2) is intrinsically combinatorial. In the simplest case $b = 2$ of the ternary weight networks, by Theorem 2.1,

$$k_0^* = \arg\min_{k_0 \in \mathbb{N}} \; g(\|W^f_{[k_0]}\|_1, k_0), \tag{2.6}$$

and the solution to (2.1) is given by:

$$Q^* = \text{sign}(W^f_{[k_0^*]}), \quad s^* = \lfloor \log_2 \frac{4\,\|W^f_{[k_0^*]}\|_1}{3\,k_0^*} \rfloor. \tag{2.7}$$

The formula (2.6)-(2.7) is first found by the authors in [25]. It shows that the weight ternarization mainly involves sorting magnitudes of the elements in W^f and computing a cumulative sum of the sorted sequence, which requires a computational complexity of $O(N \log(N))$, where N is number of entries in W^f.

When $b > 2$ and $n > 1$, solving (2.2) by direct enumeration becomes computationally too expensive for large scale problems such as convolutional neural networks and thus impractical. Hereby we propose a low-cost approximation of Q^*, motivated by the empirical quantization schemes in [17, 26]. To this end, by selecting a proper threshold value μ, we set

$$\tilde{Q}_i^* = \begin{cases} 0 & \text{if} \quad |W_i^f| < \frac{2^{2-n}}{3}\mu, \\ \text{sign}(W_i^f)2^{1-n} & \text{if} \quad \frac{2^{2-n}}{3}\mu \le |W_i^f| < 2^{2-n}\mu \\ \text{sign}(W_i^f)2^{-t} & \text{if} \quad 2^{-t}\mu \le |W_i^f| < 2^{-t+1}\mu, \; t = 1, \ldots, n-2, \\ \text{sign}(W_i^f) & \text{if} \quad \mu \le |W_i^f|. \end{cases} \tag{2.8}$$

Note that the case $t = n - 1$ in (2.8) needs special treatment because one of the neighboring quantized values is 0. The parameter μ is the only free parameter in (2.8).

Theorem 2.2. *The optimal power $\tilde{s}^*$ of the scaling factor with respect to the approximate $\tilde{Q}^*$ in* (2.8) *is*

$$\tilde{s}^* = \lfloor \log_2 \frac{4\sum_{t=0}^{n-1} 2^{-t}\|W^f_{[\tilde{k}_t^*]}\|_1}{3\sum_{t=0}^{n-1} \tilde{k}_t^* 2^{-2t}} \rfloor. \tag{2.9}$$

Here $W_{[\tilde{k}_t^*]}$ is defined as in Theorem 2.1*, and $\tilde{k}_t^*$ is the number of entries of W^f in the t-th largest group according to the division of* (2.8)*.*

Proof. Let $\tilde{Q}^*$ be defined as in (2.8). Since $\|2^s\tilde{Q}^* - W^f\|^2$ is a quadratic function in terms of 2^s, and

$$\|2^s\tilde{Q}^* - W^f\|^2 = \|\tilde{Q}^*\|^2 \left(2^s - \frac{\langle \tilde{Q}^*, W^f \rangle}{\|\tilde{Q}^*\|^2}\right)^2 - \frac{\langle \tilde{Q}^*, W^f \rangle^2}{\|\tilde{Q}^*\|^2} + \|W^f\|_1^2,$$

the minimizer $\tilde{s}^* \in \mathbb{Z}$ must occur at either $\lfloor \log_2 \frac{\langle \tilde{Q}^*, W^f \rangle}{\|\tilde{Q}^*\|^2} \rfloor$ or the ceiling $\lceil \log_2 \frac{\langle \tilde{Q}^*, W^f \rangle}{\|\tilde{Q}^*\|^2} \rceil$, the closest integer from the right. By grouping the elements in $\tilde{Q}$ according to their magnitudes, we have further

$$\langle \tilde{Q}^*, W^f \rangle = \sum_{t=0}^{n-1} 2^{-t} \|W^f_{[t]}\|_1, \quad \|\tilde{Q}^*\|^2 = \sum_{t=0}^{n-1} k_t 2^{-2t},$$

where $[t] := \{i : |\tilde{Q}^*_i| = 2^{-t}\}$, $t = 0, \ldots, b-2$, and k_t is the cardinality of $[t]$. This completes the proof. □

Remark 2.1. We remark that the output of $\tilde{Q}^*$ consists of mostly the scaled signs, hence $\tilde{Q}^*$ resembles a "phase factor". On the other hand, the scaling factor $2^{\tilde{s}^*}$ is the corresponding amplitude. Putting the two factors together, one can view the low bit-width weight approximation as an approximate polar decomposition of the real weight vector.

2.2. Training algorithm

We used a projected SGD-like algorithm as in [1, 17, 21] for training LBW-Net. At each gradient-descent step, the minibatch gradient is evaluated at the quantized weights, and a scaled gradient is subtracted from the full-precision weights instead of the quantized weights in standard projected gradient method. The quantization is done layer by layer by the formulas (2.8) and (2.9) with μ selected as $\frac{3}{4}\|W^f\|_\infty$ for each layer at bit-width 4 or above. To compute the optimal power s^* in (2.9), we find it sufficient to use the partial sums $\sum_{t=0}^{3} 2^{-t} \|W^f_{[\tilde{k}^*_t]}\|_1$ and $\sum_{t=0}^{3} \tilde{k}^*_t 2^{-2t}$ instead, as the tail values are negligible. In addition, we adopted batch normalization [10], adaptive learning rate, and Nesterov momentum [19] to promote training efficiency.

3. Experiments

We implemented our LBW-Net with the R-FCN [3] structure on PASCAL VOC [5] dataset which has 20 object categories. Same as [3], the training set is the union of VOC 2007 trainval and VOC 2012 trainval ("07+12"), and test results are evaluated on the VOC 2007 test set. So there are in total 16,551 images with 40,058 objects in the training set, and 4,952 images in the test set. The performance of object detection is measured by mean Average Precision (mAP). All mAP scores are computed with the Python version of the test codes provided by RCNN/Fast RCNN/Faster RCNN GitHub repositories. Our experiments are carried out on Caffe [11] with a Titan X GPU under Linux system.

3.1. R-FCN on PASCAL VOC

We employed ResNet-50 [8] as the backbone network architecture for R-FCN. In the experiments, we tested 4, 5, 6-bit LBW-Net and compared evaluation results with the corresponding 32-bit floating point models. For fair comparison, all these tests used the same initial weights, which are pre-trained convolutional feature maps from ResNet-50 while the weights in the other convolution layers are randomly initialized. A similar procedure is applied for experiments with ResNet-101. In [20], comparable results to ours were reported on ResNet-50 based detection. However, their method did not work on the deeper ResNet-101 based detection. Interesting

though, their approach succeeded in the classification task using ResNet-101, which suggests that quantization of detection networks is more challenging.

In the R-FCN structure, there is no fully-connected layer. We quantized all convolutional layers with the same low bit-width quantization formula for each layer.

Fig. 3.1. Curated examples of 6-bit LBW detection results on 3 sample images, compared with those from the corresponding full precision model. The left columns are results of 32-bit full-precision model, while the right images come from 6-bit LBW model. The network is R-FCN + ResNet-50, and the training data is 2007+2012 trainval. The threshold value 0.5 is used for display.

Table 3.1 shows mAP results from our experiments. With larger bit-width, LBW models achieved higher mAP values, true for both R-FCN + ResNet-50 and R-FCN + ResNet-101. The models trained with the 6-bit LBW scheme almost approach the best mAP of 32-bit full precision models. Besides these quantitative measures, in Fig. 3.1, we illustrate detection accuracies using R-FCN + ResNet-50 via samples processed by 6-bit LBW in comparison with those by the 'ground truth' full precision model. The first 2 photos are chosen from the 2007 Pascal VOC dataset and the third photo is taken at a university campus with a much more

Table 3.1: Object detection experiments on PASCAL VOC with R-FCN + ResNet-50/ResNet-101. Training set is VOC 07+12 trainval. The results are evaluated on VOC 07 test.

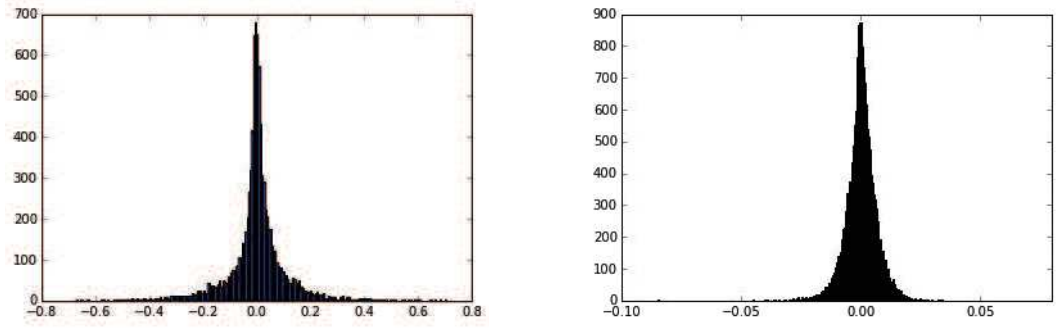
R-FCN, ResNet-50	mAP	R-FCN, ResNet-101	mAP
4-bit LBW	74.37%	4-bit LBW	76.79%
5-bit LBW	76.99%	5-bit LBW	77.83%
6-bit LBW	77.05%	6-bit LBW	78.24%
32-bit full-precision	77.46%	32-bit full-precision	78.94%

complicated visual scene. In the first 2 photos, both the 6-bit LBW and full precision models detected the major objects correctly, with nearly the same bounding box positions and high classification scores. In the third photo, the 6-bit LBW even surpassed the performance of the full precision model, by detecting a student at the very left side of the top staircase with a score of 0.710. Also the 3rd student from the right (the student in the middle) on the top staircase is detected with a score of 0.952 (0.906) by the 6 bit LBW vs. 0.886 (0.820) by the full precision model. Interestingly, these three students are all side-viewed.

3.2. Statistical Analysis of Weights

In Fig. 3.2, we illustrate the weight distributions of two floating convolutional layers by histograms. The p-values of a standard hypothesis testing procedure in statistics on normality showed up very small (less than 10^{-5}), indicating the strong non-Gaussian behavior of the floating weights in training. This phenomenon posed a challenge to the analytical effort of estimating the parameter μ in quantization using probability distribution functions as suggested for TWN [8].

Conv layer in residue block **RPN layer** [22]



Kurtosis = 6.113, Skewness = −0.112 Kurtosis = 9.398, Skewness = −0.481

Fig. 3.2. Histograms of the float weights in 2 convolutional layers of 32-bit full-precision trained R-FCN + ResNet-50 model. For both of these 2 layers, the p-values of normal distribution hypothesis testing are extremely small, less than 10^{-5}. Also the excess kurtosis measures are much larger than the value for normal distribution, which is 0. Thus these weights are far from being normally distributed.

In Table 3.2 and Table 3.3, we show the weight percentage distribution of two sample convolutional layers in R-FCN + ResNet50 between different magnitude levels of the quantization for low-bit width and full-precision models. The three low bit-width models involve truncation and encoding operations. The 6 bit-width columns appear to approach the 32-bit float columns on most rows. However, the percentages on the last three (two) rows under the low-bit LBW

Table 3.2: Statistics of low-bit and full precision weights (w) of one convolutional residual block layer in R-FCN + ResNet-50 at different bit-widths. For 4, 5, 6-bit LBW models, the weights in the first row of partition are exactly equal to 0, and come from rounding down small floating weights during training.

R-FCN, ResNet-50	4-bit LBW	5-bit LBW	6-bit LBW	32-bit full-precision
$\|w\| < 2^{-16}$	82.882%	10.072%	0.030%	0
$2^{-16} \leq \|w\| < 2^{-15}$	0	0	0.060%	0.076%
$2^{-15} \leq \|w\| < 2^{-14}$	0	0	0.141%	0.225%
$2^{-14} \leq \|w\| < 2^{-13}$	0	0	0.233%	0.271%
$2^{-13} \leq \|w\| < 2^{-12}$	0	0	0.486%	0.613%
$2^{-12} \leq \|w\| < 2^{-11}$	0	0	0.922%	1.283%
$2^{-11} \leq \|w\| < 2^{-10}$	0	0	1.964%	2.610%
$2^{-10} \leq \|w\| < 2^{-9}$	0	0	3.776%	4.945%
$2^{-9} \leq \|w\| < 2^{-8}$	0	0	7.343%	9.524%
$2^{-8} \leq \|w\| < 2^{-7}$	0	18.392%	13.509%	16.713%
$2^{-7} \leq \|w\| < 2^{-6}$	0	21.221%	21.221%	23.581%
$2^{-6} \leq \|w\| < 2^{-5}$	0	24.270%	24.270%	22.993%
$2^{-5} \leq \|w\| < 2^{-4}$	0	17.706%	17.706%	12.627%
$2^{-4} \leq \|w\| < 2^{-3}$	15.479%	6.700%	6.700%	3.784%
$2^{-3} \leq \|w\| < 2^{-2}$	1.408%	1.408%	1.408%	0.608%
$2^{-2} \leq \|w\| < 2^{-1}$	0.228%	0.228%	0.228%	0.098%
$2^{-1} \leq \|w\|$	0.003%	0.003%	0.003%	0

Table 3.3: Statistics of low-bit and full precision weights (w) of one RPN layer in R-FCN + ResNet-50 at different bit-widths. For 4, 5, 6-bit LBW models, the weights in the first row of partition are exactly equal to 0, and come from rounding down small floating weights during training.

R-FCN, ResNet-50	4-bit LBW	5-bit LBW	6-bit LBW	32-bit full-precision
$\|w\| < 2^{-19}$	58.188%	4.000%	0.016%	0.019%
$2^{-19} \leq \|w\| < 2^{-18}$	0	0	0.031%	0.022%
$2^{-18} \leq \|w\| < 2^{-17}$	0	0	0.047%	0.045%
$2^{-17} \leq \|w\| < 2^{-16}$	0	0	0.095%	0.089%
$2^{-16} \leq \|w\| < 2^{-15}$	0	0	0.185%	0.177%
$2^{-15} \leq \|w\| < 2^{-14}$	0	0	0.370%	0.355%
$2^{-14} \leq \|w\| < 2^{-13}$	0	0	0.751%	0.714%
$2^{-13} \leq \|w\| < 2^{-12}$	0	0	1.501%	1.413%
$2^{-12} \leq \|w\| < 2^{-11}$	0	0	2.993%	2.836%
$2^{-11} \leq \|w\| < 2^{-10}$	0	7.949%	5.952%	5.616%
$2^{-10} \leq \|w\| < 2^{-9}$	0	11.676%	11.685%	11.061%
$2^{-9} \leq \|w\| < 2^{-8}$	0	21.571%	21.588%	20.625%
$2^{-8} \leq \|w\| < 2^{-7}$	0	31.553%	31.539%	31.370%
$2^{-7} \leq \|w\| < 2^{-6}$	39.837%	21.137%	21.134%	23.257%
$2^{-6} \leq \|w\| < 2^{-5}$	1.953%	2.093%	2.091%	2.397%
$2^{-5} \leq \|w\| < 2^{-4}$	0.022%	0.021%	0.022%	0.004%
$2^{-4} \leq \|w\|$	0.0001%	0.0001%	0.0001%	0

models in Table 3.2 (3.3) are identical to each other and are much larger than the corresponding percentage in the full precision model. This shows that the trained low-bit LBW models

captured rather well a small percentage of the large weights. In deep CNNs, the large magnitude weights occupy a small percentage yet have a significant impact on the model accuracy. That is why we chose the partition parameter μ to be near the maximum norm of the weights.

It is worthwhile to note from the two tables that the 4-bit LBW can save lots of memory thanks to both low-bit weights and high sparsity. Over 82% (58%) of the weights are zeros in the convolutional residual block (RPN layer) of the R-FCN plus ResNet50 network. With the help of 'Mask' technology in circuit chip design, zero-valued weights will be skipped and the computational efficiency can be much improved. However, as shown in Table 3.1, the 4-bit LBW still suffers a few more percentages of accuracy loss than the 5-bit and 6-bit models. The 6-bit LBW model approximates the feature representation capability of the full precision network the best with a sufficient number of smaller levels of quantized weights. For that reason, it almost recovers the performance of the full precision model on the test set. The 6-bit LBW model saves around 5.3× weights memory with a small loss of accuracy. The memory savings and the near lossless accuracy of the 6-bit LBW may work well on a modern chip design where all multiplication operations in the convolutional layers can be replaced by bit-wise shift operations, thus highly improving the computing efficiency in applications.

4. Concluding Remarks

We discovered the exact solution of the general low-bit approximation problem of a real weight vector in the least squares sense, and proposed a low cost semi-analytical quantization scheme with a single adjustable parameter. This parameter is selected and optimized through training and testing on object detection data sets to approach the performance of the corresponding full precision model. The accuracy of our 6-bit width model is well-within 1% of the full precision model on PASCAL VOC data set, and can even outperform the full-precision model on real-world test images with complex visual scenes. Moreover, our low-bit-width model is 4× faster. In future work, we plan to improve the low bit width models (especially the 4 bit-width model) further by exploring alternative training algorithms and adapting quantization levels so that small weights are quantized with fewer levels than in the current work. Quantizing small weights with two many levels due to the restriction of powers of 2 is prone to introducing noise to the network. This problem can be solved in a more general quantization framework [20] where our quantization formulas in Theorem 2.1 and Theorem 2.2 extend.

Acknowledgments. The research of this project was partially supported by NSF grants DMS-1522383, IIS-1632935, and ONR grant N00014-16-1-2157.

References

[1] M. Courbariaux, Y. Bengio, J. David, BinaryConnect: Training Deep Neural Networks with Binary Weights during Propagations, in Advances in Neural Information Processing Systems, 2015.

[2] M. Courbariaux, I. Hubara, D. Soudry, R. El-Yaniv, Y. Bengio, Binarized Neural Networks: Training Neural Networks with Weights and Activations Constrained to +1 or -1, arXiv:1602.02830, 2016.

[3] J. Dai, Y. Li, K. He, J. Sun, R-FCN: Object Detection via Region-based Fully Convolutional Networks, in Advances in Neural Information Processing Systems, 2016; arXiv:1605.06409.

[4] J. Deng, W. Dong, R. Socher, L. Li, K. Li, F. Li, ImageNet: A Large-Scale Hierarchical Image Database, in IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2009.

[5] M. Everingham, L. Van Gool, C. Williams, J. Winn, A. Zisserman, A. The PASCAL Visual Object Classes (VOC) Challenge, IJCV, 2010.
[6] Y. Guo, A. Yao, Y. Chen, Dynamic Network Surgery for Efficient DNNs, in Advances in Neural Information Processing Systems, 2016.
[7] S. Han, H. Mao, W. Dally, (2016) Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding, International Conference on Learning Representations, 2016; arXiv:1510.00149.
[8] K. He, X. Zhang, S. Ren, J. Sun, Deep Residual Learning for Image Recognition, arXiv:1512.03385, 2015.
[9] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, Y. Bengio, Quantized Neural Networks: Training Neural Networks with Low Precision Weights and Activations, arXiv:1609.07061, 2016.
[10] S. Ioffe, C. Szegedy, Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift, arXiv:1502.03167, 2015.
[11] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, T. Darrell, Caffe: Convolutional Architecture for Fast Feature Embedding, arXiv:1408.5093, 2014.
[12] A. Krizhevsky, Learning Multiple Layers of Features from Tiny Images, 2009, www.cs.toronto.edu /*sim*kriz/index.htm.
[13] A. Krizhevsky, I. Sutskever, G. Hinton, Imagenet classification with deep convolutional neural networks, in Advances in Neural Information Processing Systems, 2012.
[14] Y. LeCun, Y. Bengio, G. Hinton, Deep Learning, *Nature*, **521**:7553 (2015), 436-444.
[15] Y. LeCun, B. Boser, J. Denker, D. Henderson, R. Howard, W. Hubbard, L. Jackel, Backpropagation Applied to Handwritten Zip Code Recognition, *Neural Computation*, **1**:4 (1989), 541-551.
[16] Y. LeCun, L. Bottou, Y. Bengio, R. Haffner, Gradient-based Learning Applied to Document Recognition, *Proceedings of the IEEE*, **86**:11 (1998), 2278-2324.
[17] F. Li, B. Zhang, B. Liu, Ternary Weight Networks, arXiv preprint arXiv:1605.04711, 2016.
[18] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, Ssd: Single shot multibox detector, arXiv:1512.02325, 2015.
[19] Y. Nesterov, A Method for Solving the Convex Programming Problem with Convergence Rate $O(1/k^2)$, *Soviet Mathematics Doklady*, **27**:2 (1983), 372-376.
[20] E. Park, J. Ahn, S. Yoo, Weighted-Entropy-Based Quantization for Deep Neural Networks, CVPR, 2017.
[21] M. Rastegari, V. Ordonez, J. Redmon, A. Farhadi, XNOR-Net: ImageNet Classification Using Binary Convolutional Neural Networks, *European Conference on Computer Vision*, (2016), 525-542.
[22] S. Ren, K. He, R. Girshick, J. Sun, Faster r-cnn: Towards Real-time Object Detection with Region Proposal Networks, in Advances in Neural Information Processing Systems, 2015.
[23] S. Simonyan, A. Zisserman, Very Deep Convolutional Networks for Large-scale Image Recognition, arXiv:1409.1556, 2014.
[24] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, A. Rabinovich, Going deeper with convolutions, in IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015
[25] P. Yin, S. Zhang, J. Xin, Y.Y. Qi, Training Ternary Neural Networks with Exact Proximal Operator, arXiv:1612.06052v1, 2016.
[26] A. Zhou, A. Yao, Y. Guo, L. Xu, Y. Chen, Incremental Network Quantization: Towards Lossless CNNs with Low-Precision Weights, in International Conference on Learning Representations, 2017; arXiv.1702.03044.
[27] S. Zhou, Y. Wu, Z. Ni, X. Zhou, H. Wen, Y. Zou, DoReFa-Net: Training Low Bitwidth Convolutional Neural Networks with Low Bitwidth Gradients, arXiv: 1606.06160, 2016.
[28] C. Zhu, S. Han, H. Miao, W. Dally, Trained Ternary Quantization, arXiv:1612.01064, 2016.