






Learning Compression from Limited Unlabeled Data

Xiangyu He^{1,2}  and Jian Cheng^{1,2,3}  

¹ National Laboratory of Pattern Recognition, Institute of Automation, Chinese Academy of Sciences, Beijing, China

{xiangyu.he, jcheng}@nlpr.ia.ac.cn

² University of Chinese Academy of Sciences, Beijing, China

³ Center for Excellence in Brain Science and Intelligence Technology, Beijing, China

Abstract. Convolutional neural networks (CNNs) have dramatically advanced the state-of-art in a number of domains. However, most models are both computation and memory intensive, which arouse the interest of network compression. While existing compression methods achieve good performance, they suffer from three limitations: (1) the inevitable retraining with enormous labeled data; (2) the massive GPU hours for retraining; (3) the training tricks for model compression. Especially the requirement of retraining on original datasets makes it difficult to apply in many real-world scenarios, where training data is not publicly available. In this paper, we reveal that re-normalization is the practical and effective way to alleviate the above limitations. Through quantization or pruning, most methods may compress a large number of parameters but ignore the core role in performance degradation, which is the Gaussian conjugate prior induced by batch normalization. By employing the re-estimated statistics in batch normalization, we significantly improve the accuracy of compressed CNNs. Extensive experiments on ImageNet show it outperforms baselines by a large margin and is comparable to label-based methods. Besides, the fine-tuning process takes less than 5 min on CPU, using 1000 unlabeled images.

Keywords: Deep neural networks · Label-free network compression

1 Introduction

Convolutional neural networks (CNNs) have achieved impressive performances in many challenging problems [15, 24], and even surpass human-level for certain tasks such as ImageNet classification [16]. As CNN-based recognition systems [3] continue to grow, it is critical to improve inference efficiency while maintaining accuracy [5].

Since network compression introduces efficient approximations to CNNs and compressed models require less memory and fewer operations, parameter quantization [8, 18, 30], pruning [11, 13] and low-rank [33, 38] representations have

become a topic of interest in the deep learning community. Especially quantization, with the boom of AI chips, will be the workhorse in industry. While these techniques have driven advances in power efficiency, they still face a considerable accuracy loss under low-bit or highly sparse compression. Retraining on the original dataset is usually inevitable. Unfortunately, the retraining process needs a sufficiently large open training dataset which is inaccessible for many real-world applications, such as medical diagnosis [9], drug discovery and toxicology [1]. Therefore, it is imperative to avoid retraining or require no training data.

In this work, we alleviate the accuracy degradation in direct network compression through label-free fine-tuning. For network quantization, which comprises two primary components: weight quantization and feature map quantization. Intuitively, the quantization error determines the performance loss. Therefore, we propose the Quasi-Lloyd-Max algorithm to minimize the weight quantization error. To further improve the accuracy of compressed networks, we explore the reason of feature map distortion. In light of Bayesian networks, we reveal that statistic-shift of batch normalization results in the accuracy degradation of direct compression. When network parameters misfit approximate Gaussian distribution, the prior assumption of mean and variance should mismatch the corrupted features. By employing the re-estimated statistics in batch normalization, the performance of compressed CNNs can be rapidly recovered. Extensive experiments on 4-bit quantization and pruning demonstrate the robustness of this viewpoint.

Compared with conventional label-based compression methods, the main contributions of this paper are as follows:

- We reveal the hidden factor why direct network compression results in performance degradation, and prove that 4-bit or sparse representation remains capable of original tasks without retraining.
- A Quasi-Lloyd-Max algorithm is proposed to minimize the weight quantization error on 4-bit networks.
- The fine-tuning time decreases from days (GPU) to minutes (CPU), by using limited unlabeled data.

2 Related Work

The redundant parameters of deep neural networks induce inefficient computation and large memory footprint. Most compression approaches can be viewed as the regularization techniques to solve these problems. Recently, along with the existence of TPU [22] and low precision BLAS [10], parameter fixed-point representation has frequently been discussed. Traditional hash-based vector quantization, such as HashNet [2], may not directly benefit from customized hardware. In contrast, 8-8-bit structures get TensorRT [27] or TPU [22] support easily. Bit-oriented methods with potential $64\times$ acceleration, such as BC [7], FFN [34], BNN [6] and XNOR-Net [30], compress DNNs to extreme 1 bit ($\sim 32\times$ compression), while suffering an irreversible accuracy loss. INQ [39] shows the reasonable

performance of 2^n framework; nevertheless, plenty of labeled data is required for retraining.

For low-rank representation, early studies share the same starting point: using matrix low-rank approximation to reduce the computation. Conventional network structures with regular filters and large feature maps are friendly to matrix decomposition. Generalized Singular Vector Decomposition [38], Tucker Decomposition [23] and Tensor Block Term Decomposition [33, 35] are widely used on AlexNet [24], VGG-16 [31] and GoogleNet [32]. At the cost of negligible loss in accuracy, they gain several times acceleration with a certain amount of compression. Very recently, MobileNet [17] with channel-wise convolution shows the potential capability to extract distinguishing features within limited parameters. Most notably, this network structure is identical to the decomposed matrix, which invalidates the current decomposition methods. Similar problems still arise in ResNet [16] with 1×1 filters.

Weight pruning benefits from Sparse GEneral Matrix Multiplication (Sparse GEMM) and highly optimized hardware design [14]. Combining with clustering and Huffman coding [13], promising compression results without accuracy loss were reported. The problem is hundreds [11] or even thousands [14] of retraining epochs are time-consuming, and is still heavily reliant on labeled datasets.

By using feature map fitting, [4, 36] implicitly learn from the well-trained networks through the Euclidean distance between feature maps of full-precision and compressed networks. Nevertheless, deeper network structures and imbalanced class samples would be the nightmare to hand-tuned layer by layer analysis.

3 Weight Quantization

Since quantization has been the mainstream compression technique in industry, we first review the cause of quantization, then discuss three hardware-friendly quantizers under different metrics. The scheme with least accuracy loss is adopted in further feature map recovery.

3.1 Cause

In early research, [12, 19] show that it is possible to train deep neural networks using 16-bit fixed-point numbers. Fixed-point computation with high speed and low power consumption is much more friendly to embedded devices. The small circuits would allow for the configuration of more arithmetic units. Besides, the low-bit data representation minimizes the memory footprint, which reduces the data transmission time for the customized device like FPGA [12, 13, 25].

3.2 ℓ_2 Norm Metric

Since customized hardware units have fully supported fixed-point multiplication and addition, quantizing float numbers to their nearest fixed-point representations by *shift* and *carry* operations can easily accelerate inference time. Suppose

the fixed-point number is represented as $[I_{bit} : F_{bit}]$, and the Integer part plus the Fraction part yields the real number. Mathematically this problem, dubbed round-to-nearest, can be stated as follows,

$$\begin{aligned} \mathbf{Q}^* &= \arg \min_{\mathbf{Q}} J(\mathbf{Q}) = \|\mathbf{W} - \mathbf{Q}\|_2^2 \\ \text{s.t. } \mathbf{Q}_i &\in \{-2^I/2, -2^I/2 + 2^{-F}, \dots, 0, \dots, 2^I/2 - 2^{-F}\} \end{aligned}$$

where \mathbf{Q} is forced to fit the large number in \mathbf{W} . This metric minimizes the loss function at the cost of small numbers and becomes much more sensitive to outliers. That is, large numbers determine the bit-width selection of I_{bit} and F_{bit} . To partly solve this problem, a scaling factor $\alpha \in \mathbb{R}$ is introduced,

$$\alpha^*, \mathbf{Q}^* = \arg \min_{\alpha > 0, \mathbf{Q}} J(\alpha, \mathbf{Q}) = \|\mathbf{W} - \alpha \mathbf{Q}\|_2^2$$

It has been proved that scaling factor could dramatically enlarge the domain of values [30]. Although the function is convex in each variable only, they are not convex in each variable together. It is infeasible to solve $J(\alpha, \mathbf{Q})$ in the sense of finding global minima, especially under the discrete constraint. However, it is possible to find local minima using iterative numerical optimization. Consider the following problem,

$$\alpha^*, \mathbf{Q}^* = \arg \min_{\alpha > 0} (\alpha^2 \mathbf{Q}^T \mathbf{Q} - 2\alpha \mathbf{Q}^T \mathbf{W} + c), \quad (1)$$

where \mathbf{Q} corresponds to a set of fixed-point numbers and $c = \sum_i \mathbf{W}_i^2$ is an α and \mathbf{Q} independent constant. Thus, for any given \mathbf{Q} , the optimal α is

$$\alpha^* = \frac{\mathbf{Q}^T \mathbf{W}}{\mathbf{Q}^T \mathbf{Q}}. \quad (2)$$

By substituting α^* into (1), the optimization problem leads to the partial derivatives $\frac{\partial J(\alpha, \mathbf{Q})}{\partial \mathbf{Q}}$. Setting it to zero, then project the solution to given discrete space

$$\mathbf{Q}^* \approx \mathbf{Fix}(\mathbf{W}/\alpha^*). \quad (3)$$

Algorithm 1 iteratively updates α^* and \mathbf{Q}^* through quantizer $\mathbf{Fix}(\cdot)$, such as round-to-nearest, 2^n (i.e., quantized to nearest power of 2) or uniform quantization (i.e., quantized to nearest quantization interval endpoints). Following the iterative update rule, the Euclidean distance between \mathbf{W} and $\alpha \mathbf{Q}$ is optimized in each iteration.

3.3 Discrete Entropy Metric

Similar to the squared Euclidean distance (ℓ_2) which is the canonical example of a Bregman distance, another useful measure is the generalized Kullback-Leibler divergence (KL) generated by the convex function $\sum_i p_i \ln p_i$. In this case,

$$\begin{aligned} \alpha^*, \mathbf{Q}^* &= \arg \min_{\alpha > 0, \mathbf{Q}} D(\alpha \mathbf{Q} \|\mathbf{W}) = \sum_i (|\mathbf{W}_i| \ln \frac{\mathbf{W}_i}{\alpha \mathbf{Q}_i} - |\mathbf{W}_i| + \alpha |\mathbf{Q}_i|) \\ \text{s.t. } \alpha > 0, \mathbf{Q}_i &\in \{\pm 2^0 \Delta, \pm 2^1 \Delta, \dots, \pm 2^{k-1} \Delta\} \end{aligned}$$

Algorithm 1. Quasi-Lloyd-Max Algorithm.

Require: Full precision weights \mathbf{W} , metric $\mathbf{J}(\cdot)$ (ℓ_2 , KL, etc.) and quantizer $\mathbf{Fix}(\cdot)$.

Ensure: Updated $\widetilde{\mathbf{W}} \approx \alpha^* \mathbf{Q}^*$ and fixed-point \mathbf{Q}^* .

- 1: **for** k^{th} filter in l^{th} layer **do**
 - 2: $\alpha_{l,k} \leftarrow$ Initialize parameters
 - 3: **repeat**
 - 4: $\alpha_{l,k}^*, \mathbf{Q}_{l,k}^* \leftarrow \arg \min_{\alpha, \mathbf{Q}} \mathbf{J}(\alpha_{l,k}, \mathbf{Q}_{l,k});$ {Fix α , solve \mathbf{Q} ; Fix \mathbf{Q} , solve α }
 - 5: $\text{swap}(\alpha_{l,k}, \alpha_{l,k}^*);$
 - 6: **until** convergence of parameters $\alpha_{l,k}$
 - 7: $\mathbf{Y}_l \leftarrow \widetilde{\mathbf{W}}_l * \widetilde{\mathbf{X}}_l \approx \alpha_l^* (\mathbf{Q}_l^* \otimes \widetilde{\mathbf{X}}_l);$ {low-bit convolution + cblas_sscal}
 - 8: $\widetilde{\mathbf{X}}_{l+1} \leftarrow \psi(\mathbf{Y}_l);$ {ReLU + INT 8-bit quantization}
 - 9: **end for**
-

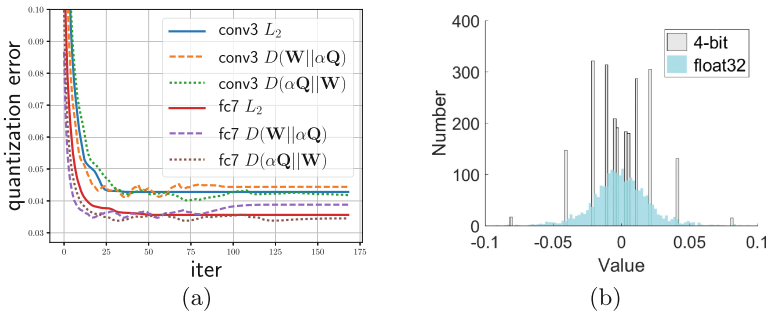


Fig. 1. (a) Quasi-Lloyd-Max convergence comparison of different metrics. The Euclidean distance between \mathbf{W} and $\alpha\mathbf{Q}$ is reported. (b) The distributions of quantized 4-bit (2^n quantization) and full-precision weights of the third convolution layer (some bars get merged for clearness)

where Δ corresponds to the minimum value of 2^n in k -bit quantization. Like ℓ_2 norm metric, this loss function is also lower bounded by zero.

Consider the function of $D(\alpha\mathbf{Q}||\mathbf{W})$ to be differentiated partially with respect to the elements of \mathbf{Q} , which is

$$\frac{\partial D(\alpha\mathbf{Q}||\mathbf{W})}{\partial \mathbf{Q}_i} = -\frac{|\mathbf{W}_i|}{\mathbf{Q}_i} + \alpha \cdot \text{sgn}(\mathbf{Q}_i). \tag{4}$$

With fixed α , similarly we have

$$\frac{\partial D(\alpha\mathbf{Q}||\mathbf{W})}{\partial \alpha} = -\frac{1}{\alpha} \sum_i |\mathbf{W}_i| + \sum_i |\mathbf{Q}_i|. \tag{5}$$

By setting both equations to zero, we obtain a pair of local minima of KL Divergence. Hence, the solutions to $D(\alpha\mathbf{Q}||\mathbf{W})$ are $\alpha^* = \frac{\sum_i |\mathbf{W}_i|}{\sum_i |\mathbf{Q}_i^*|}$ and $\mathbf{Q}^* = \mathbf{Fix}(\frac{\mathbf{W}}{\alpha^*})$ through Quasi-Lloyd-Max iterations.

In mathematics, generalized Kullback-Leibler Divergence is similar to a metric, but satisfies neither the triangle inequality nor symmetry. We further test

$D(\mathbf{W}||\alpha\mathbf{Q})$ as a weight quantization loss on AlexNet, shown in Fig. 1(b). Following the same procedure, we obtain

$$\frac{\partial D(\mathbf{W}||\alpha\mathbf{Q})}{\partial \alpha} = \sum_i |\mathbf{Q}_i| \ln \alpha + \sum_i |\mathbf{Q}_i| \left(\ln \frac{|\mathbf{Q}_i|}{|\mathbf{W}_i|} \right) \quad (6)$$

$$\frac{\partial D(\mathbf{W}||\alpha\mathbf{Q})}{\partial \mathbf{Q}_i} = \alpha \cdot \text{sgn}(\mathbf{Q}_i) \cdot \ln \frac{\alpha \mathbf{Q}_i}{\mathbf{W}_i}. \quad (7)$$

In this case, $\alpha^* = \exp\left(\frac{\sum_i |\mathbf{Q}_i^*| \ln \frac{|\mathbf{W}_i|}{|\mathbf{Q}_i^*|}}{\sum_i |\mathbf{Q}_i^*|}\right)$ and \mathbf{Q}^* remains the same as Eq. (4).

Taking the third convolution layer and the second fully connected layer of AlexNet as examples, Fig. 1(a) shows the convergence under different metrics. In our evaluations, all metrics converge in the first few iterations and obtain nearly the same quantization error. Since ℓ_2 yields more steady convergence speed, we evaluate the accuracy of different quantizer under ℓ_2 norm metric. As listed in Table 1 (whole network quantization except the first layer), 2^n outperforms other quantizers by a large margin; thus we follow this setting in the next experiments.

Table 1. Quantizer comparison of 4-bit Weights and 8-bit activations (ℓ_2 Norm)

Models		Round-to-nearest	Uniform	2^n	Full-precision
AlexNet	Top-1	48.32	43.12	58.66	60.43
	Top-5	72.93	67.73	81.17	82.47
ResNet-18	Top-1	45.92	50.18	55.76	69.08
	Top-5	71.33	75.63	79.75	89.03
ResNet-50	Top-1	54.61	55.99	68.14	75.30
	Top-5	77.98	79.00	87.98	92.11

3.4 Feature-Based Metric

In general, feature map extracted from input data is more crucial than weights in computer vision tasks. To fit the output features rather than pre-trained weights would further improve the performance [36]. Taking full-precision features \mathbf{Y} and quantized input activations $\tilde{\mathbf{X}}$ into account, we obtain the multi-objective optimization problem:

$$\alpha^*, \mathbf{Q}^* = \arg \min_{\alpha > 0, \mathbf{Q}} \|\mathbf{W} - \alpha \mathbf{Q}\|_2^2 + \lambda \|\mathbf{Y} - \alpha \tilde{\mathbf{X}} \mathbf{Q}^T\|_2^2. \quad (8)$$

With $\lambda = 0$, Eq. (8) degrades to ℓ_2 metric. For large λ , feature map fitting becomes more crucial. This problem could be solved by Quasi-Lloyd-Max in a

similar way. The closed-form solutions of each step are

$$\alpha^* = \frac{\lambda \sum_{i=1}^m \mathbf{Q}^T \tilde{\mathbf{X}}_i^T \mathbf{Y}_i + \mathbf{Q}^T \mathbf{W}}{\lambda \sum_{i=1}^m \mathbf{Q}^T \tilde{\mathbf{X}}_i^T \tilde{\mathbf{X}}_i \mathbf{Q} + \mathbf{Q}^T \mathbf{Q}} \quad (9)$$

$$(\alpha^* \lambda \sum_{i=1}^m \tilde{\mathbf{X}}_i^T \tilde{\mathbf{X}}_i + \alpha^* \mathbf{I}) \mathbf{Q} = \lambda \sum_{i=1}^m \tilde{\mathbf{X}}_i^T \mathbf{Y}_i + \mathbf{W}, \quad (10)$$

where \mathbf{I} corresponding to $n \times n$ unit matrix and m refers to m samples. If $\tilde{\mathbf{X}}_i^T \tilde{\mathbf{X}}_i$ is symmetric positive definite, then by using *modified Cholesky decomposition*, one may simplify Eq.(10) as $\alpha^* (\lambda \sum_i \tilde{\mathbf{X}}_i^T \tilde{\mathbf{X}}_i + \mathbf{I}) = \mathbf{LDL}^T$, where L is a lower triangular matrix with unit diagonal elements and D is a diagonal matrix with positive elements on the diagonal. To solve $\mathbf{LDL}^T \mathbf{x} = \mathbf{y}$, we only need to address $\mathbf{Lx}' = \mathbf{y}$ and $\mathbf{DL}^T \mathbf{x} = \mathbf{x}'$, which is faster and with better numerical stability.

However, given limited unlabeled data, there is no global measurement to facilitate the selection of λ . In our experiments, the iterative numerical approximation to solve \mathbf{Q} can be hugely affected by the different settings. Hence, the explicit feature-map based method is deprecated in our further evaluations. Compared with various metrics, Table 2 shows that the ℓ_2 norm could better reflect the weight fitting error.

Table 2. Metric Comparison of Direct 4-bit 2^n Weight Quantization

Models		ℓ_2 Norm	$D_{(\mathbf{W} \alpha\mathbf{Q})}$	$D_{(\alpha\mathbf{Q}) \mathbf{W}}$	Fmap-based
AlexNet	Top-1	58.90	56.30	57.59	–
	Top-5	81.43	79.40	80.26	–

4 Feature Recovery

To further improve the performance of compressed networks, we focus on the ‘‘Gaussian-like’’ feature distribution. From a Bayesian perspective, the conjugate prior induced by batch normalization results in the performance gap between full-precision network and post hoc compression. Therefore, we can use batch normalization to refine a well-trained network with low-bit or sparse representation.

4.1 Bayesian Networks

The methodology of CNNs is to find the maximum a posteriori (MAP) weights given a training dataset (\mathbf{D}) and a prior distribution $p(\mathbf{W})$ over model parameters \mathbf{W} . Suppose that \mathbf{D} consists of N batch samples $\{(x_i, y_i)_{i=1:N}\}$, then $p(\mathbf{W}|\mathbf{D}) = \frac{p(\mathbf{D}|\mathbf{W})p(\mathbf{W})}{p(\mathbf{D})}$. Due to the difficulty in calculating $p(\mathbf{D})$, it is common to approximate $p(\mathbf{W}|\mathbf{D})$ using a variational distribution $q_\tau(\mathbf{W})$. By optimizing

the variational parameters τ so that the Kullback-Leiber (KL) divergence is minimized:

$$\mathcal{L}(\tau) = -\mathbb{E}_{q_\tau(\mathcal{W})}[\log p(\mathbf{D}|\mathcal{W})] + KL(q_\tau(\mathcal{W})||p(\mathcal{W})) \quad (11)$$

$$= -\int_{\mathcal{W}} q_\tau(\mathcal{W}) \log p(\mathbf{D}|\mathcal{W}) d\mathcal{W} + KL(q_\tau(\mathcal{W})||p(\mathcal{W})). \quad (12)$$

Equation (13) is known as the evidence-lower-bound (ELBO), assuming i.i.d. observation noise.

In practice, a Monte Carlo integration is usually employed to estimate the expectation term $\mathbb{E}_{q_\tau(\mathcal{W})}[\log p(\mathbf{D}|\mathcal{W})]$. Using weight samples $\hat{\mathcal{W}}^i \sim q_\tau(\mathcal{W})$ for each batch i , leads to the following approximation:

$$\mathcal{L}(\tau) := -\frac{1}{N} \sum_{i=1}^N \log p(\mathbf{D}|\hat{\mathcal{W}}^i) + KL(q_\tau(\mathcal{W})||p(\mathcal{W})) \quad (13)$$

$$:= \underbrace{-\frac{1}{N} \sum_{i=1}^N \log p(\mathbf{y}_i|\mathbf{x}_i, \hat{\mathcal{W}}^i)}_{\text{negative log-likelihood}} + \underbrace{KL(q_\tau(\mathcal{W})||p(\mathcal{W}))}_{\text{KL divergence}}. \quad (14)$$

Especially, for batch normalization parameters $\{\mu_B, \sigma_B\} \in \mathcal{W}$, we regard the inference at training time as a stochastic process, estimated mean and variance based on samples in a mini-batch are two stochastic variables. Assume i.i.d. M samples where $\mathbf{z}_i = \overline{Wx} \sim \mathcal{N}(\mu, \sigma^2)$ and $\mu_i = \frac{1}{M} \sum_{k=1}^M \mathbf{z}_k$. By using central limit theorem (CLT) for sufficient random sampling through SGD, we have $\mu_B \sim \mathcal{N}(\mu, \frac{\sigma^2}{M})$. Due to $\mathbb{E}[(\mathbf{z}_i - \mu)^2] = \sigma^2$, similarly we obtain $\sigma_B^2 \sim \mathcal{N}(\sigma^2, \frac{\mathbb{E}[(\mathbf{z}_i - \mu)^4] - \sigma^4}{M})$.

4.2 KL Divergence and Weight Regularization

Probabilistically, $p(\mathbf{D}|\mathcal{W}) = \prod_{i=1}^N p(\mathbf{y}_i|\mathbf{x}_i, \mathcal{W})$, the posterior $p(\mathbf{y}_i|\mathbf{x}_i, \mathcal{W})$ expresses a predictive distribution generated by a parametric model \mathcal{W} , e.g., the cross-entropy criterion for multi-classification. The negative loglikelihood defines \mathcal{L} as follows:

$$\mathcal{L}(\mathbf{y}) = -\frac{1}{N} \sum_{i=1}^N \log p(\mathbf{y}_i|\mathbf{x}_i, \mathcal{W}) + \frac{\lambda}{2} \|\omega\|_2^2. \quad (15)$$

where ω is learnable parameters such as weights, and \mathcal{W} also includes random parameters such as μ_B, σ_B .

Since both $\mathcal{L}(\tau)$ and $\mathcal{L}(\mathbf{y})$ are solved by gradient descent, the second terms of Eqs. (15) and (17) illustrate the connection between KL divergence (i.e., $p(\mathcal{W})$ w.r.t the estimated distribution $q_\tau(\mathcal{W})$) and weight regularization:

$$\frac{\partial KL(q_\tau(\mathcal{W})||p(\mathcal{W}))}{\partial \omega} = \frac{\partial \frac{\lambda}{2} \omega^T \omega}{\partial \omega}. \quad (16)$$

The regularization term can be viewed as a log-prior distribution over weights, such as Gaussian derived from ℓ_2 norm. Under the constraint of low-bit or sparsity, the penalty term introduces different priors (e.g., spike-and-slab in pruning) which hugely affect the approximation to $p(\mathbf{W})$. We now describe how weight compression corrupts batch normalization parameters.

For random variables in batch normalization, the KL divergence between approximation $\mathcal{N}(\mu_q, \sigma_q^2)$ and true distribution $\mathcal{N}(\mu_p, \sigma_p^2)$ can be calculated using:

$$KL(q(\mathcal{W})||p(\mathcal{W})) = \frac{(\mu_q - \mu_p)^2}{2\sigma_p^2} + \log \frac{\sigma_p}{\sigma_q} + \frac{\sigma_q^2}{2\sigma_p^2} - \frac{1}{2}.$$

Since μ_p, σ_p won't change during training, which is independent to ω , thus $\mu'_p = \sigma'_p = 0$, and then $\frac{\partial KL}{\partial \omega} = \frac{(\mu_q - \mu_p)\mu'_q}{\sigma_p^2} + \frac{(\sigma_q^2 - \sigma_p^2)\sigma'_q}{\sigma_q \sigma_p^2}$. The optimal approximation $\mu_q \rightarrow \mu_p, \sigma_q^2 \rightarrow \sigma_p^2$ reaches its limit when regularization term solved by SGD (partial derivative is zero). When we compress the well-trained networks, the weight regularization has changed implicitly, in another word, former estimations should introduce a great bias. Fortunately, as proved in Sect. 4.2, the expectations of μ_q and σ_q^2 converge to the real distribution parameters, then it is possible to renew the distorted features through re-estimation.

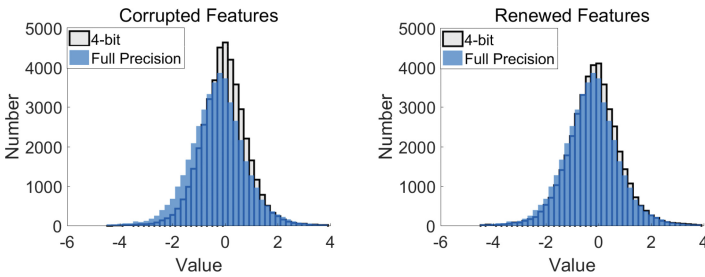


Fig. 2. Feature distribution comparison for AlexNet 5th batch-normalization layer

4.3 Renew Distorted Features

While it is impractical to update weights through inference on unlabeled data, re-estimation on $\mu_{\mathbf{B}}$ and $\sigma_{\mathbf{B}}$ is still feasible. From [21], the mean and variance of activations holds that

$$\mathbb{E}[\hat{x}] := \mathbb{E}[\tilde{\mu}_{\mathbf{B}}] \tag{17}$$

$$Var[\hat{x}] := \frac{m}{m-1} \mathbb{E}[\tilde{\sigma}_{\mathbf{B}}^2], \tag{18}$$

where $\mathbb{E}(\tilde{\mu}) = \frac{1}{m} \sum_{i=1}^m \tilde{x}_i$ and $\mathbb{E}(\tilde{\sigma}^2) = \frac{1}{m} \sum_{i=1}^m (\tilde{x}_i - \tilde{\mu})^2$.

In Bayesian theory, if the posterior distribution is in the same probability distribution family as the prior, then the prior is called a conjugate prior for the likelihood function. Especially, Gaussian distribution is a conjugate prior for the likelihood that is also Gaussian. In this case, we have shown that batch normalization parameters obey normal distribution and combine the empirical observations that output feature of batch normalization is “more Gaussian” [20, 21], one may derive that convolution, or inner-product layer tends to be a Gaussian likelihood. Thus, after compression, by choosing a new Gaussian prior (i.e., re-normalization or re-estimation), it will be more likely that the posterior distribution is also Gaussian:

$$P_{Gaussian} \propto P_{likelihood} \times P_{normal}.$$

Since batch normalization is commonly employed after convolution, the distribution of distorted features can be directly renewed. After the re-normalization, Fig. 2 shows that the distribution has been restored. Nevertheless, interpreting compressed networks as a likelihood function is a weak approximation. The performance of extremely quantized networks, such as binary or ternary, will not be improved since the corruption of likelihood function. In those cases, retraining on the original dataset is somehow inevitable.

5 Experiments

In this section, we verify the effectiveness of proposed methods on ImageNet dataset (ILSVRC 2012). Generally speaking, training-free quantization or pruning on deep neural networks is challenging, but we achieve much closer accuracy to full precision networks. We implement weight pruning and low-bit quantization on three representative CNNs: AlexNet [30], ResNet-18 [16] and MobileNet [17]. Besides, we also evaluate on ResNet-50 [16] to examine the validity of re-normalization on deeper network structures. All images are resized to have 256 pixel at short dimension and then a central crop of 224×224 is selected for re-normalization and evaluation. No data augmentation was used for all experiments.

5.1 Network Quantization

8-bit quantization with few samples or, ideally, without input data is becoming the workhorse in industry. As shown in Table 3, our 8-8-bit has reached the comparable accuracy with the full precision network. To achieve higher efficiency on embedded devices, we prove that even 4-bit weights could reach approximately 32-bit level. Using the same 4-bit weights in Sect. 3.2, we re-normalize those models on 1K images randomly selected from ILSVRC 2012 training dataset without label information.

As shown in Table 4, the performance of 4-8-bit network (except the first layer) was hugely improved from direct quantization. Compared with Nvidia

Table 3. Results of 8-8-bit (whole network weights & features 8-bit) quantization on ILSVRC2012 validation dataset. Round-to-nearest with ℓ_2 metric was adopted in 8-bit weights. For 8-bit feature maps, we just quantize float numbers to nearest fixed-points

Models		Our baseline	Our Gap	TensorRT Baseline	TensorRT[27] Gap
AlexNet	Top-1	60.43	+0.50	57.08	-0.08
	Top-5	82.47	+0.29	80.06	-0.08
ResNet-18	Top-1	69.08	-0.08	-	-
	Top-5	89.03	-0.06	-	-
ResNet-50	Top-1	75.30	-0.27	73.23	-0.20
	Top-5	92.11	-0.02	91.18	-0.03

Table 4. Final Performance of Network 2^n Quantization. Accuracy loss corresponding to full precision network is reported (4-bit Weights & 8-bit Activations)

Models		Baseline	w/o ReNorm	w/ ReNorm
AlexNet	Top-1	60.43	-1.77	-0.39
	Top-5	82.47	-1.30	-0.20
ResNet-18	Top-1	69.08	-13.21	-1.83
	Top-5	89.03	-9.28	-1.01
ResNet-50	Top-1	75.30	-7.16	-2.14
	Top-5	92.11	-4.13	-0.99
MobileNet	Top-1	70.81	-70.80	-9.75
	Top-5	89.85	-89.82	-6.37

TensorRT, 1250 images were used to update the parameters of 8-bit networks; we need 1000 images to learn 4-bit quantization. Results on AlexNet, ResNet-18, and ResNet-50 show the steady performance improvements, which have nearly approached the 32-bit level. MobileNet, with channel-wise convolution layers, is far more challenging to quantize. After straightforward 4-bit weight quantization, the accuracy dropped to nearly zero. This delicate network structure is equivalent to the low-rank representation of Tensor Block Term Decomposition [33]. For this reason, channel-wise convolution with little redundancy is naturally difficult to compress. Since the runtime speed of 8-bit MobileNet on CPU has already only 31 ms (Tensorflow 1.1.0), 4-bit could be a trade-off between even higher speed and lower accuracy.

Table 5 further shows the comparison between accuracy and learning cost. Our 4-8-bit is still competitive with retraining methods. In some cases, 4-8-bit even outperforms some label-based counterparts on AlexNet. For 4-4-bit, slightly different from Sect. 3.2, we quantize features to nearest 2^n (without scale) during the process of re-normalization.

Compared with the 8-8-bit framework, 4-8-bit achieves not only $2\times$ model compression but higher runtime speed. Low bit-width enables more fixed-point

Table 5. Quantization comparison for AlexNet and ResNet-18. Top-1 and Top-5 gap to the corresponding full-precision network is reported. Label-based retraining methods are marked as “+Label”. The bit width before and after “+” is for weight and activation respectively. Not reported retraining epoch was shown as “*”. “ ~ 0 ” requires no backward propagation

AlexNet				
Bit Precision	Method	Top-1 gap	Top-5 gap	Epochs
8 + 8	DoREFA [40]	-2.90	-	* + Label
	Going Deeper [29]	-0.88	-1.06	~ 0
	Ours	+0.50	+0.29	~ 0
5 + 32	INQ [39]	+0.15	+0.23	~ 8 + Label
4 + 4	WQ [28]	-1.2	-1.1	~ 6 + Label
4 + 8	Ours	-0.39	-0.20	~ 0
5 + 4	LogQuant [26]	-	-3.20	*
4 + 4	Ours	-3.24	-2.13	~ 0
ResNet-18				
4 + 32	INQ [39]	+0.62	+0.32	~ 8 + Label
4 + 8	Ours	-1.83	-1.01	~ 0

multiplications at the same clock frequency of the chip. This could provide dramatic data-level parallelism to achieve higher speedup. Besides, retraining methods can still benefit from feature map recovery. 3-8-bit AlexNet with +25.43% Top-1 and +26.86% Top-5 improvement yields 50.69% (Top-1) and 74.87% (Top-5) accuracy. This result provides a better starting point for retraining 3-bit networks.

5.2 Weight Pruning

To further verify the conclusion in Sect. 4.2, we apply network pruning (based on absolute value) to well-trained parameters. Figure 3(a) shows the trade-off between compression rate and accuracy. Within one iteration, i.e., using 1K images, we recover the performance to the practical level (solid line in Fig. 3(a)). This steady performance improvement not only appeared in network quantization but also in weight pruning.

Since AlexNet with over-parameterized inner-product layers is the typical network structure to examine the effectiveness of pruning approach, we compare the typical pruning approach [14] with ours on compression rate. As listed in Table 6, our method even pruned more parameters on two layers, especially Fc1 with most parameters in AlexNet. The overall compression rate of FC was still very close. Considering the training cost of both methods, ours has a significant advantage of high-efficiency. Due to the accuracy loss under high compression rate, we show the trade-off between training cost and performance in Table 7.

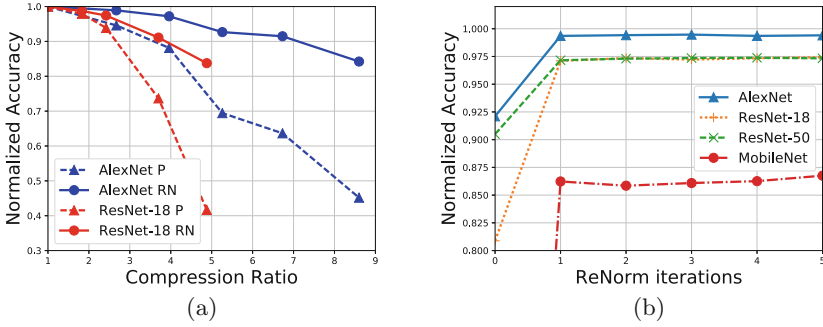


Fig. 3. (a) Normalized accuracy of the compressed networks under different compression rate. “1” indicates original network precision. The performance of direct pruning and re-normalized network are shown as “P” and “RN”. We stop pruning when the accuracy drops to 0.85 (normalized). (b) Normalized accuracy changes over re-normalization iterations. 1K different images were used in each iteration

In our experiments, deeper networks, such as ResNet-50, and lightweight structures, such as MobileNet, obtain the same results. For $3\times$ pruning, MobileNet achieves +53.82% Top-5 improvement to 78.43%, with 43% convolution layer and 7.3% fully connected layer parameters. ResNet-50 yields +6.92% Top-5 improvement to 90.00%, with 35% convolution layer and 10% fully connected layer parameters. The performance improvements are consistent in all our experiments, indicating that better performance becomes available by higher performance network.

Table 6. Model Sparsity Comparison on AlexNet

Layer	$\frac{ w \neq 0 }{ w }$ % [14]	Total. %	$\frac{ w \neq 0 }{ w }$ % (Ours)	Total. %
Conv1	~84%	~37%	63.1%	48.6%
Conv2	~38%		42.1%	
Conv3	~35%		49.8%	
Conv4	~37%		50.0%	
Conv5	~37%		49.0%	
Fc1	~9%	~10%	7.6%	12.6%
Fc2	~9%		14.5%	
Fc3	~25%		52.4%	
10,000 iterations			1 iteration	
120W labeled images			1K unlabeled images	

Table 7. The comparison for different compressed models with the number of training epochs and the final compression rate. “*” indicates the not reported training epoch. Label-based retraining methods are marked as “+Label”

Methods	Top-1	Epochs	Compression	Parameters
Dynamic Surgery [11]	56.91	~ 140 + Label	17.7 \times	3.48M
Fastfood-32-AD [37]	58.07	* + Label	2 \times	32.8M
Fastfood-16-AD [37]	57.10	* + Label	3.7 \times	16.4M
Han et al. [14]	57.23	≥ 960 + Label	9 \times	6.7M
Naive Cut [14]	42.82	0	4.4 \times	13.8M
Ours	55.28	\sim 0	6.73 \times	9.26M

5.3 Time Consumption

As listed in Table 8, most networks take only a few minutes to refine the distorted features, and as illustrated in Fig. 3(b), using more images has almost no contribution to the final accuracy. Setting batch size to 1K is just a trade-off between memory size and the sampling error of $\mathbb{E}(\hat{x})$ and $Var(\hat{x})$. By using large memory GPU, the whole process may take only a few seconds. This should lead to reduced time consumption of several orders of magnitudes. We believed that learning time speedup with limited unlabeled data is far more practical in real-world applications since slightly accuracy loss is unnoticeable to customers.

Table 8. Time consumption of feature recovery (1 batch = 1K images), evaluated on Intel Xeon CPU E5-2680 v4 @2.40 GHz x2

	AlexNet	ResNet-18	ResNet-50	MobileNet
1 batch	64s	172s	295s	197s

6 Conclusion

In this paper, we analyze the compression loss from Bayesian perspective and prove that batch normalization statistics misfit is one of the crucial reason for the performance loss. By using the proposed Quasi-Lloyd-Max and re-normalization, we quantize 4-bit networks to nearly full-precision level without retraining. In the experiments of network pruning, we further prove the robustness of this theorem. Our learning process is much more efficient than existing methods since considerably less data are required. In conclusion, we partly solve the real-world challenge of learning from limited unlabeled data to compress deep neural networks, which could be applied in a wide range of applications.

Acknowledgements. This work was supported in part by National Natural Science Foundation of China (No. 61332016), the Strategic Priority Research Program of Chinese Academy of Science, Grant No. XDBS01000000.

References

1. Burbidge, R., Trotter, M.W.B., Buxton, B.F., Holden, S.B.: Drug design by machine learning: support vector machines for pharmaceutical data analysis. *Comput. Chem.* **26**(1), 5–14 (2002)
2. Chen, W., Wilson, J.T., Tyree, S., Weinberger, K.Q., Chen, Y.: Compressing neural networks with the hashing trick. In: Bach, F.R., Blei, D.M. (eds.) *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6–11 July 2015*, pp. 2285–2294. JMLR.org (2015). <http://jmlr.org/proceedings/papers/v37/chenc15.html>
3. Chen, X., Kundu, K., Zhang, Z., Ma, H., Fidler, S., Urtasun, R.: Monocular 3D object detection for autonomous driving. In: *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, 27–30 June 2016*, pp. 2147–2156. IEEE Computer Society (2016)
4. Cheng, J., Wu, J., Leng, C., Wang, Y., Hu, Q.: Quantized CNN: A unified approach to accelerate and compress convolutional networks. *IEEE Trans. Neural Netw. Learn. Syst.*, 1–14 (2017)
5. Cheng, J., Wang, P., Li, G., Hu, Q., Lu, H.: Recent advances in efficient computation of deep convolutional neural networks. *Front. IT EE* **19**(1), 64–77 (2018). <https://doi.org/10.1631/FITEE.1700789>
6. Courbariaux, M., Bengio, Y.: BinaryNet: training deep neural networks with weights and activations constrained to +1 or -1. arXiv [abs/1602.02830](https://arxiv.org/abs/1602.02830) (2016)
7. Courbariaux, M., Bengio, Y., David, J.: Binaryconnect: Training deep neural networks with binary weights during propagations. In: Cortes, C., Lawrence, N.D., Lee, D.D., Sugiyama, M., Garnett, R. (eds.) *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, 7–12 December 2015, Montreal, Quebec, Canada*, pp. 3123–3131. Curran Associates, Inc. (2015). <http://papers.nips.cc/paper/5647-binaryconnect-training-deep-neural-networks-with-binary-weights-during-propagations>
8. Dettmers, T.: 8-bit approximations for parallelism in deep learning. arXiv [abs/1511.04561](https://arxiv.org/abs/1511.04561) (2015)
9. Djuric, U., Zadeh, G., Aldape, K., Diamandis, P.: Precision histology: how deep learning is poised to revitalize histomorphology for personalized cancer care. *NPJ Precis. Oncol.* **1**, 22 (2017)
10. Group, G.: gemmlowp: a small self-contained low-precision GEMM library (2016). <https://github.com/google/gemmlowp>
11. Guo, Y., Yao, A., Chen, Y.: Dynamic network surgery for efficient DNNs. In: Lee, D.D., Sugiyama, M., von Luxburg, U., Guyon, I., Garnett, R. (eds.) *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, 5–10 December 2016, Barcelona, Spain*, pp. 1379–1387. Curran Associates, Inc. (2016). <http://papers.nips.cc/paper/6165-dynamic-network-surgery-for-efficient-dnns>

12. Gupta, S., Agrawal, A., Gopalakrishnan, K., Narayanan, P.: Deep learning with limited numerical precision. In: Bach, F.R., Blei, D.M. (eds.) Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6–11 July 2015, pp. 1737–1746. JMLR.org (2015). <http://jmlr.org/proceedings/papers/v37/gupta15.html>
13. Han, S., Mao, H., Dally, W.J.: Deep compression: compressing deep neural network with pruning, trained quantization and huffman coding. arXiv [abs/1510.00149](https://arxiv.org/abs/1510.00149) (2015)
14. Han, S., Pool, J., Tran, J., Dally, W.J.: Learning both weights and connections for efficient neural network. In: Cortes, C., Lawrence, N.D., Lee, D.D., Sugiyama, M., Garnett, R. (eds.) Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, 7–12 December 2015, Montreal, Quebec, Canada. pp. 1135–1143. Curran Associates, Inc. (2015). <http://papers.nips.cc/paper/5784-learning-both-weights-and-connections-for-efficient-neural-network>
15. He, K., Gkioxari, G., Dollár, P., Girshick, R.B.: Mask R-CNN. In: IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, 22–29 October 2017. pp. 2980–2988. IEEE Computer Society (2017). <https://doi.org/10.1109/ICCV.2017.322>
16. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, 27–30 June 2016, pp. 770–778. IEEE Computer Society (2016)
17. Howard, A.G., et al.: MobileNets: efficient convolutional neural networks for mobile vision applications. arXiv [abs/1704.04861](https://arxiv.org/abs/1704.04861) (2017)
18. Hu, Q., Wang, P., Cheng, J.: From hashing to CNNs: training binary weight networks via hashing. In: McIlraith, S.A., Weinberger, K.Q. (eds.) Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, New Orleans, Louisiana, USA, 2–7 February 2018. AAAI Press (2018). <https://www.aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/16466>
19. Hwang, K., Sung, W.: Fixed-point feedforward deep neural network design using weights +1, 0, and -1. In: 2014 IEEE Workshop on Signal Processing Systems, SiPS 2014, Belfast, United Kingdom, 20–22 October 2014, pp. 174–179. IEEE (2014), <https://doi.org/10.1109/SiPS.2014.6986082>
20. Hyvärinen, A., Oja, E.: Independent component analysis: algorithms and applications. *Neural Netw.* **13**(4-5), 411–430 (2000). [https://doi.org/10.1016/S0893-6080\(00\)00026-5](https://doi.org/10.1016/S0893-6080(00)00026-5)
21. Ioffe, S., Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: Bach, F.R., Blei, D.M. (eds.) Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6–11 July 2015. pp. 448–456. JMLR.org (2015). <http://jmlr.org/proceedings/papers/v37/ioffe15.html>
22. Jouppi, N.P., et al.: In-datacenter performance analysis of a tensor processing unit. arXiv [abs/1704.04760](https://arxiv.org/abs/1704.04760) (2017)
23. Kim, Y., Park, E., Yoo, S., Choi, T., Yang, L., Shin, D.: Compression of deep convolutional neural networks for fast and low power mobile applications. arXiv [abs/1511.06530](https://arxiv.org/abs/1511.06530) (2015)

24. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Bartlett, P.L., Pereira, F.C.N., Burges, C.J.C., Bottou, L., Weinberger, K.Q. (eds.) *Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems 2012*. Proceedings of a meeting held 3–6 December 2012, Lake Tahoe, Nevada, United States, pp. 1106–1114. Curran Associates, Inc. (2012). <http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks>
25. Li, G., Li, F., Zhao, T., Cheng, J.: Block convolution: towards memory-efficient inference of large-scale CNNs on FPGA. In: *2018 Design, Automation & Test in Europe Conference & Exhibition, DATE 2018*, Dresden, Germany, 19–23 March 2018, pp. 1163–1166. IEEE (2018)
26. Miyashita, D., Lee, E.H., Murmann, B.: Convolutional neural networks using logarithmic data representation. arXiv [abs/1603.01025](https://arxiv.org/abs/1603.01025) (2016)
27. NVIDIA: 8-bit inference with tensorrt (2017). <http://on-demand.gputechconf.com/gtc/2017/presentation/s7310-8-bit-inference-with-tensorrt.pdf>
28. Park, E., Ahn, J., Yoo, S.: Weighted-entropy-based quantization for deep neural networks. In: *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, Honolulu, HI, USA, 21–26 July 2017, pp. 7197–7205. IEEE Computer Society (2017)
29. Qiu, J., et al.: Going deeper with embedded FPGA platform for convolutional neural network. In: Chen, D., Greene, J.W. (eds.) *Proceedings of the 2016 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, Monterey, CA, USA, 21–23 February 2016, pp. 26–35. ACM (2016)
30. Rastegari, M., Ordonez, V., Redmon, J., Farhadi, A.: XNOR-Net: ImageNet classification using binary convolutional neural networks. In: Leibe, B., Matas, J., Sebe, N., Welling, M. (eds.) *ECCV 2016*. LNCS, vol. 9908, pp. 525–542. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46493-0_32
31. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. arXiv [abs/1409.1556](https://arxiv.org/abs/1409.1556) (2014)
32. Szegedy, C., et al.: Going deeper with convolutions. In: *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015*, Boston, MA, USA, 7–12 June 2015, pp. 1–9. IEEE Computer Society (2015)
33. Wang, P., Cheng, J.: Accelerating convolutional neural networks for mobile applications. In: Hanjalic, A., et al. (eds.) *Proceedings of the 2016 ACM Conference on Multimedia Conference, MM 2016*, Amsterdam, The Netherlands, 15–19 October 2016, pp. 541–545. ACM (2016)
34. Wang, P., Cheng, J.: Fixed-point factorized networks. In: *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, Honolulu, HI, USA, 21–26 July 2017, pp. 3966–3974. IEEE Computer Society (2017)
35. Wang, P., Hu, Q., Fang, Z., Zhao, C., Cheng, J.: DeepSearch: a fast image search framework for mobile devices. *TOMCCAP* **14**(1), 6:1–6:22 (2018)
36. Wu, J., Leng, C., Wang, Y., Hu, Q., Cheng, J.: Quantized convolutional neural networks for mobile devices. In: *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016*, Las Vegas, NV, USA, 27–30 June 2016, pp. 4820–4828. IEEE Computer Society (2016)
37. Yang, Z., et al.: Deep fried convnets. In: *2015 IEEE International Conference on Computer Vision, ICCV 2015*, Santiago, Chile, 7–13 December 2015, pp. 1476–1483. IEEE Computer Society (2015)

38. Zhang, X., Zou, J., Ming, X., He, K., Sun, J.: Efficient and accurate approximations of nonlinear convolutional networks. In: IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, 7–12 June 2015, pp. 1984–1992. IEEE Computer Society (2015)
39. Zhou, A., Yao, A., Guo, Y., Xu, L., Chen, Y.: Incremental network quantization: towards lossless CNNs with low-precision weights. arXiv [abs/1702.03044](https://arxiv.org/abs/1702.03044) (2017)
40. Zhou, S., Ni, Z., Zhou, X., Wen, H., Wu, Y., Zou, Y.: DoReFa-Net: training low bitwidth convolutional neural networks with low bitwidth gradients. arXiv [abs/1606.06160](https://arxiv.org/abs/1606.06160) (2016)