IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received March 21, 2019, accepted April 9, 2019, date of publication April 16, 2019, date of current version April 29, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2911536

Learning Sparse Convolutional Neural Network via Quantization With Low Rank Regularization

XIN LONG[1], ZONGCHENG BEN[2], XIANGRONG ZENG[1], YAN LIU[1], MAOJUN ZHANG[1], AND DIANLE ZHOU[3]

[1]School of Systems Engineering, National University of Defense Technology, Changsha 410073, China
[2]School of Computer, National University of Defense Technology, Changsha 410073, China
[3]School of Intelligent Science, National University of Defense Technology, Changsha 410073, China

Corresponding author: Xiangrong Zeng (zengxrong@gmail.com)

This work was supported in part by the National Nature Science Foundation of China under Grant 61602494 and in part by the NUDT under Grant ZK16-03-16.

ABSTRACT With the refinement of tasks in artificial intelligence, bringing in exponential level increments in computation cost and storage. Therefore, the augment of computation resource for complicated neural networks severely hinders their applications on limited-power devices in recent years. As a result, there is an impending necessity to compress and accelerate the deep networks by special ways. Considering the different peculiarities of weight quantization and sparse regularization, in this paper, we propose a low rank sparse quantization (LRSQ) method to quantize network weights and regularize the corresponding structures at the same time. Our LRSQ can: 1) obtain low-bit quantized networks to reduce memory and computation cost and 2) learn a compact structure from complex convolutional networks for subsequent channel pruning which has significant reduction on FLOPs. In experimental sections, we evaluate the proposed method on several popular models such as VGG-7/16/19 and ResNet-18/34/50, and results show that this method can dramatically reduce parameters and channels of the network with slight inference accuracy loss. Furthermore, we also visualize and analyze the four-dimensional weight tensors, which shows the low rank and group-sparsity structure of it. Finally, we try pruning unimportant channels which are zero-channels in our quantized model, and finding even a little better precision than the standard full-precision network.

INDEX TERMS Convolutional neural network (CNN), weight quantization, spectral regularization, sparsity, visualization, channel pruning.

I. INTRODUCTION

Recently, the world has witnessed a new round of artificial intelligence revolution. Deep neural networks (DNNs), especially convolutional neural networks(CNNs), have made great success in various areas. From handwritten character recognition to image recognition, and burgeoning AI applications [1]–[3]. However, all of these achievements are rely on complex deep neural network models, which contain hundreds of millions of parameters. In the 2012 ILSVRC contest, Krizhevsky et al. constructed a multi-layers network with over 60 million parameters (650 thousand neurons), and this large network has exceeded all previous methods in terms of classification accuracy [4]. But it took more than two days to train the entire network. With the complication of network structure, we could certainly reach a level that traditional methods cannot, while model size (such as almost 50, 200, 250, and 500 MByte for GoogleNet, ResNet-101, AlexNet, and VGG-Net respectively) [5] and computational complexity requirements would increase exponentially. Therefore, embedding these high performances models into smart mobile devices could be a large challenge. There is impending necessity to compress and accelerate the deep networks without affecting the performance. Effective compression methods could have significant impacts on artificial intelligence systems, embedded devices, and mobile devices. Since the huge computation cost of CNNs is mainly dominated by convolutional operation, which is exactly the dot-product

The associate editor coordinating the review of this manuscript and approving it for publication was Guitao Cao.

between weights and activations [5]. Based on this, many studies are performed to compress the scale of CNNs without accuracy loss, including parameter pruning and sharing, low bit quantization, special model architecture designment, low rank approximation of weight and adding sparsity regularization etc. Parameter pruning and sharing approaches, however, often produce non-structured connectivity which may hurt inference accuracy. On the other hand, low rank approximation can indeed obtain compact structure while needing more resources to decompose 4D-tensors. Inspired by the facts that (1) quantization is an efficient method to decrease memory and computation in practical limited-power applications, which has aroused widespread concerns among a large number of scholars; (2) some regularization constraints could get special model structure, such as kernel and channel sparsity model. In this paper, we propose a Low Rank Sparse Quantization (LRSQ) method to learn a compact and sparse structure of deep CNNs by weight quantization with regularization constraint during training process. In fact, our LRSQ method combines weight quantization with spectral relaxation regularization constraint, achieves not only reduce the computation cost and memory but also make it possible for channel pruning. This promises the event-driven hardware design for efficient mobile intelligence.

II. RELATED WORK

Until recent years, methods of neural network compression and acceleration can be roughly divided into the following categories.

A. PARAMETER PRUNING AND SHARING

Han *et al.* [6]–[8] and Liu *et al.* [9] mainly try finding and pruning the redundant parameters and connections in the network model, sharing analogous weights, and keeping important connections with greater weights to achieve the purpose of reducing the storage and computation complexity.

B. LOW RANK APPROXIMATION

Denton *et al.* [10] and Jaderberg *et al.* [11] propose methods to train high-dimensional weight tensors which are expressed as the product of several low-order vectors, and retain significant eigenvalues to obtain key feature information of network so as to reduce the computation cost without performance loss.

C. SPECIAL MODEL ARCHITECTURE

Constraining the model architecture by reducing convolutional layers, removing fully connected layers, using convolutional filters of small size, designing special architecture for different tasks, such as NIN [12], Squeeze-Net [13] and Mobile-Net [14].

D. REGULARIZATION CONSTRAINT

As we all know, Group Lasso is an efficient regularization to learn sparse structures in numerous studies. Kim and Xing [15] and Feng and Darrell [16] used group lasso to regularize the structure of filters in DNNs. Wen *et al.* [17] applied group lasso to regularize multiple DNN structures (filters, channels, filter shapes and layer depth).

E. LOW BIT QUANTIZATION

Another branch of approaches for model compression and acceleration is low bit quantization which means that the weights and activations of deep neural networks are represented by discrete values, decreasing the floating-point calculations in the actual computational process and also reducing the storage space. Courbariaux *et al.* [18] and Li *et al.* [19] groundbreakingly constrain network weights to the binary or ternary space, which remove the multiplication operations. It follows that both weights and activations are mapped into binary space at the same time, i.e. binary neural networks [20], XNOR-Net [21], which directly replace multiply-accumulate operations by binary logic operations. Furthermore, DoReFa-Net [22] not only quantizes weights and activations, but also quantizes gradients to low-bit width floating-point numbers with discrete states in the backward propagation. Recently, Wu *et al.* [23] develop a new method termed as "WAGE" to discretize parameter in both training and inference process, where weights (W), activations (A), gradients (G) and errors (E) among layers are shifted and linearly constrained to low-bit width integers.

F. CHANNEL PRUNING

Compared with parameter pruning and weight quantization, channel pruning removes redundant channels other than parameters from the network. The pivotal issue of channel pruning is to evaluate the importance of weight channels. Li *et al.* [30] measures the importance of channels by just calculating the sum of absolute values of weights, which is simple and ineffective. Hu *et al.* [31] defines the average percentage of zeros (APoZ) to evaluate the activation of neurons, which are considered to be redundant with high values of APoZ. Considering the need of learning precision, [32]–[35] try to train network with sparsity regularizer, and transform the problem of measuring channel importance into optimization problem about reconstruction error of output feature.

III. LOW RANK SPARSE QUANTIZATION

In this work, we propose using, as a regularizer for convolutional layers, the regularization constraint of spectral clustering. We also train the network combined with our weight quantization method at the same time. Assuming a typical DNN with L layers can be summarized as the following forms

$$\begin{aligned} \mathbf{x}_{\mathrm{L}} &= \mathrm{f}_{\mathrm{L}}(\mathbf{x}_{\mathrm{L}-1}\mathbf{W}_{\mathrm{L}} + \mathbf{b}_{\mathrm{L}}), \\ \mathbf{x}_{\mathrm{L}-1} &= \mathrm{f}_{\mathrm{L}-1}(\mathbf{x}_{\mathrm{L}-2}\mathbf{W}_{\mathrm{L}-1} + \mathbf{b}_{\mathrm{L}-1}), \ldots, \\ \mathbf{x}_{1} &= \mathrm{f}_{1}(\mathbf{I}\mathbf{W}_{1} + \mathrm{b}_{1}) \end{aligned} \tag{1}$$

where f_i is the nonlinear activation function, $\mathbf{b}_i$ denotes the set of bias, and $\mathbf{x}_i$ is output activation of each layer, $\mathbf{I}$ means the input image data. Given Loss($\mathbf{W}'$) be the general loss function

between predictions and data labels, while $\mathbf{W}'$ is quantized weight tensor. Thus the final total loss objective function with regularization loss Loss_reg($\mathbf{W}'$) is

$$\min_{\mathbf{W}'} \text{Loss}\left(\mathbf{W}'\right) + \sum_{l=1}^{\text{L}} \text{Loss\_reg}\left(\mathbf{W}'\right) \tag{2}$$

where Loss_reg $\left(\mathbf{W}'\right)$ is the regularization constraint of spectral clustering for each layer. In the following subsections, we first investigate the problem of proposed weight quantization method. Later we will exploit how to implement accurate gradients in the back propagation process. Then spectral clustering regularization is introduced during training process, and a unified framework of low rank sparse quantization method will be presented. Finally, we introduce other types of regularization constraint for comparison in experimental section.

A. SPARSE WEIGHT QUANTIZATION

In this section, we mainly concentrate on sparse weight quantization for networks. Similarity to XNOR-net [21], we represent an L-layer CNN with [$\mathbf{I}$, $\mathbf{W}$, $*$], where $\mathbf{I}$ is the input tensor, $\mathbf{W}$ is the weight tensor, and $*$ represents convolutional operation between $\mathbf{I}$ and $\mathbf{W}$ (In this section, we ignore bias terms of convolutional filters). Furthermore, $\mathbf{I} \in \mathcal{R}^{\text{c}\times \text{w}_{\text{in}} \times \text{h}_{\text{in}}}$, where (c, w_{in}, h_{in}) represents channel, width and height of input data respectively. And for $\mathbf{W} \in \mathcal{R}^{\text{c}_{\text{out}} \times \text{c}_{\text{in}} \times s \times s}$, where ($\text{c}_{\text{out}}$, c_{in}, s, s) represents output channels, input channels and filters size. In convolutional layers, we need to reshape 4-dimensional weight tensor $\mathbf{W}$ into 2-dimensional tensor, i.e., $\mathbf{W}^{\text{c}_{\text{out}} \times \text{c}_{\text{in}} \times s \times s} \longrightarrow \mathbf{W}^{\text{c}_{\text{out}} \times (\text{c}_{\text{in}} \text{ss})}$, and apply our quantization function on it. In back propagation process, we compute the derivatives of quantized compound functions for weights update.

1) WEIGHT QUANTIZATION IN FORWARD PROCESS

In order to constrain weights to low-bit region in CNNs, we introduce a scaling factor vector $\alpha \in \text{R}^{+}$ for every output channels and a low-bit vector $\mathbf{b}$ such that $\mathbf{w} \approx \alpha\mathbf{b}$ (where $\mathbf{w}$ is real-value weight vector for reshaped weight matrix $\mathbf{W}^{\text{c}_{\text{out}} \times (\text{c}_{\text{in}} \text{ss})}$). Obviously, every layer has different c_{out} values of α. By bringing this scaling factor, memory usage and convolutional operation computation would be reduced drastically. Furthermore, channels could be clipped when $\alpha = 0$. According to the least squares method, our goal is to solve the following optimization function.

$$\text{minF}\left(\alpha, \mathbf{b}\right) = \min \|\mathbf{w} - \alpha\mathbf{b}\|^2 \tag{3}$$

By expanding above equation, we have

$$\alpha^*, \mathbf{b}^* = \text{argmin}_{\alpha,\mathbf{b}}\text{F} = \text{argmin}_{\alpha,\mathbf{b}}(\alpha^2\mathbf{b}^{\text{T}}\mathbf{b} - 2\alpha\mathbf{w}^{\text{T}}\mathbf{b} + \mathbf{w}^{\text{T}}\mathbf{w}) \tag{4}$$

Referring to XNOR-Net, we assume $\mathbf{b} \in \{\pm 1\}^n$ (n equals to (c_{in}ss)) which is binary quantization function. Since $\mathbf{b}^{\text{T}}\mathbf{b}$ and $\mathbf{w}^{\text{T}}\mathbf{w}$ are constant at the same time, and α is a positive value, the equation (4) could be simplified into

$$\mathbf{b}^* = \text{argmax}_{\mathbf{b}}\mathbf{w}^{\text{T}}\mathbf{b} \tag{5}$$

Therefore the optimal solution of (5) is $\mathbf{b}^* = \text{sign}(\mathbf{w})$, which is similar to XNOR-Net. Here we can extend the quantization level to k-bits (take the place of sign($\mathbf{w}$) function) by using following expression

$$\text{q}_{\text{k}}\left(\text{w}_i\right) = \frac{\left[2^{\text{k}-1}\left(\text{w}_i + 1\right)\right]}{2^{\text{k}-1}} - 1 \tag{6}$$

where marking $[*]$ refers to round operation in math. We approximate sign function by using q_{k} in (6), which is shown below (and its approximation gradient which referring to equation (12))

In actual training process, we usually constrain quantization function q_{k} to $[-1, 1]$, which do not influence training and inference accuracy. Next, we consider finding optimal scaling factor α and corresponding suitable mapping function q_{k}. We compute the partial derivative of first and second order on equation (4) with respect to α and $\mathbf{b}$

$$\begin{cases} \dfrac{\partial F}{\partial \alpha} = 2\alpha\mathbf{b}^{\text{T}}\mathbf{b} - 2\mathbf{w}^{\text{T}}\mathbf{b} = \mathbf{0} \\ \dfrac{\partial \text{F}^2}{\partial^2 \alpha} = 2\mathbf{b}^{\text{T}}\mathbf{b} \geq \mathbf{0} \end{cases} \tag{7}$$

$$\begin{cases} \dfrac{\partial F}{\partial \mathbf{b}} = 2\alpha\mathbf{b}^{\text{T}} - 2\alpha\mathbf{w}^{\text{T}} = 0 \\ \dfrac{\partial \text{F}^2}{\partial^2 \mathbf{b}} = 2\alpha^2 \geq 0 \end{cases} \tag{8}$$

Solving (7) and (8) respectively, both second order derivatives are always greater than or equal to zero, which satisfy the necessary conditions for the optimal value. Thus, we can obtain the following equation

$$\begin{cases} \alpha^* \in R, & \mathbf{b} = 0 \\ \alpha^* = \dfrac{\mathbf{w}^{\text{T}}\mathbf{w}}{\mathbf{b}^{\text{T}}\mathbf{b}} \geq 0, & \mathbf{b} \neq 0 \end{cases} \tag{9}$$

$$\begin{cases} \mathbf{b}^* \in \text{R}^{\text{n}}, & \alpha = 0 \\ \mathbf{b}^* = \dfrac{\mathbf{w}}{\alpha} \geq 0, & \alpha \neq 0 \end{cases} \tag{10}$$

Discussing the above equation simultaneously, there are obvious four different combined conditions. By kicking out the impossible status, the final optimal result for α is

$$\begin{cases} \alpha^* = 0, & \text{if } \mathbf{b} = 0 \\ \alpha^* = \dfrac{\mathbf{w}^{\text{T}}\mathbf{b}}{\mathbf{b}^{\text{T}}\mathbf{b}}, & \text{if } \mathbf{b} = \text{q}_{\text{k}}(*) \end{cases} \tag{11}$$

Therefore, all weights of one output channel may be pruned when $\alpha = 0$ during subsequent training process. Different from ternary weight networks, we use a scaling factor in each channel here in order to obtain sparse channels which could be pruned in next training process. Specially, the net becomes sparse XNOR-Net when $\text{k} = 1$. In DNNs, the computational complexity is mainly dominated by the convolution operation, and could decrease dramatically after our quantization, as shown in experiment section. In this paper, we conduct experiments mainly aiming at $\text{k} = 1$ in order to get more sparse parameters and channels, which facilitate subsequent channel pruning.

2) BACK PROPAGATION PROCESS

Since we use approximation function q_k applying for weight quantization, it is necessary to compute the compound derivation of $\mathbf{w}$ for compensation of back propagation. In order to obtain the gradient of q_k, we follow the same approach of sign function in [21]

$$\frac{\partial \text{sign}}{\partial r} = r1_{|r|\leq 1} \tag{12}$$

It is worth noting that q_k is not continuous and non-differentiable in discontinuity points, as shown in above section (for $x = \pm 0.5$ in Figure 1), which makes it difficult

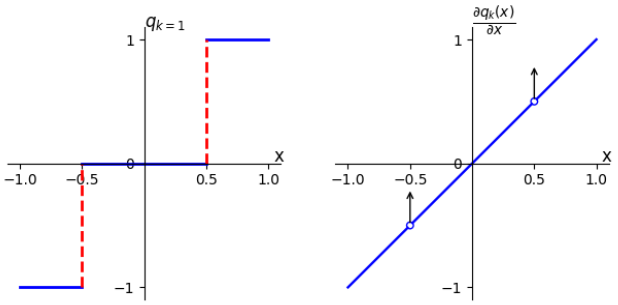

FIGURE 1. Quantization function $q_k(k = 1)$ and its derivates.

for back propagation. To address this issue, we could approximate the partial derivatives of q_k as follows [24] based on (12)

$$\frac{\partial q_k(x)}{\partial x} = \begin{cases} \frac{1}{2a}, & \text{if } r - a \leq |x| \leq r + a \\ w_i, & \text{others} \end{cases} \tag{13}$$

$$\frac{\partial q_k(x)}{\partial x} = \begin{cases} -\frac{1}{a^2}(|x| - (r + a)), & \text{if } r \leq |x| \leq r + a \\ \frac{1}{a^2}(|x| - (r - a)), & \text{if } r - a \leq |x| \leq r \\ w_i, & \text{others} \end{cases} \tag{14}$$

Here r is the discontinuity point of function q_k, and a is a small positive parameter which need to be determined in training process. It is can be seen that when $a \to 0$, the gradient of q_k approaches the function in Fig.1. In this paper, we adopt equation (13) in experimental section. Next we implement the gradient after quantization in backward. Assume the original weights and quantized weights as

$$\mathbf{W} = [\mathbf{w}_1, \mathbf{w}_2 \cdots \mathbf{w}_n] \approx \widetilde{\mathbf{w}} = [\widetilde{\mathbf{w}_1}, \widetilde{\mathbf{w}_2}, \cdots \widetilde{\mathbf{w}_n}] \tag{15}$$

And in forward process we need to quantize weight into

$$\begin{aligned} \widetilde{\mathbf{w}} &= \alpha \mathbf{b} = [\alpha_1 \mathbf{b}_1, \alpha_2 \mathbf{b}_2, \cdots, \alpha_n \mathbf{b}_n] \\ &= [\alpha_1 q_k(\mathbf{w}_1), \alpha_2 q_k(\mathbf{w}_2), \cdots \alpha_n q_k(\mathbf{w}_n)] \end{aligned} \tag{16}$$

where n is the output channel numbers of different layers. Assuming L is the general loss function, and considering interaction between neutrons, the backward gradient after above approximation is

$$\frac{\partial L}{\partial \mathbf{w}_i} = \sum_{m=1}^{n} \left(\frac{\partial L}{\partial \widetilde{\mathbf{w}}_m} \cdot \frac{\partial \widetilde{\mathbf{w}}_m}{\partial \mathbf{w}_i} \right) \tag{17}$$

We ignore $\mathbf{b} = 0$ here, consider compound derivation and obtain

$$\frac{\partial L}{\partial \mathbf{w}_i} = \alpha \cdot \frac{\partial L}{\partial \widetilde{\mathbf{w}}_i} \cdot \frac{\partial \mathbf{b}_i}{\partial \mathbf{w}_i} + \frac{1}{\mathbf{b}_i^T \mathbf{b}_i} [\mathbf{b}_i + \frac{\partial \mathbf{b}_i}{\partial \mathbf{w}_i} (\mathbf{w}_i - 2\alpha \mathbf{b}_i)] \cdot \sum_m \frac{\partial L}{\partial \widetilde{\mathbf{w}}_m} \mathbf{b}_j \tag{18}$$

There are no unknown parameters in (18), which is easy to be implemented programmatically. For $\mathbf{b} = 0$, we ignore above changes, and do not adopt any other special operations. Furthermore, the parameter and learning rate could get updated by stochastic gradient decent (SGD) method with Pytorch in this paper.

B. REGULARIZATION CONSTRAINT OF SPECTRAL CLUSTERING

Clustering is one of most fundamental research topics in both data mining and machine learning communities. It aims to parameter sharing in recent studies, such as [25]–[27], all uses of k-means method. Compression is achieved through weight sharing by only recording important clusters and weight assignment indexes. In order to compatible with weight quantization, we would like re-training on a compact model which encourages the weight to be concentrated tightly around. The optional way is to retrain the quantized network by adding regularization constraint of spectral clustering to loss function. In this section, we mainly combine spectral regularization with our quantization function to obtain low rank model without accuracy loss.

1) BASIC REGULARIZATION EXPRESSION OF SPECTRAL CLUSTERING

Given a set of c-dimensional data vector $\boldsymbol{a}_i (i = 1, 2 \ldots, n)$, we form the c-by-n data matrix $\boldsymbol{A} = [\boldsymbol{a}_1, \ldots, \boldsymbol{a}_n]$. Mathematically, the result of clustering algorithm can be represented by a cluster assignment indication matrix $\boldsymbol{P}_{m \times C}$, such that $\boldsymbol{P}_{ij} = 1$ if data $\boldsymbol{a}_i$ belongs to cluster j, and $\boldsymbol{P}_{ij} = 0$ otherwise. That means there is only one 1 for each row of matrix $\boldsymbol{P}$, and the rest of the elements are all zeros [28]. Generally, a clustering partition of the data vectors can be written in the following form [27]

$$\boldsymbol{AP} = [\boldsymbol{A}_1, \ldots, \boldsymbol{A}_k], \boldsymbol{A}_i = [\boldsymbol{a}_1^{(i)}, \ldots, \boldsymbol{a}_{s_i}^{(i)}] \tag{19}$$

where $\boldsymbol{P}$ is a permutation matrix, and $\mathbf{A}_i$ is m-by-s_i matrix which represents the i-th cluster. For equation (19), we aim to minimize the sum of squared distance between the data points and their corresponding cluster centers.

$$\begin{cases} \min J = \sum_{i=1}^{k} \sum_{s=1}^{s_i} \| \mathbf{a}_s^{(i)} - \mathbf{m}_i \| \\ \text{s.t. } \mathbf{m}_i = \sum_{s=1}^{s_i} \mathbf{a}_s^{(i)} / \mathbf{s}_i \end{cases} \tag{20}$$

where $\boldsymbol{m}_i$ is the mean vector of the data in cluster i. In order to maintain consistency of dimensions with weight quantization above, we assume the weight of a convolutional layer is $\mathbf{W} \in \mathcal{R}^{c_{out} \times c_{in} \times s \times s}$, where ($c_{out}$, c_{in}, s, s) represents out channels, input channels and filters size respectively. We reshape it into a matrix $\mathbf{W} \in \mathbf{W}^{c_{out} \times n}$, where $n = c_{in} \times s \times s$. Each row

vector being a output channel from 4-dimentinal tensors. Considering our quantization method shown above, we could get sparse low rank training models by adding proper regularization, make useful weights being tied more compact, and even obtain zero rows of weight tensor. The spectral relaxation regularization technique was initially introduced in [27] used for weight clustering, extracting important features by singular value decomposition (SVD). According equation (20), the sum of squares cost function for spectral relaxation can be written as

$$\min \left\{ \boldsymbol{trace}\left(\mathbf{W}^{\mathrm{T}}\mathbf{W}\right) - \boldsymbol{trace}(\mathbf{X}^{\mathrm{T}}\mathbf{W}^{\mathrm{T}}\mathbf{W}\mathbf{X}) \right\} \tag{21}$$

where ***trace*** is matrix trace, $\mathbf{W} \in \mathbf{W}^{c_{\mathrm{out}} \times D}$ is reshaped weight tensor and $\mathbf{X}$ is the normalized cluster index matrix used for solving clustering problem actually. In [27], the author let $\mathbf{X}$ be an arbitrary orthogonal matrix and obtain a relaxed maximization problem

$$\begin{cases} \mathrm{J} = \max \{\boldsymbol{trace}(\mathbf{X}^{\mathrm{T}}\mathbf{W}^{\mathrm{T}}\mathbf{W}\mathbf{X})\} \\ \text{s.t. } \mathbf{X}^{\mathrm{T}}\mathbf{X} = \mathbf{I_k} \end{cases} \tag{22}$$

Furthermore, it has a closed-form solution according to Ky Fan theorem. Let λ_{k} be the k-largest eigenvalues of matrix $\mathbf{W}^{\mathrm{T}}\mathbf{W}$, so that $\mathrm{J}^* = \sum_{\mathrm{n}=1}^{\mathrm{k}} \lambda_{\mathrm{n}}$ and $\mathbf{X}^* = [\mathbf{u}_1, \mathbf{u}_2 \cdots \mathbf{u}_{\mathrm{k}}]\mathbf{Q}$ with $\mathbf{u}_{\mathrm{k}}$ is the corresponding eigenvector of λ_{k} and $\mathbf{Q}$ is an arbitrary orthogonal matrix. In training process, we could easily update matrix $\mathbf{X}$ by using the closed-form of (22) in back propagation process and obtain regularization term loss for experimental section.

2) TRAINING ALGORITHMS

Since we have got a sparse low-bit network through weight quantization in section 3.1. To make the whole net more compact, we are motivated to take advantage of (22) as a regularization term on $\mathbf{W}$ during quantization process. It would appear zero rows within reshaped $\mathbf{W} \in \mathbf{W}^{c_{\mathrm{out}} \times D}$ which may be pruned as a channel in experimental work. As shown above, the final loss objective function with regularization loss Loss_reg($\mathbf{W}$) is

$$\min_{\mathrm{W,X}} \mathrm{Loss}\,(\mathbf{W}) + \sum_{l=1}^{\mathrm{L}} \mathrm{Loss\_reg}\,(\mathbf{W}) \tag{23}$$

Here, we adopt the sum of squared distance between the data points and their corresponding cluster centers as regularization term penalties. Furthermore, regularization loss for each layer of the neural network is

$$\begin{cases} \mathrm{Loss\_reg}\,(\mathbf{W}) = \boldsymbol{trace}\left(\mathbf{W}^{\mathrm{T}}\mathbf{W}\right) - \boldsymbol{trace}(\mathbf{X}^{\mathrm{T}}\mathbf{W}^{\mathrm{T}}\mathbf{W}\mathbf{X}) \\ \text{s.t. } \mathbf{X}^{\mathrm{T}}\mathbf{X} = \mathbf{I_k} \end{cases} \tag{24}$$

During the training process, matrix $\mathbf{W}$ and $\mathbf{X}$ need to be updated alternately. Generally, $\mathbf{W}$ is updated using common SGD in back propagation, and $\mathbf{X}$ is updated by composing the eigenvector of the k-largest eigenvalues of matrix $\mathbf{W}^{\mathrm{T}}\mathbf{W}$. To save training memory and reduce the complexity of algorithms, we update $\mathbf{X}$ every three epochs in practical training process other than each iteration. The training algorithm of our LRSQ is summarized in Algorithm 1.

Algorithm 1

Input: An L-layer CNN with [$\mathbf{I}$, $\mathbf{W}$, $*$], current weight $\mathbf{W}^{\mathrm{t}}$ and learning rate η^{t}
$\mathbf{W_0}$ and $\mathbf{X_0}$ initialization
For each epoch in iteration $\mathbf{t}$ do
  Weights quantization: $\mathbf{w_t} \rightarrow \acute{\mathbf{w}}_\mathbf{t} \approx \alpha\mathbf{b}$
  Forward process:
  $\mathbf{x}_1 = \mathbf{f}_1(\mathbf{I}\acute{\mathbf{W}}_\mathbf{1} + \mathbf{b}_1), \ldots, \mathbf{x_L} = \mathbf{f_L}(\mathbf{x_{L-1}}\acute{\mathbf{W}}_\mathbf{L} + \mathbf{b_L})$
  computing $\mathbf{Loss}(\acute{\mathbf{W}}_\mathbf{t}) + \sum \mathbf{Loss\_reg}\left(\acute{\mathbf{W}}_\mathbf{t}\right)$
  Backward process:
  computing $\nabla(Loss(\acute{\mathbf{W}}_\mathbf{t}) + \sum \mathbf{Loss\_reg}\left(\acute{\mathbf{W}}_\mathbf{t}\right))$
  $\mathbf{W_{t+1}}$ = Update the parameters($\acute{\mathbf{W}}_\mathbf{t}$, $\eta_\mathbf{t}$, gradients) using SGD
  $\eta_{\mathbf{t+1}}$ = Update the parameters($\eta_\mathbf{t}$, t, weight decay)
End for
Update $\mathbf{X_t} = \mathbf{V_t}(:, \mathbf{0} : \mathbf{k-1})$ every three epochs
where $\mathbf{W_t^T W_t} = \mathbf{U_t} \sum_\mathbf{t} \mathbf{V_T^T}$ (Using SVD)

C. OTHER TYPESS OF REGULARIZATION CONSTRAINT FOR COMPARISON

In order to show the superiority of our cluster regularization than others, we introduce additional regularization constraints for comparison. In machine learning, the model is too complicate to cause overfit. To avoid overfitting, one of the most common methods is to use regularization, such as l_1 and l_2 norm. In this paper, we compare these two methods with proposed clustering regularization.

As shown above, to maintain dimension consistency with weight quantization, we assuming the weight tensor of a convolutional layer is $\mathbf{W} \in \mathcal{R}^{c_{\mathrm{out}} \times c_{\mathrm{in}} \times s \times s}$, and we reshape it into a 2-dimensional matrix $\acute{\mathbf{W}} \in \mathbf{W}^{c_{\mathrm{out}} \times n}$, where $\mathrm{n} = c_{\mathrm{in}} \times s \times s$. Each row vector being a output channel from 4-dimensional weight tensors. Then the proposed generic optimization of a DNN with l_1 norm regularization can be formulated as

$$\begin{cases} \min_{\mathbf{W}} \mathrm{Loss}\,(\mathbf{W}) + \lambda \sum_{\mathrm{l}=1}^{\mathrm{L}} \mathrm{Loss_r eg}\left(\acute{\mathbf{W}}\right) \\ \text{s.t.Loss\_reg}\left(\acute{\mathbf{W}}\right) = \sum_{\mathrm{i}} \sum_{\mathrm{j}} |\mathrm{w}_{ij}| \end{cases} \tag{25}$$

where w_{ij} is a element in $\acute{\mathbf{W}}$, and L are layers of the DNN. As we all know, l_1 norm is easy to lead sparsity solution, we combine it with our sparse XNOR method for comparison. Furthermore, the proposed optimization with l_2 norm regularization is formulated as

$$\begin{cases} \min_{\mathbf{W}} \mathrm{Loss}\,(\mathbf{W}) + \lambda \sum_{\mathrm{l}=1}^{\mathrm{L}} \mathrm{Loss\_reg}\left(\acute{\mathbf{W}}\right) \\ \text{s.t. Loss\_reg}\left(\acute{\mathbf{W}}\right) = \sum_{\mathrm{i}} \sum_{\mathrm{j}} \mathrm{w}_{ij}^2 \end{cases} \tag{26}$$

which is similar to methods in [15]–[17]. The training algorithm with l_1 norm and l_2 norm is the same as Algorithm 1.

IV. EXPERIMENTS

In this section, we mainly evaluate the effectiveness of our method using common models (VGG-7 [12], ResNet-18) on published datasets (CIFAR-10), and (VGG-16, VGG-19,

TABLE 1. Comparison for different k value.

(%)	k=1	k=2	k=3	FWN
VGG-7	93.56	93.86	93.90	93.61
Resnet-18	94.93	94.91	95.00	95.13
Bits	2	3	4	32

TABLE 2. Comparison with state-of-art algorithms.

(%)	FWN	BWN	XNOR	TWN	OURS
VGG-7	93.61	91.73	93.37	92.56	93.56
Resnet-18	95.13	-	94.61	-	94.93
Bits	32	1	1	2	2

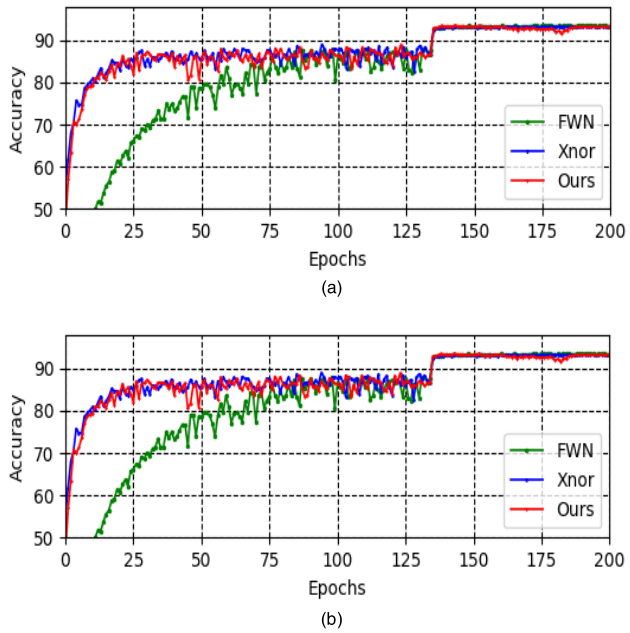

FIGURE 2. Training curves on VGG-7 and ResNet-18. (a) VGG. (b) ResNet.

ResNet-34 and ResNet-50) on (CIFAR-100) for image classification by Pytorch. Most previous works do not quantize the first and last layers. In our method, we also adopt the same strategy and report the averaged results over three runs for each experiment by SGD optimizer. Firstly, we make comparison with existed methods for weight quantization. Then learning the effects of adding spectral regularization constraint. Finally, we also attempt to prune zero output channels of each layers and retrain the network. We evaluate the effect of the different regularizers using the following quantities: parameter sparsity = (#zero parameters) / (#total parameters), channel sparsity = (#zero channels) / (#total channels), compression rate = (#full precision memory) / (#memory after quantization). Our source code is available at Github.

A. PERFORMANCE COMPARISON FOR WEIGHT QUANTIZATION

CIFAR-10: In this section, we compare inference accuracy under the same conditions which only quantize weight not activations. The network structure for "VGG-7" is "2(128-C3) + MP2 + 2(256-C3) + MP2 + 2(512-C3) + MP2 + 2(1024-FC) + Softmax", and "ResNet-18" is standard ResNet-18. The learning rate starts at 0.1 and we use the learning rate decay equal to 0.1 at epochs number 60, 120 and 160 for the whole 200 epochs. For our quantization function, we make comparison for k = 1, 2, 3. The experimental results are shown in Table 1. Interestingly, the performance of k = 2, 3 is better than full precision in VGG-7. Furthermore, Table 2 shows the comparison result of other mainstream methods, where "FWN" represents full precision without any quantization, "BWN" is binary weight networks [12], "TWN" is ternary weight networks [10] and "XNOR" refers to XNOR-Net [11]. Specially, "OURs" refers to k = 1 (the following experiments keep the same k).

It could be seen that the proposed method achieves almost equal accuracy compared with full precision, and has outperform a little more than other several state-of-art algorithms. Fig. 2 presents the training curve comparison among FWN, XNOR and our method on CIFAR-10. It can be seen that our method and XNOR achieve more rapid convergence than full precision, and minimal difference in accuracy.

TABLE 3. Comparison for different networks on CIFAR-100.

top1/top5(%)	FWN	XNOR	OURS
VGG-16	73.86/91.61	68.63/89.15	70.15/90.29
VGG-19	72.85/91.18	67.84/87.85	70.69/90.30
ResNet-34	78.98/94.31	73.87/92.11	77.14/93.82
ResNet-50	79.06/94.84	74.30/92.27	76.89/93.72

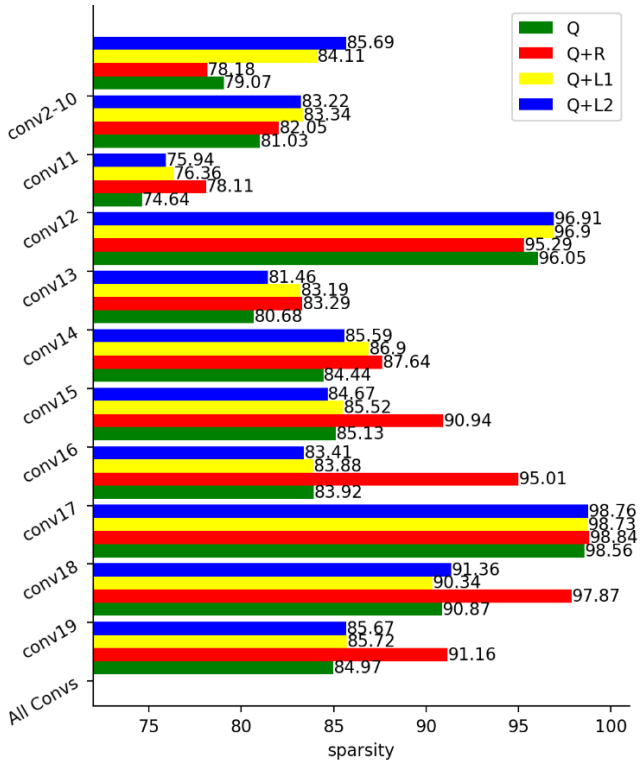

FIGURE 3. Parameter sparsity on ResNet-18.

TABLE 4. Output channel sparsity by regularization.

Layers (channels)	Q	Q+L1	Q+L2	Q+R
conv2(64)	4	6	5	**10**
conv3(64)	6	**8**	4	2
conv4(64)	7	7	**17**	12
conv5(64)	2	**3**	**3**	0
conv6(128)	0	2	4	**9**
conv7(128)	5	1	**11**	6
conv8(128)	10	21	16	**32**
conv9(128)	19	**23**	18	**23**
conv10(128)	1	0	2	**5**
conv11(256)	9	7	11	**25**
conv12(256)	3	8	3	**9**
conv13(256)	7	14	8	**28**
conv14(256)	34	41	34	**51**
conv15(256)	2	3	0	**4**
conv16(512)	3	6	2	**18**
conv17(512)	5	6	5	**50**
conv18(512)	60	61	56	**172**
conv19(512)	7	1	9	**179**
Total	184	218	208	**635**
Accuracy(%)	94.93	94.64	94.50	94.89

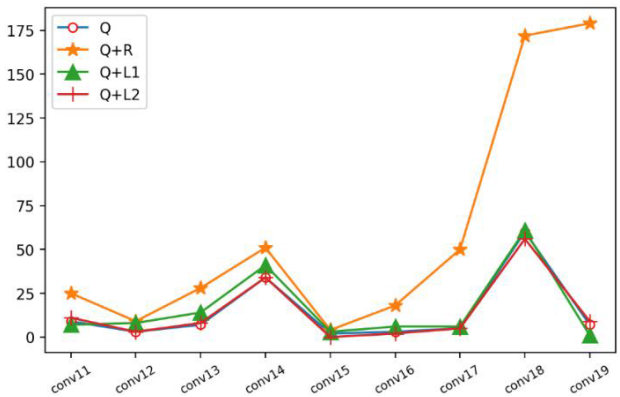

FIGURE 4. Channel sparsity on ResNet-18.

CIFAR-100: To demonstrate the effectiveness of our method, we conduct our experiments on more complicated datasets (CIFAR-100). We also use more popular network like VGG-16/VGG-19 and ResNet-34/ResNet-50 to train here. The learning rate starts at 0.1 and we use the learning rate decay equal to 0.2 at epochs number 60, 120 and 160 for the whole 200 epochs.

Considering the best performance among BWN, XNOR and TWN, we mainly compared XNOR with OURS on CIFAR-100 by different network. The results are shown in Table 3, which shows that our method is much better than XNOR and closes to FWN. Obviously, our method performs better in complex data sets.

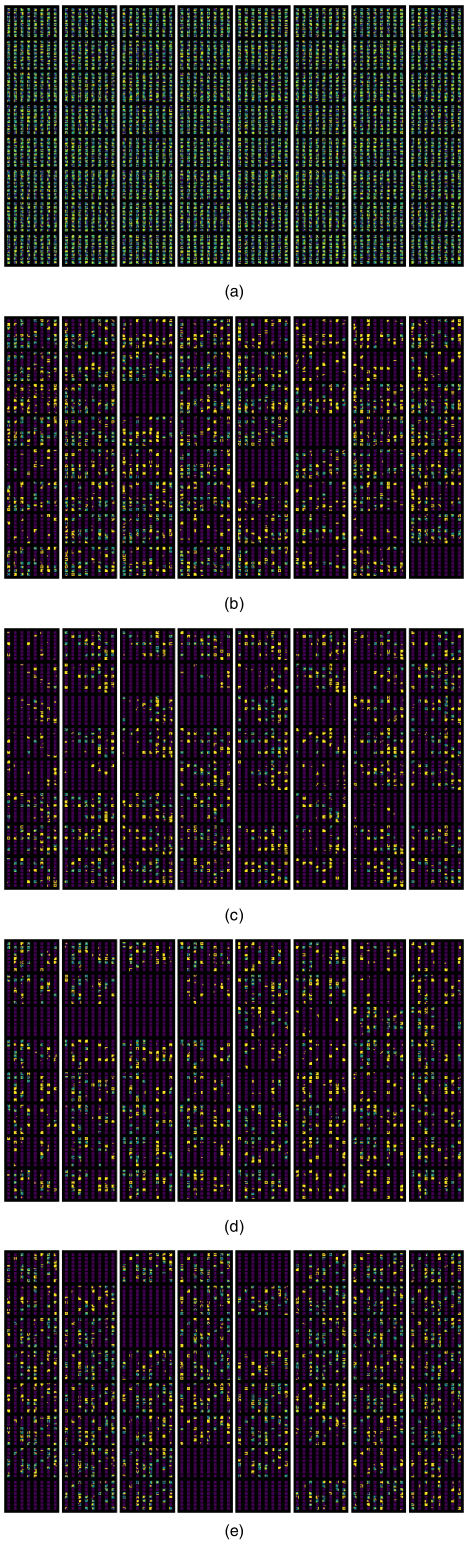

FIGURE 5. Visualization results of 4-dimensional weight tensor. (a) Conv2 in full precision. (b) Conv2 in Q. (c) Conv2 in Q + L1. (d) Conv2 in Q + L2. (e) Conv2 in Q + R.

B. EFFECTS WITH SPECTRAL REGULARIZATION CONSTRAINT

We explore the effects of spectral regularization on ResNet-18 for example in this section (more results about

TABLE 5. Required memory (MByte) and compression rate (CR) for VGG-7 and ResNet-18.

		FULL	Q	Q+R
VGG-7	(MByte)	16.88	0.20	0.12
	(CR)		84×	141×
ResNet-18	(MByte)	67.12	0.82	0.56
	(CR)		82×	120×

TABLE 6. Test error and FLOPs after different channels pruning strategy.

	FULL	Q	Q+R
Test Error	4.87%	4.50%	4.54%
FLOPs	10.44×10^8	9.66×10^8	8.27×10^8

TABLE 7. channel sparsity on CIFAR-100.

VGG-16		
Layers(channels)	Q	Q+R
conv2(64)	6	**16**
conv3(128)	2	**21**
conv4(128)	8	**37**
conv5(256)	10	**51**
conv6(256)	0	**34**
conv7(256)	2	**41**
conv8(512)	0	**77**
conv9(512)	0	**73**
conv10(512)	0	**45**
conv11(512)	0	**68**
conv12(512)	0	**121**
Total	28	**584**
Accuracy(%)	**70.15/90.29**	68.98/89.54

other network will show in appendix). As explain above, a more compact and sparse structure is achieved after adding weight regularization to our quantization loss function. Fig.3 shows the network parameter sparsity of each convolutional layer, where "Q" is by our quantization only, "Q + R" is quantization with spectral regularization, "Q + L1" is quantization with l_1 norm, and "Q + L2" is quantization with l_2 norm. It could be seen that our LRSQ obtain greater sparsity than other methods in back convolution layers with large amount of parameters, and the sparseness promotion of the L1 norm is also reflected in the following figure. Table 4 and Fig.3 counts the zero rows of the reshaped 4D tensors, which means that weights of one output channel are all quantized into zero.

Evidently, our quantization method has greatly promoted the parameter sparsity. Furthermore, adding regularizations make it more obvious, especially in layers with more parameters, while do not have accuracy loss. For output channel sparsity, there will have more zero output channels by LRSQ

TABLE 8. channel sparsity on CIFAR-100.

VGG-19		
Layers(channels)	Q	Q+R
conv2(64)	4	**8**
conv3(128)	1	**12**
conv4(128)	10	**26**
conv5(256)	4	**34**
conv6(256)	11	**43**
conv7(256)	0	**29**
conv8(256)	0	**35**
conv9(512)	1	**128**
conv10(512)	0	**141**
conv11(512)	0	**230**
conv12(512)	0	**135**
conv13(512)	0	**87**
conv14(512)	0	**76**
conv15(512)	0	**139**
Total	31	**1123**
Accuracy(%)	**70.69/90.30**	70.01/89.62

than other several methods with l_1 norm and l_2 norm as shown in Fig.4 and Table 4, which represents more compact structure for the network and provides possibility for channel pruning. We also show the visualization result of convolutional kernel on ResNet-18 in Fig 5 (black pixels mean that parameter value equals to zero).

C. MEMORY COMPRESSION AND FLOPs REDUCTION

We have already explained the sparsity of parameters and channels in above experimental section, and considering quantization effects of net weight, the required memory and computational float-point operations (FLOPs) for CNNs could be decreased greatly. For parameter memory, we use Huffman coding in [6] to save weight parameters. The general coding formula can be expressed as

$$\text{Huffman}_{\text{length}} = -\text{p} * \log_2 \text{p} \tag{27}$$

where p is the occurrence probability of unique element. We calculate the memory consumption of the whole network except the first and last layer, results are shown in Table 5. Our LRSQ could compress the net from 80× to 120 × (140×) by adding regularization constraint. Our quantization network is so small that can be easily fitted into portable devices theoretically. The key compression factor of our method is using scaling factor each channel and obtain zero channel with low rank regularization.

To compute the number of FLOPs, we assume convolution is implemented as a sliding window and nonlinearity is computed for free. For convolutional layers we have [29]

$$\text{FLOPs} = 2\text{HW}(\text{C}_{\text{in}}\text{s}^2+1)\text{C}_{\text{out}} \tag{28}$$

where H and W are weight and width of the input feature map. However, bias is ignored in our experiments, which simplify

TABLE 9. channel sparsity on CIFAR-100.

ResNet-34		
Layers(channels)	Q	Q+R
conv2(64)	5	**21**
conv3(64)	6	**13**
conv4(64)	7	**19**
conv5(64)	2	**9**
conv6(64)	11	**24**
conv7(64)	1	**6**
conv8(128)	1	**8**
conv9(128)	2	**6**
conv10(128)	3	**9**
conv11(128)	20	**41**
conv12(128)	2	**4**
conv13(128)	21	**52**
conv14(128)	1	**3**
conv15(128)	24	**57**
conv16(128)	1	**2**
conv17(256)	4	**36**
conv18(256)	16	**25**
conv19(256)	17	**25**
conv20(256)	31	**92**
conv21(256)	13	**21**
conv22(256)	27	**112**
conv23(256)	9	**16**
conv24(256)	42	**116**
conv25(256)	8	**15**
conv26(256)	46	**120**
conv27(256)	6	**13**
conv28(256)	28	**120**
conv29(256)	3	**8**
conv30(512)	0	**48**
conv31(512)	24	**47**
conv32(512)	25	**47**
conv33(512)	54	**325**
conv34(512)	7	**27**
conv35(512)	0	**65**
Total	521	**1672**
Accuracy(%)	77.14/93.82	**77.40/93.82**

above computation. In this section, we only compute FLOPs of ResNet-18 except the first and last layer. The results are summarized in Table 6.

D. CHANNELS PRUNING

As shown above, there exists plenty of redundancy in ResNet-18 for image classification, and zero channels are

TABLE 10. Channel sparsity on CIFAR-100.

ResNet-50		
Layers(channels)	Q	Q+R
conv2(64)	52	**64**
conv3(64)	55	**64**
conv4(256)	194	**256**
conv5(256)	131	**145**
conv6(64)	47	**55**
conv7(64)	45	**53**
conv8(256)	161	**186**
conv9(64)	30	**36**
conv10(64)	43	**45**
conv11(256)	137	**126**
conv12(128)	78	**86**
conv13(128)	65	**69**
conv14(512)	**285**	280
conv15(512)	231	**253**
conv16(128)	53	**73**
conv17(128)	91	**98**
conv18(512)	**266**	252
conv19(128)	34	**59**
conv20(128)	76	**90**
conv21(512)	**257**	**257**
conv22(128)	35	**50**
conv23(128)	57	**67**
conv24(512)	**211**	204
conv25(256)	27	**40**
conv26(256)	46	**61**
conv27(1024)	225	**309**
conv28(1024)	187	**288**
conv29(256)	38	**81**
conv30(256)	73	**99**
conv31(1024)	210	**282**
conv32(256)	27	**74**
conv33(256)	37	**94**
conv34(1024)	228	**269**
conv35(256)	23	**55**
conv36(256)	71	**96**
conv37(1024)	**221**	211
conv38(256)	15	**67**
conv39(256)	98	**137**
conv40(1024)	**235**	207
conv41(256)	44	**85**
conv42(256)	141	**159**
conv43(1024)	197	**201**
conv44(512)	64	**141**

TABLE 10. ***(Continued.)*** **Channel sparsity on CIFAR-100.**

conv45(512)	57	**104**
conv46(2048)	714	**735**
conv47(2048)	697	**719**
conv48(512)	38	**167**
conv49(512)	87	**226**
conv50(2048)	887	**1079**
conv51(512)	23	**135**
conv52(512)	31	**137**
Total	7375	**9126**
Accuracy(%)	**76.89/93.72**	76.55/94.06

ubiquitous after our quantization with spectral regularization. To further compress the model, we attempt to investigate the influence of channels pruning on ResNet-18, retrain the full precision network which zero channels are clipped. The results are summarized in Table 6. Evidently, inferring accuracy improving may occur when pruning some unimportant channels. Moreover, fewer channels means less inference time and memory in hardware applications. In view of this, we conduct FLOPs comparison according to section C, comparison results are also shown below. Obviously, our LRSQ have achieved great performance, which reducing nearly 21% FLOPs without accuracy loss. From the level of network interpretability, those zero channels are redundant ingredients for training process which could be clipped in practical applications

V. CONCLUSION AND FUTURE WORK

In deep convolutional networks, the parameter size is key factor that directly affects the learning performance. Models compression aims to reduce the redundancy of complex models and accelerate training and inference process. Therefore, we introduce a Low Rank Sparse Quantization (LRSQ) method to achieve the goal, and learn a compact and sparse structure of deep CNN during training process. We find that network accuracy drop slightly by our method. Interestingly, our quantified model adding spectral clustering regularization has obtained compact structure with evident group sparsity, and zero channels of weight tensor which could be pruned. Experiments show that model memory and FLOPs consumption also decrease obviously. In future work, we wish our work may be implemented on hardware applications.

APPENDIX

Various networks have been proposed to compute the channel sparsity on CIFAR-100, which demonstrates the effectiveness of our method for channel sparsity. It can be seen that the phenomenon of channel sparsity on VGG is more obvious than ResNet, where "Q" is by our quantization without any regularization, "Q + R" denotes quantization with spectral regularization.

REFERENCES

[1] H. Geoffrey, S. Nitsh, and S. Kevin, "Neural networks for machine learning," Coursera, Video Lectures, Tech. Rep. 264, 2012.

[2] D. Bahdanau, K. Cho, and Y. Bengio. (2014). "Neural machine translation by jointly learning to align and translate." [Online]. Available: https://arxiv.org/abs/1409.0473

[3] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *Proc. IEEE Int. Conf. Comput. Vis.*, Dec. 2015, pp. 1026–1034.

[4] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. Advance Neural Inf. Process. Syst.*, 2012, pp. 1097–1105.

[5] Z. Cai, X. He, J. Sun, and N. Vasconcelos, "Deep learning with low precision by half-wave Gaussian quantization," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jul. 2017, pp. 5406–5414.

[6] S. Han, H. Mao, and W. J. Dally. (2015). "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding." [Online]. Available: https://arxiv.org/abs/1510.00149

[7] S. Han *et al.*, "EIE: Efficient inference engine on compressed deep neural network," in *Proc. 43rd Annu. Int. Symp. Comput. Archit.*, Jun. 2016, pp. 243–254.

[8] S. Han *et al.*, "Deep compression and EIE: Efficient inference engine on compressed deep neural network," in *Proc. Hot Chips Symp.*, 2016, pp. 1–6.

[9] B. Liu, M. Wang, H. Foroosh, M. Tappen, and M. Penksy, "Sparse convolutional neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2015, pp. 806–814.

[10] E. L. Denton, W. Zaremba, J. Bruna, Y. LeCun, and R. Fergus, "Exploiting linear structure within convolutional networks for efficient evaluation," in *Proc. Adv. Neural Inf. Process. Syst.*, 2014, pp. 1269–1277.

[11] M. Jaderberg, A. Vedaldi, and A. Zisserman. (2014). "Speeding up convolutional neural networks with low rank expansions." [Online]. Available: https://arxiv.org/abs/1405.3866

[12] M. Lin, Q. Chen, and S. Yan. (2013). "Network in network." [Online]. Available: https://arxiv.org/abs/1312.4400

[13] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer. (2016). "SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5MB model size." [Online]. Available: https://arxiv.org/abs/1602.07360

[14] A. G. Howard *et al.* (2017). "MobileNets: Efficient convolutional neural networks for mobile vision applications." [Online]. Available: https://arxiv.org/abs/1704.04861

[15] S. Kim and E. P. Xing, "Tree-guided group lasso for multi-task regression with structured sparsity," in *Proc. 27th Int. Conf. Mach. Learn.*, 2010, p. 1.

[16] J. Feng and T. Darrell, "Learning the structure of deep convolutional networks," in *Proc. IEEE Int. Conf. Comput. Vis.*, Dec. 2015, pp. 2749–2757.

[17] W. Wen, C. Wu, Y. Wang, Y. Chen, and H. Li. (2016). "Learning structured sparsity in deep neural networks." [Online]. Available: https://arxiv.org/abs/1608.03665

[18] M. Courbariaux, Y. Bengio, and J.-P. David, "Binaryconnect: Training deep neural networks with binary weights during propagations," in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, pp. 3123–3131.

[19] F. Li, B. Zhang, and B. Liu. (2016). "Ternary weight networks." [Online]. Available: https://arxiv.org/abs/1605.04711

[20] M. Courbariaux, I. Hubara, D. Soudry, R. El-Yaniv, and Y. Bengio. (2016). "Binarized neural networks: Training deep neural networks with weights and activations constrained to +1 or −1." [Online]. Available: https://arxiv.org/abs/1602.02830

[21] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, "XNOR-Net: ImageNet classification using binary convolutional neural networks," in *Proc. Eur. Conf. Comput. Vis.*, 2016, pp. 525–542.

[22] S. Zhou, Y. Wu, Z. Ni, X. Zhou, H. Wen, and Y. Zou. (2016). "DoReFa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients." [Online]. Available: https://arxiv.org/abs/1606.06160

[23] S. Wu, G. Li, F. Chen, and L. Shi. (2018). "Training and inference with integers in deep neural networks." [Online]. Available: https://arxiv.org/abs/1802.04680

[24] L. Deng, P. Jiao, J. Pei, Z. Wu, and G. Li, "GXNOR-NET: Training deep neural networks with ternary weights and activations without full-precision memory under a unified discretization framework," *Neural Netw.*, vol. 100, pp. 49–58, Apr. 2018.

[25] J. Wu, Y. Wang, Z. Wu, Z. Wang, A. Veeraraghavan, and Y. Lin. (2018). "Deep k-means: Re-training and parameter sharing with harder cluster assignments for compressing deep convolutions." [Online]. Available: https://arxiv.org/abs/1806.09228
[26] M. M. Fard, T. Thonet, and E. Gaussier. (2018). "Deep k-means: Jointly clustering with k-means and learning representations." [Online]. Available: https://arxiv.org/abs/1806.10069
[27] H. Zha, X. He, C. Ding, H. Simon, and M. Gu, "Spectral relaxation for K-means clustering," in *Proc. Int. Conf. Neural Inf. Process. Syst., Natural Synth.* Cambridge, MA, USA: MIT Press, 2001, pp. 1057–1064.
[28] F. Wang, C. Zhang, and T. Li, "Clustering with local and global regularization," *IEEE Trans. Knowl. Data Eng.*, vol. 21, no. 12, pp. 1665–1678, Dec. 2009.
[29] P. Molchanov, S. Tyree, T. Karras, T. Aila, and J. Kautz, "Pruning convolutional neural networks for resource efficient inference," in *Proc. Int. Conf. Learn. Represent.*, 2017, pp. 1–17.
[30] H. Li, A. Kadav, L. Durdanovic, H. Samet, and H. P. Graf, "Pruning filters for efficient convnets," in *Proc. Int. Conf. Learn. Represent.*, 2017, pp. 1–13.
[31] H. Hu, R. Peng, Y.-W. Tai, and C.-K. Tang. (2016). "Network trimming: A data-driven neuron pruning approach towards efficient deep architectures." [Online]. Available: https://arxiv.org/abs/1607.03250
[32] J. M. Alvarez and M. Salzmann, "Learning the number of neurons in deep networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, pp. 2270–2278.
[33] Y. He, X. Zhang, and J. Sun, "Channle pruning for accelerating very deep neural networks," in *Proc. Int. Conf. Comput. Vis.*, 2017, pp. 1389–1397.
[34] Z. Liu, J. Li, Z. Shen, G. Huang, S. Yan, and C. Zhang, "Learning efficient convolutional networks through network slimming,"in *Proc. IEEE Int. Conf. Comput. Vis.*, Oct. 2017, pp. 2755–2763.
[35] J. Luo, J. Wu, and W. Lin, "ThiNet: A filter level pruning method for deep neural network compression," in *Proc. IEEE Int. Conf. Comput. Vis.*, Oct. 2017, pp. 5058–5066.

XIN LONG received the B.S. degree in management science from Beijing Normal University, Beijing, China, in 2014, and the M.S. degree in management science from the National University of Defense Technology, Changsha, Hunan, China, in 2017, where he is currently pursuing the Ph.D. degree with the School of Systems Engineering. His current research interests include deep learning, computer vision, and image processing.

ZONGCHENG BEN received the B.S. degree in information management and information system from Guangxi University, Nanning, Guangxi, China, in 2013. He is currently pursuing the M.S. degree with the School of Computer, National University of Defense Technology. His current research interests include deep learning, computer vision, and video object segmentation.

XIANGRONG ZENG received the B.S. degree in management engineering and the M.S. degree in system engineering from the National University of Defense Technology, Changsha, Hunan, China, in 2008 and 2010, respectively, and the Ph.D. degree in electrical and computer engineering from the Instituto Superior Tecnico, Universidade de Lisboa, Portugal, in 2015. He is currently an Assistant Professor with the School of Systems Engineering, National University of Defense Technology. His current research interests include computational photography, and convex optimization methods in signal and image processing.

YAN LIU received the B.S. degree in applied mathematics from Northeastern University at Qinhuangdao, Qinhuangdao, Hebei, China, in 2013, and the M.S. degree in management science from the National University of Defense Technology, Changsha, Hunan, China, in 2017, where he is currently pursuing the Ph.D. degree with the School of Systems Engineering. His current research interests include deep learning, computer vision, and image processing.

MAOJUN ZHANG received the B.S. and Ph.D. degrees in system engineering from the National University of Defense Technology, Changsha, China, in 1992 and 1997, respectively, where he is currently a Professor with the Department of System Engineering. His current research interests include computer vision, information system engineering, and system simulation.

DIANLE ZHOU received the B.S. degree in control engineering and science and the M.S. degree in instrument science and technology from Beihang University, Beijing, China, in 2004 and 2006, respectively, and the Ph.D. degree in instrument science and technology from Télécom SudParis, Paris, in 2011. He is currently an Assistant Professor with the School of Intelligent Science, National University of Defense Technology, Changsha, China. His current research interests include computational photography, signal and image processing, and face recognition.