Received September 14, 2019, accepted October 9, 2019, date of publication October 16, 2019, date of current version October 29, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2947846

Compressing by Learning in a Low-Rank and Sparse Decomposition Form

KAILING GUO, *, (Member, IEEE), XIAONA XIE*, XIANGMIN XU, (Senior Member, IEEE), AND XIAOFEN XING
School of Electronic and Information Engineering, South China University of Technology, Guangzhou 510000, China

Corresponding author: Xiangmin Xu (xmxu@scut.edu.cn)

*Kailing Guo and Xiaona Xie contributed equally to this work.

This work was supported in part by the National Natural Science Foundation of China under Grant U1801262, Grant U1636218, Grant 61702192, and Grant 61802131, in part by the China Postdoctoral Science Foundation under Grant 2018M630944, in part by the Natural Science Foundation of Guangdong Province, China, under Grant 2018A030313474, and in part by the Fundamental Research Funds for the Central Universities, under Grant 2018MS79, Grant 2019PY21, and Grant 2019MS028.

ABSTRACT Low-rankness and sparsity are often used to guide the compression of convolutional neural networks (CNNs) separately. Since they capture global and local structure of a matrix respectively, we combine these two complementary properties together to pursue better network compression performance. Most existing low-rank or sparse compression methods compress the networks by approximating pre-trained models. However, the optimal solutions to pre-trained models may not be optimal to compressed networks with low-rank or sparse constraints. In this paper, we propose a low-rank and sparse learning framework that trains the compressed network from scratch. Our compressing process can be described as the following three stages. (a) In the structure designing stage, we decompose a weight matrix into sum of low-rank matrix and sparse matrix, and then the low-rank matrix is further factorized into product of two small matrices. (b) In training stage, we add ℓ_1 regularization to the loss function to force the sparse matrix to be sparse. (c) In the post-processing stage, we remove the unimportant connection of sparse matrix according to its energy distribution. The pruning process in the post-processing stage reserves most of capacity of the network and keeps the performance of the network to a great extent. The performance can be further improved with fine-tuning, along with sparse masked convolution. Experiments on several common datasets demonstrate our model is superior to other network compression methods based on low-rankness or sparsity. On CIFAR-10, our method compresses VGGNet-19 to 3.14% and PreActResNet-56 to 29.78% without accuracy drop. 62.43% of parameters of ResNet-50 are reduced with 0.55% top-5 accuracy loss on ImageNet.

INDEX TERMS Convolutional neural networks, low-rank, sparse, network compression.

I. INTRODUCTION

Convolution Neural Networks (CNNs) date back to the 1980s and become popular after the proposition of AlexNet [1] because of its unprecedented result on ImageNet [2]. Since then, CNNs have made extraordinary achievements in a lot of applications, e.g., image classification [3], [4], object detection [5], [6], and natural language processing [7]–[9], just to name a few. A noticeable trend in this area is that deeper and wider network architectures are designed constantly to pursue better performance.

Although CNNs have outstanding performance on many tasks, parameter increment brought by deeper and wider network architecture makes it difficult to employ them in daily life. For example, the number of VGG-19's parameters [3] reaches 143 millions, which hinders its deployment on embedded devices and mobile phones seriously. The contraction between network size and real-time application attracts increasing research attention on compressing and accelerating CNNs.

Low-rank approximation and pruning are popularly used for compressing CNNs. Low-rankness captures the global property of a matrix and is widely used for removing redundancy in high-dimensional data [10]. A low-rank matrix can be factorized into sequence of small matrices, and thus the corresponding memory storage and matrix computation can be reduced. Low-rank approximation is utilized to compress pre-trained networks under different reconstruction guidance,

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

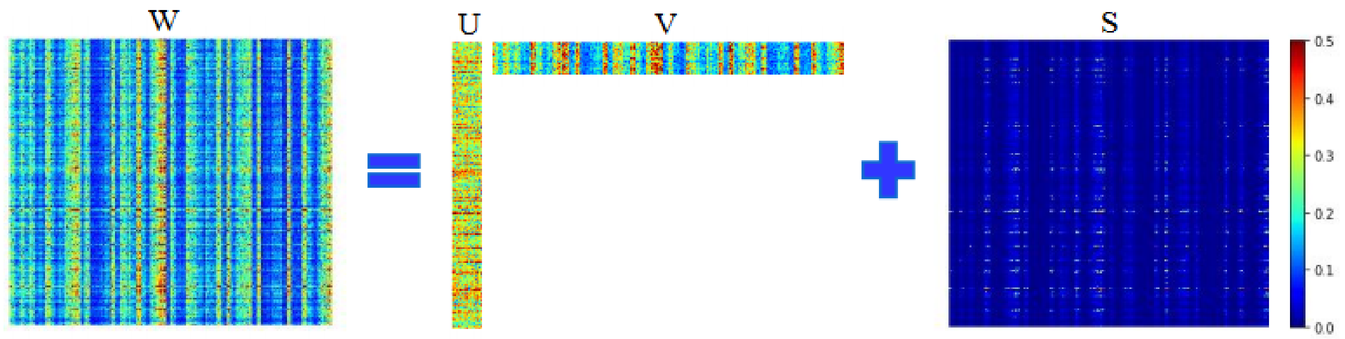


FIGURE 1. **Visualization of weight matrix decomposition. These matrix parameters are extract from convolutional layer conv4_2 from compressed VGGNet-19. For better visualization, we take the first 64 channels of each convolution kernel in the matrix V and S, and the matrix U is entire. We also use the power function $f(x) = x^{0.25}$ on all matrices to amplify small parameters near 0, which highlights the sparse structure of the matrix S.**

e.g., layer-by-layer [11], in a global perspective [12], and knowledge transfer [13]. Pruning is an effective compression strategy that removes unimportant weights to obtain sparse matrix, and thus reduce storage and avoid unnecessary computation. The pioneering work of Han et al. [14] showed encouraging compression rate without loss of accuracy by simply removing weights with small magnitude and combining with fine-tuning. Recently, new pruning criteria from different aspects, e.g., reconstruction error [15], [16], neuron importance score propagation (NISP) [17], and discriminative information [18], are proposed to measure the importance of neurons and to improve the performance of compressed network. Since low-rank decomposition and pruning actually lead to new structure of network, compressing network by heuristically approximating a pre-trained model may not be optimal choice for the new compressed structure. There is another trend in model compression that changes approximation problem into optimization problem with regularization or constraint. For example, low-rankness is obtained from optimizing nuclear norm regularized problem [19] and pruning is applied after optimizing sparse constraint problems [20], [21]. These optimization-based methods learn the compressed structures dynamically through training rather than approximating pre-trained networks.

Most existing works study low-rank decomposition and pruning for network compression separately. Since they describe global and local property of a matrix respectively, they are complementary and can be combined together for better performance. Both [19] and [22] verifies this idea. However, the work [19] forced a matrix to be both low-rank and group sparse, and the learned weight matrix can only be compressed by either low-rank approximation or pruning. The work [22] decomposed a weight matrix into the sum of a low-rank matrix and a sparse matrix, and compressed the network by both strategies. However, the compressed network in [22] was obtained via approximating a pre-trained original network and needed iterative fine-tuning.

Since the network is optimized to find the optimal parameters of the original architecture in the training stage, compressing a pre-trained model by low-rank decomposition or pruning in the post-processing stage will unavoidably affect its performance significantly. To mitigate this limitation, we propose a low-rank and sparse decomposition (LRSD) framework that compresses network via learning. Following [22], the network architecture is designed by decomposing a weight matrix W into the sum of a low-rank component L and a sparse component S (see Figure 1 for visualization). The low-rank matrix L can be factorized into product of two small matrices U and V according to its rank, resulting in sequenced convolution with less computation and memory cost. Sparsity is a straightforward strategy to reduce storage and the computation corresponding to the sparse elements can be omitted. During training, an ordinary weight matrix is replaced by the mixture of U, V, and S. We add ℓ_1 regularization in the loss function to guarantee the sparsity of S. After training, unimportant weight values of S shrink to zeros and we can remove the unimportant weights to obtain the final sparse matrix. Since S tends to learn sparse structure, pruned S is closer to the learned S compared to methods without ℓ_1 regularization. Thus, the proposed compressed method works well without fine-tuning. Its performance can be further improved with a few epochs of fine-tuning.

To demonstrate the effect of our proposed method, we conduct experiments on several popular image classification datasets with some representative network structures. On CIFAR-10, our framework achieves a balance between high compression rate and high accuracy before retraining. On CIFAR-100 and ImageNet, a few epochs of fine-tuning are necessary to retain performance of pruned models. However, compared to other works, our method always has a better performance before retraining.

The main contributions of the proposed method are as follows.

(a) We show that low-rank decomposition and pruning can be combined to compress networks via training from scratch and achieve better performance than its counterpart methods that approximate pre-trained networks.
(b) Compare to methods that setting the sparse rate for pruning explicitly, our sparse rate is adaptively learned in the proposed method by optimization with sparsity regularization.

(c) We show that combining low-rank decomposition with pruning achieves higher compression rate than most existing pruning methods.

II. RELATED WORK

A. COMPRESSION BY LOW-RANKNESS ONLY

Most low-rank methods compress networks by approximating pre-trained networks. Usually they replaced a pre-trained weight matrix/tensor with a low-rank matrix/tensor (factorized into small matrices/tensors) by minimizing weight reconstruction error [23], [24] or output reconstruction error [11]–[13], [23]. Denton *et al.* [24] exploited low-rank decomposition of weight tensors along different dimensions via SVD to reconstruct weight tensors. Jaderberg *et al.* [23] decomposed the filters into rank 1 filters and investigated the effect of minimizing weight reconstruction error and linear response reconstruction error. They show that response reconstruction outperforms weight reconstruction. Rather than approximating linear filters or linear responses, Zhang *et al.* [11] minimized non-linear response error and achieved more precise approximation. Lin *et al.* [13] used low-rank decomposition of a pre-trained network as initialization, and then retrained the low-rank compressed network by knowledge transfer and distillation. Unlike methods mentioned above, Alvarez *et al.* [19] utilized low-rankness from another perspective. They obtained low-rank matrix during training with nuclear norm regularization which encouraged the singular values of a matrix to approach zero. In the post-processing stage, they used SVD-based to decompose constrained matrix and then cut the small singular values to generate two small matrices. They showed that accounting for compression during training led to more compact networks.

B. COMPRESSION BY SPARSITY/PRUNING ONLY

Pruning is another intuitive strategy for compressing networks. LeCun *et al.* [25] set the unimportant parameters of model to zeros according to second derivation, which required large memory and computation costs due to complex calculation of the Hessian matrix. Since then, different pruning criteria have been proposed to evaluate the importance of network parameters. Han *et al.* [14] used magnitude of weight values as pruning criterion and showed that it was able to compress VGG-16 model by 15× with iterative pruning and retraining. Average Percentage of Zeros (APoZ) [26], which refers to the statical property of the output feature map, is utilized to judge the importance of a neuron. ThiNet [16] propose to prune filters based on statistics information computed from its next layer. Molchanov *et al.* [27] proposed a new criterion based on Taylor expansion that approximated the change in the cost function induced by pruning network parameters. Yu *et al.* [17] proposed to compute the importance of each neuron by propagation according to the reconstruction of the final response layer. Intrinsically, they all define the pruning criteria based on certain property of pre-trained networks. Discrimination-aware channel pruning (DCP) [18] adds extra discriminative information in middle layers with sparse regularization to guide the pruning of a pre-trained network. In the contrast, Slimming [28] adds channel sparsity through training the network without relying on pre-trained networks.

C. COMBINING LOW-RANKNESS AND SPARSITY/PRUNING

In the area of network compression, low-rankness and sparsity/pruning are developed almost independently. Since they capture global and local property of network weights, which are complementary, some recent efforts combine them together for network compression. The work [19] combined $\ell_{2,1}$ regularization and nuclear norm for learning weight matrices that were both low-rank and group sparse. They showed that the joint constraint significantly outperformed single low-rank constraint. Greedy Bilateral Decomposition (GreBDec) [22] utilize low-rank and sparse decomposition for network compression, which is similar to the proposed method. However, our approach is different from [22] in the following aspects. Firstly, the low-rank and sparse decomposition framework proposed in [22] compressed a pre-trained network in the post-processing stage, which needed to minimize the reconstruction error, but our method trains a compact network from scratch. Secondly, we force the parameters of S to approach zeros by adding regularization to loss function rather than pruning them based on magnitude directly.

D. COMPRESSION BY OTHER STRATEGIES

Other compression strategies can be categorized into weight quantization, knowledge distillation, and compact architecture design. Weight quantization is to map the values of network parameters from the real number field to finite subset, or to express network parameters in fewer bits. Vector quantization [29] was employed to reduce parameter redundancy of pre-trained networks. In the same trend as low-rankness and pruning, lots of recent quantization methods [30]–[32] proposed to directly learn quantized model parameters instead of approximating pre-trained networks. Knowledge distillation [33] used a teacher model to guide the training of compressed model by adding a distillation loss that penalized the difference between the softmax outputs of the two models. It adds extra guided information, which is beyond the scope of this paper, while we focus on the network structure. Recently, some researchers designed compact modules for compression directly, e.g. depthwise separable convolutions in MobileNet [34] and group convolution with shuffle in ShuffleNet [35]. However, they are designed for specific networks, which are less general for compressing other deep models.

III. PROPOSED METHOD

In this section, we introduce the framework and compression process of our proposed method.

A. FRAMEWORK

We design the compressed network architecture by decomposing the original weight matrix into the sum of low-rank

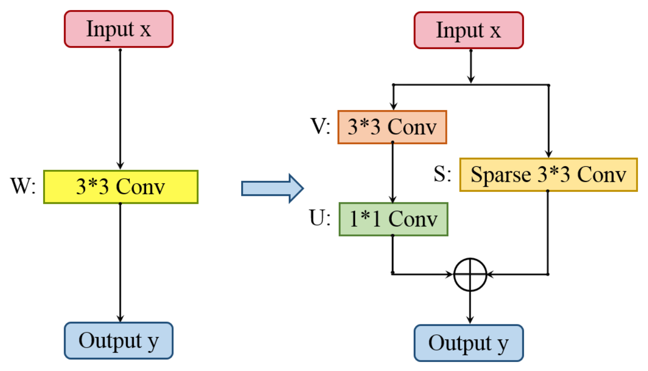


FIGURE 2. New compressed covolutional module.

and sparse matrices. By further factorizing the low-rank matrix into product of small matrices, an ordinary convolution/fully connected layer is replaced by sequenced convolution/fully connected layers adding a sparse convolution/fully connected layer. An example for a new 3 × 3 convolutional module is shown in Figure 2. With a small rank of low-rank matrix and a high sparse ratio of sparse matrix, the new convolutional module will have fewer parameters than the original module. Detailed explanations are given in the following sub-sections.

B. THE LOW-RANK DECOMPOSITION OF MATRIX L

Convolution can be implemented by matrix multiplication after reshaping the input and weight tensor. In this paper, we use a matrix $W \in \mathbb{R}^{K\times m}$ to represent the weight of a convolution or fully connected layer. When W represents the weight of a convolution layer, we have $m = Cd^2$, where C is the input channels and d is the kernel size.

The weight matrix is decomposed into low-rank matrix L and sparse matrix S as follows

$$W = L + S. \tag{1}$$

To model low-rankness, a common way is to utilize nuclear norm [36]. Defining the loss function of CNN is $f(\mathcal{W})$, where $\mathcal{W} = \{W_1, W_2, \cdots, W_l\}$ is the set of l layers of weights. We can reformulate the objective function as

$$\min_{L_i,S_i} f(\mathcal{W}) + \lambda \sum_{i=1}^{l} \|L_i\|_*, \quad s.t. W_i = L_i + S_i, \tag{2}$$

where $\|\cdot\|_*$ represents nuclear norm.

However, optimizing nuclear norm needs to compute SVD at every iteration for all the layers and SVD is computational unfriendly for GPU in existing deep learning frameworks. Therefore, we abandon this constraint method.

Another choice for modeling low-rankness is to factorize the weight matrix beforehand. Suppose the rank of the low-rank matrix L is r, we can factorize it into product of small matrices

$$L = UV, \tag{3}$$

where $U \in \mathbb{R}^{K\times r}$ and $V \in \mathbb{R}^{r\times m}$.

When rank $r \ll min(K, m)$, we have $Kr + rm \ll Km$. Thus, replacing W with U and V will lead to less memory cost. Suppose the input is a vector $\mathbf{x} \in \mathbb{R}^m$, so the computation complexity of $W\mathbf{x}$ and $U(V\mathbf{x})$ are $O(Km)$ and $O(Kr + rm)$, respectively. Thus, the factorization reduces not only the storage but also the computation cost when r is small. Specifically, when $r = 1$, the storage and computation costs reach the minimum. By such factorization, a low-rank fully connected layer can be replaced by two successive small fully connected layers, and a low-rank convolution layer can be replaced by a convolution with the same kernel size but less output channels named V and a 1 × 1 convolution named U (see Figure 2).

r is a hyper-parameter that affects the architecture of the network and makes trade-off between the efficiency and performance. We will discuss how to set the value of r detailedly in Section IV.

C. THE SPARSE STRATEGY OF MATRIX S

Pruning pre-trained model is a common used strategy to obtain sparse weight matrix [26], [27]. However, there is a distinct performance gap between the compressed model and pre-trained model, and fine-tuning is necessary to regain good performance. The effect of pruning will be less if the sparse structure is learned at the training stage [20]. We also obtain the sparse structure via learning. In this way, the number and location of unimportant parameters are automatically chosen, and the pruning of unimportant parameters will result in less accuracy drop or even keep accuracy.

We add regularization to force the matrix to be sparse. ℓ_0 norm counts the number of non-zero elements and is a straightforward sparsity measure. However, optimizing ℓ_0 norm is a NP-hard problem. An alternative strategy is to ℓ_1 norm, which is the optimal convex approximation of ℓ_0 norm. Combining with low-rank approximation, the loss function is reformulated as follows.

$$\min_{L_i,S_i} f(\mathcal{W}) + \lambda \sum_{i=1}^{l} \|S_i\|_1, \quad s.t. W_i = U_iV_i + S_i, \tag{4}$$

where λ is trade-off parameter that balance precision and sparsity.

Stochastic gradient descent (SGD) is popularly used for optimizing deep learning networks. However, ℓ_1 norm is non-smooth and incompatible with SGD. Here, we use sub-gradient instead of gradient. ℓ_1 norm is the sum of absolute value of each element, thus we can compute the sub-derivative of each element independently to obtain the sub-gradient. The sub-derivative of absolute value function is given by

$$\partial\|s\|_1 = \begin{cases} 1, & \text{if } s > 0 \\ [-1, 1], & \text{if } s = 0 \\ -1, & \text{if } s < 0 \end{cases} \tag{5}$$

When applying SGD, we simply set the sub-derivative to zero if s is zero.

D. PRUNING CRITERION

ℓ_1 regularization results more parameters near zero but not exactly zero. We need to prune the unimportant parameters to obtain sparse structure. Previous pruning strategy [14] prunes weights whose magnitude below a threshold across all layers. As we know, each layer plays a different role in a neural network. The same weight magnitude may not be the same important at different layers. Thus, the importance of a neuron cannot be simply judged by its weight magnitude.

In this paper, we derive a new criterion for pruning. To keep the performance, a layer should keep similar statical property after pruning. By reshaping a weight matrix S into a vector $\mathbf{s}$, we define the energy of a weight matrix S as follows.

$$E(\mathbf{s}) = \sum_i |s_i| \,, \tag{6}$$

where s_i denotes the i^{th} element of $\mathbf{s}$. We expect the pruned matrix maintain most energy of the layer with parameters as few as possible. Suppose the element number of $\mathbf{s}$ is n and the energy ratio reserved after pruning is α. Pruning is formulated as the following optimization problem that finds the optimal subset of index.

$$\begin{aligned} &\min \operatorname{card}(I) \\ &s.t. \quad I \subseteq \{1, \ldots, n\}, \quad \sum_{i \in I} |s_i| \geq \alpha E(\mathbf{s}), \end{aligned} \tag{7}$$

where card(I) denotes the element number of set I. This problem can be solved by sorting the entries of $|\mathbf{s}|$ from large to small, and then add the entries one by one until the sum is larger than $\alpha E(\mathbf{s})$. The index of the last entry used in the summing process is the objective value, denoted by k. Then, the weight matrix is pruned by keeping the k largest entries remain the same and setting the others to zeros.

The work [22] approximates pre-trained network by low-rank and sparse decomposition for net compression, but they need to set the sparse rate layer by layer tediously. By utilizing energy ratio as the pruning criterion, the proposed method determines the sparse rate automatically.

Suppose the sparse rate is β and the input vector is $\mathbf{x} \in \mathbb{R}^m$, then the computation complexity of sparse convolution $S\mathbf{x}$ reduces from $O(Km)$ to $O(\beta Km)$. The total computation complexity of the proposed module after pruning is $O(Kr + rm + \beta Km)$. When r and β are small enough, the computation complexity is smaller than the original complexity $O(Km)$.

In general, fine-tuning is an effective way to regain the accuracy of pruned model. When fine-tune the pruned model, we add a binary sparse mask to the sparse matrix S. That is to say, we fix the position of zero parameters of sparse matrix S and only fine-tune the non-zero parameter. For other dense matrices, all parameters are involved.

IV. EXPERIMENTS

In this section, we evaluate the proposed method on several datasets. The experimental results show that our method is superior or comparable to other state-of-the-art methods, including ThiNet [16], Slimming [28], NISP [17], DCP [18], DCP-Adapt [18], and GreBdec [22]. In the following, we use "N_o" and "N_c" to represent the parameter number of original and compressed model respectively, and thus "N_c/N_o" reflects the compression degree of model. Our experiments are implemented on PyTorch [37].

A. DATASETS

1) CIFAR

The CIFAR datasets [38] consisit of CIFAR-10 and CIFAR-100. CIFAR-10 is composed of 60,000 32 × 32 color images in 10 classes. Each class has 6,000 images, 5,000 of which are in the training set and the rest are in the test set. CIFAR-100 is just like the CIFAR-10 except it has 100 classes containing 600 images each. For each class, there are 500 training images and 100 testing images. We adopt the same data augmentation as Slimming [28]. For training, we pad 4 pixels with zeros on each side of a image and then randomly crop a 32 × 32 sample from the padded image. Finally, we horizontally flip the image randomly with probability 0.5. For testing, we directly input the original 32 × 32 image to network. Both training and testing images are normalized with channel means and standard deviations.

2) ImageNet

There are 1,281,167 training images and 50,000 validation images from 1,000 classes in ImageNet dataset [2]. We adopt the default setting of PyTorch for data agmentation. For validation images, we crop out the central 224 × 224 patch.

B. COMPARISON ON SMALL DATASETS

For small datesets CIFAR-10 and CIFAR-100, we carry out compression experiments of VGGNet [28] and PreActResNet [39]. VGGNet is a variation of VGG [3], which reduces fc layers and replaces the last maximum pooling with an average pooling and adds BN layers. For VGGNet, we choose VGGNet-19 and VGGNet-7 to show that LRSD works for both small and large models. For Pre- ActResNet, we choose PreActResNet-56 and PreActResNet-164 for comparison.

1) IMPLEMENT DETAILS

We add a batch normalization layer after layer U in the proposed module (see Figure 2) since we empirically find that it improves the performance on small datasets. We compress all fully connected and 1 × 1 convolutional layers only with sparse constraint. We train all networks from scratch with SGD by using a mini-batch size 64 for 160 epochs on CIFAR datasets. The initial learning rate is set to 0.1, and is divided by 10 at 50% and 75% of the total number of epochs. The weight decay is 1e-4 and momentum of SGD is 0.9. The trade-off parameter λ is set to 2e-6 for CIFAR datasets. Without extra specification, the rank r is set to 1 and the energy ratio α is set to 0.9. We fine-tune the pruned model using the same setting as Slimming [28]. All experiments are repeated for five times and the average results are reported.

TABLE 1. Experimental results of VGGNet-19 and PreActResNet-56 on CIFAR-10.

CIFAR-10		N_c/N_o (%)	Acc. of Pruned Model (%)	Acc. of Fine-tuned Model (%)
VGGNet-19 (Acc.:93.77%, Param.:20.04M)	Slimming	11.23	32.54	93.78
	DCP	28.00	86.30	92.56
	DCP-Adapt	7.28	69.49	89.27
	LRSD	5.11	**93.54**	**93.88**
	LRSD (α=0.7)	**3.14**	91.16	93.78
PreActResNet-56 (Acc.:93.80%, Param.:0.86M)	Slimming	40.08	11.46	91.31
	DCP	50.33	88.29	92.87
	DCP-Adapt	86.73	88.75	93.28
	LRSD	49.75	**93.20**	**94.00**
	LRSD (α=0.7)	**29.78**	85.64	93.89

2) COMPARISON EXPERIMENTS OF LARGE MODELS

We compare LRSD with Slimming [28], and DCP [18], and DCP-Adapt [18]. To avoid potential difference between Torch and PyTorch, we adopt the result of Slimming from the official PyTorch implementation[1] instead of the paper [18]. For PreActResNet-56, we run the PyTorch code to obtain the results. Note that DCP trains the networks with much more epochs than the other methods. For fair comparison, we run the experiments of DCP with the same epochs as the other methods, i.e., 160 epochs, by using its official released code.[2] The pruning rate of DCP is set to 0.5 for all the experiments. Since DCP and DCP-Adapt are very time-consuming, we run their experiments for three times. We set the pruning rate of Slimming according to its default settings [28].

In Table 1, we show the compression ratio and accuracy after pruning and fine-tuning of the compression methods on CIFAR-10. After pruning, LRSD achieve significant higher accuracy than the compared methods. Slimming even can not work without fine-tuning. After fine-tuning, all the methods can be improved but LRSD is still the best and surpass the baseline by 0.11% and 0.2% for VGGNet-19 and PreActResNet-56, respectively. With $\alpha = 0.9$, our compression rate is a little higher than Slimming for PreActResNet-56. However, by setting α to 0.7, LRSD can achieve lower compression rate and the accuracy after pruning and fine-tuning are still much higher than Slimming.

Table 2 summarizes the experimental results on CIFAR-100. For VGGNet-19, LRSD achieves better accuracy but its compression rate is higher than DCP-Adapt by the default setting. We further set α to a smaller value 0.5 to reduce the compression rate of LRSD and show that its accuracy is much better than DCP-Adapt. The accuracy of LRSD with default setting is lower than Slimming. However, we show that when the rank is set to a higher value $K/10$, the accuracy of LRSD is higher than Slimming with lower compression rate. Here $K/10$ means setting the rank to 1/10 of the number of output channels. For PreActResNet-164, LRSD achieves the best accuracy with default setting but its compression rate is higher than DCP. We show that, by setting α to 0.7, LRSD

[1]https://github.com/Eric-mingjie/network-slimming
[2]https://github.com/SCUT-AILab/DCP

TABLE 2. Experimental results of VGGNet-19 and PreActResNet-164 on CIFAR-100.

CIFAR-100		N_c/N_o (%)	Acc. of Pruned Model (%)	Acc. of Fine-tuned Model (%)
VGGNet-19 (Acc.: 72.74%, Param.: 20.08M)	Slimming	24.55	5.31	73.32
	DCP	28.16	58.39	69.79
	DCP-Adapt	4.03	18.26	59.39
	LRSD	10.54	71.18	72.80
	LRSD (r=K/10)	23.21	**72.11**	**73.45**
	LRSD (α=0.5)	**3.65**	23.16	71.91
PreActResNet-164 (Acc.: 77.35%, Param.: 1.73M)	Slimming	73.26	1.87	76.95
	DCP	39.83	63.66	75.24
	DCP-Adapt	80.41	72.59	76.60
	LRSD	52.83	**75.70**	**77.76**
	LRSD (α=0.7)	**32.10**	57.28	77.14

TABLE 3. Experimental results of VGGNet-7 on CIFAR-10 and CIFAR-100.

	N_c/N_o (%)	Acc. of Pruned Model (%)	Acc. of Fine-tuned Model (%)
VGGNet-7 on CIFAR-10 (Acc.: 86.12%, Param.: 0.34M)	23.91	83.28	86.17
VGGNet-7 on CIFAR-100 (Acc.: 65.65%, Param.: 1.08M)	34.78	64.31	66.16

outperforms DCP with lower compression rate and higher accuracy after fine-tuning.

3) COMPRESSION EXPERIMENTS OF SMALLER MODEL

We also use our LRSD algorithm to compress smaller model VGGNet-7. Experimental results are show in Table 3. LRSD reduces more than 60% parameters of VGGNet-7 but remains its performance after fine-tuning on both CIFAR datasets. Experiments demonstrate that our compression algorithm is also robust to small model.

C. COMPRESSION ON IMAGENET

To demonstrate that our approach is not only effective on small datasets, we also implement experiments on large-scale dataset, ImageNet. We carry out experiments with ResNet-50 [4].

1) IMPLEMENT DETAILS

We compress the fully connected layer only with sparse constraint. We train ResNet-50 with the initial learning rate of 0.1 and the weight decay of 1e-4. The learning rate is divided by 10 for every 30 epochs for all networks. The momentum of optimizer SGD is 0.9. The rank is set to 1 and the trade-off parameter λ is set to 1e-6. All networks are trained with a minibatch size 256 for 90 epochs. When fine-tuning, we retrain the pruned model for 20 epochs with a learning rate of 1e-3, and divide learning rate by 10 for every 10 epochs.

2) COMPARISON WITH OTHER METHODS

The comparison results of ResNet-50 on ImageNet are showed in Table 4. For GreBdec, we reimplement it in

TABLE 4. Experimental results of ResNet-50 on ImageNet.

Model	N_c/N_o (%)	Top-1 err.↑ (%)	Top-5 err.↑ (%)
ThiNet	66.27	+0.84	+0.47
NISP	56.18	+0.89	–
DCP	48.55	+1.06	+0.61
GreBdec	57.63	+2.49	+1.31
LRSD ($\alpha = 0.5$)	58.17	**+0.25**	**+0.09**
LRSD-conv1×1 ($\alpha = 0.9$)	**37.57**	+0.97	+0.55

PyTorch. Since the proposed method combines two strategies low-rank and pruning, we compare it with methods that also adopt low-rank or pruning strategies. The bottleneck blocks in ResNet-50 introduce a lot of 1×1 convolutional layers, which have 9.36 million parameters in 1×1 convolutional layer of ResNet-50 and account for 36.62% of the parameters. To study the effect of compressing 1×1 convolutional layers in bottleneck blocks, we conduct experiments of LRSD with and without compressing 1×1 convolutional layers in bottleneck blocks, denoted by LRSD and LRSD-conv1×1, respectively. From Table 4, we can see that LRSD compress about half of the parameters with lowest error increment. The compression rate of DCP is much better than LRSD. However, by taking 1×1 convolutional layers in bottleneck modules into account, LRSD-conv1×1 outperforms DCP both in compression rate and classification errors.

D. PARAMETER ANALYSIS

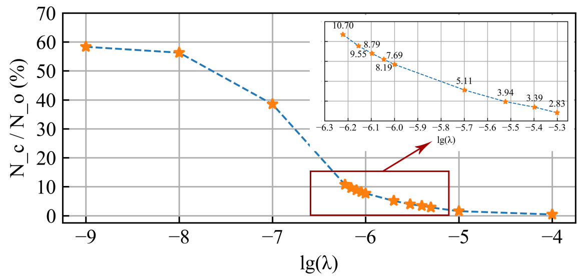
In this section, we analyze the influence of parameters $\{\lambda, r, \alpha\}$ on LRSD. Here we conduct experiments on CIFAR-10. The implementation details are the same as in Sub-Section IV-B.

1) HYPER-PARAMETER λ

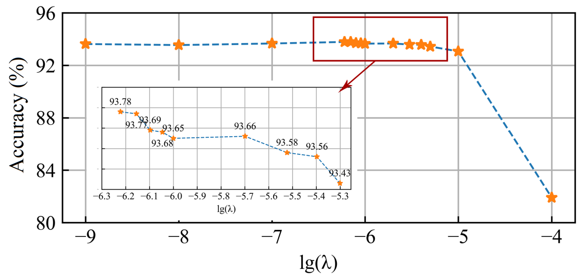
We vary λ while fixing the other parameters. We adjust λ within a large range to find a candidate interval for the optimal λ, and then choose the optimal λ in this candidate interval. Figure 3 shows the compression rate, accuracy before pruning, and accuracy after pruning respectively versus logarithm of λ. The compression rate goes down as λ goes high. The accuracy before and after pruning are stable when λ is small and drop down quickly when λ is larger than 1e-5. Since the accuracy before and after pruning are relatively higher than the others at λ = 1e-6, we further evaluate 8 points (6e-7, 7e-7, 8e-7, 9e-7, 2e-6, 3e-6, 4e-6, 5e-6) around 1e-6. We enlarge the sub-figures for clear visualization. The accuracy after pruning achieves maximum at λ = 2e-6. Thus, we set λ to 2e-6 for our experiments on CIFAR-10. Note that it is a trade-off between compression rate and accuracy. In practice, we can sacrifice some accuracy to pursue compression rate by setting λ to a larger value.

2) HYPER-PARAMETER r

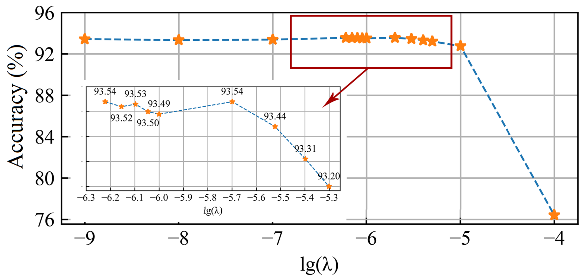
Table 5 shows the experimental results of VGGNet-19 on CIFAR-10 with different rank r. "$r = K/n$" means setting the rank to $1/n$ of the output channels K of the convolutional layer. When r gets smaller, the accuracy before and after pruning are stable, but the number of parameters is significantly reduced. This is due to the sparse matrix S helping keep the ability of model. Therefore, we can always set r to an extremely small value for network compression.



(a) The compression rate



(b) The accuracy before pruning



(c) The accuracy after pruning

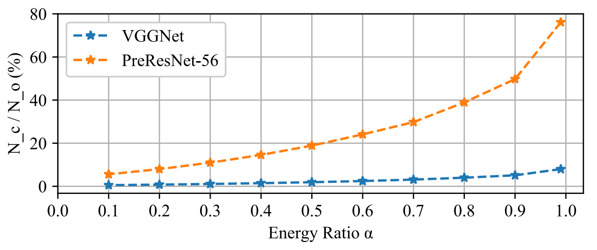
FIGURE 3. Experimental results of VGGNet-19 on CIFAR-10 with varying λ.

TABLE 5. Experimental results of VGGNet-19 on different rank *r* on CIFAR-10.

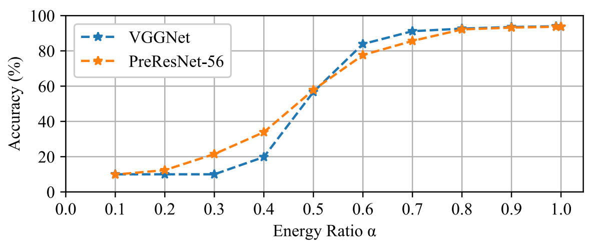
	Acc. of Dense Model (%)	Acc. of Pruned Model (%)	N_c/N_o (%)
Baseline	93.77	–	100
r = K/5	93.83	93.59	27.67
r = K/10	93.72	93.57	16.54
r = K/20	93.72	93.60	11.00
r = 2	93.77	93.51	5.53
r = 1	93.66	93.54	5.11

3) HYPER-PARAMETER α

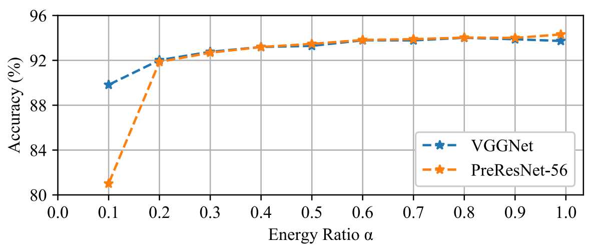
Figure 4 shows the compression rate, accuracy after pruning, and accuracy after fine-tuning respectively versus energy ratio. Note that the points at α = 1 in Figure 4(b) denote



(a) The reserved parameter ratio

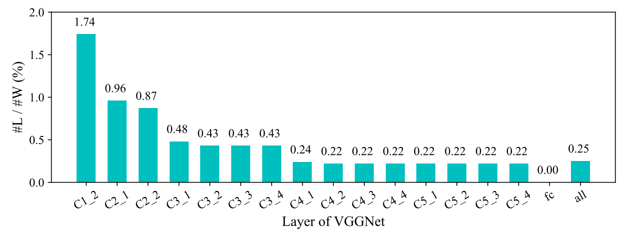


(b) The accuracy after pruning

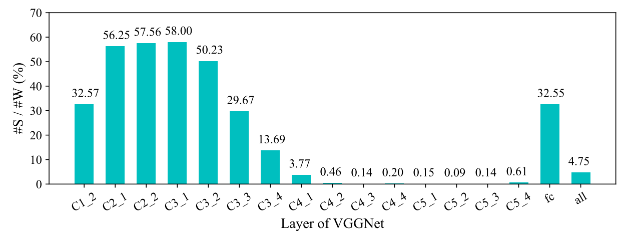


(c) The accuracy after fine-tuning

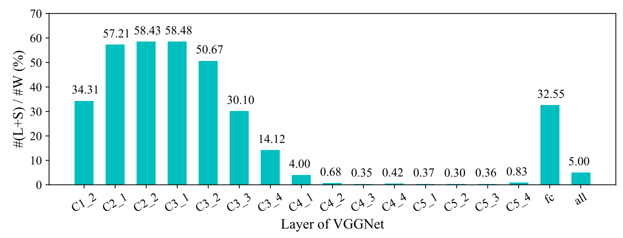
FIGURE 4. Experimental results on different energy ratio α on CIFAR-10.



(a) The parameter ratio of low-rank component to original weight



(b) The parameter ratio of sparse component to original weight



(c) The parameter ratio of both low-rank and sparse component to original weight

FIGURE 5. Compression rate of VGGNet-19 on CIFAR-10. To clearly show the parameter ratio of matrix L and S to W, we do not calculate the parameters of bn and bias in this figure. If take them into account, the total compression parameter ratio should be 5.11%.

TABLE 6. Ablation experimental results of VGGNet-19 on CIFAR-10.

	Parameters	Acc. of Dense Model (%)	Acc. of Pruned Model (%)	Acc. of Fine-tuned Model (%)
LRSD (L)	1.03 M	91.92	–	–
LRSD (S)	1.01 M	**93.71**	93.45	93.81
LRSD	1.02 M	93.66	**93.54**	**93.88**

accuracy before pruning. Even with a very large energy ratio $\alpha = 0.99$, the parameters of VGGNet-19 can be pruned to 7.98% without accuracy drop. This suggests that adding ℓ_1 norm to the loss function is very effective to learn sparse matrices. With shortcut, the residual block is more compact than the VGG block, which makes it harder to compress residual networks. However, LRSD can still cut about 24% of the parameters with very high energy ratio $\alpha = 0.99$. When α is larger than 0.8, the accuracy drop is negligible with 25× compression of VGGNet-19 and 2.6 × compression of PreActResNet-56. When α is less than 0.7, the accuracy of both networks drop sharply. However, the accuracy can be retained by fine-tuning. Even with a low energy ratio $\alpha = 0.1$, the accuracy of VGGNet-19 and PreResNet-56 are still larger than 80% after fine-tuning.

E. COMPRESSION EFFECTS ON DIFFERENT LAYERS

Here we analyze the compression effects of LRSD on different layers. Figure 5 shows the ratio of parameter number of low-rank component and sparse component to parameter number of original weight for different layers of VGGNet-19 on CIFAR-10, respectively. The experiment settings are the same as IV-B. "Ca_b" denotes the b^{th} layer of the a^{th} block. For all convolutional layers, the parameter ratio of low-rank component to original weight $\#L/\#W$ is low since we set the rank to only 1. Since we only compress the fc layer with sparse constraint, the ratio $\#L/\#W$ for fc layer is zero. The ratio of sparse component to original weight varies in different layers. This shows that the energy distribution of sparse matrix is adaptively learned and thus pruning according to energy ratio leads to different sparse rate. The sparse rate of the bottom convolutional layers are much larger than the top convolutional layers. This is because the bottom layers extract basic feature representation of images and are more important than the top layers for the final performance.

F. ABLATION STUDY

As we can see in Figure 5, the sparse components account for most parameters of the compressed network. However, it does not mean the low-rank components are ineffective. Here we conduct ablation experiments on CIFAR-10 with VGGNet-19 to show the effects of low-rank component and sparse component more clearly. We use LRSD (L) and LRSD (S) to denote only using low-rank component and only using sparse component, respectively. We set the rank for LRSD (L) the same as LRSD, i.e., $r = 1$. Similarly, we set the trade-off parameter λ to 1e-6 for LRSD (S). We prune a sparse matrix of LRSD (S) such that its parameter number is the same as the parameter number of the sum of low-rank and sparse components in the corresponding layer in LRSD. Since LRSD compress the fully connected layer only with sparse component, the fully connected layer in LRSD (L) is without compression. Thus, LRSD (L) has slightly larger number of parameters. The results are shown in Table 6. We can see that, the combination method LRSD outperforms single low-rank or sparse component in all the cases.

V. CONCLUSION

In this paper, we integrate the low-rank decomposition and sparsity pruning into a unified framework, which captures global and local structure of a weight matrix. Comprehensive experiments demonstrate that LRSD outperforms other low-rankness or sparsity-based compression methods on both small and large scale datasets. We show that it is a good strategy to combine low-rankness and sparsity and train the compressed network from scratch. We will further combine LRSD with other extra information like discriminative information and knowledge distillation to further improve the performance of LRSD in the future.

ACKNOWLEDGMENT

(Kailing Guo and Xiaona Xie contributed equally to this work.)

REFERENCES

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, Lake Tahoe, NV, USA, Dec. 2012, pp. 1097–1105.

[2] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Miami, FL, USA, Jun. 2009, pp. 248–255.

[3] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Proc. Int. Conf. Learn. Represent.*, San Diego, CA, USA, May 2015, pp. 1–14.

[4] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Las Vegas, NV, USA, Jun. 2016, pp. 770–778.

[5] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Las Vegas, NV, USA, Jun. 2016, pp. 779–788.

[6] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," in *Proc. IEEE Int. Conf. Comput. Vis.*, Venice, Italy, Oct. 2017, pp. 2999–3007.

[7] D. Amodei, S. Ananthanarayanan, R. Anubhai, J. Bai, E. Battenberg, C. Case, J. Casper, B. Catanzaro, Q. Cheng, and G. Chen, "Deep speech 2: End-to-end speech recognition in English and Mandarin," in *Proc. Int. Conf. Mach. Learn.*, New York, NY, USA, Jun. 2016, pp. 173–182.

[8] I. Lopez-Moreno, J. Gonzalez-Dominguez, O. Plchot, D. Martinez, J. Gonzalez-Rodriguez, and P. Moreno, "Automatic language identification using deep neural networks," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, Florence, Italy, May 2014, pp. 5337–5341.

[9] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Proc. Adv. Neural Inf. Process. Syst.*, Lake Tahoe, NV, USA, Dec. 2013, pp. 3111–3119.

[10] K. Guo, L. Liu, X. Xu, D. Xu, and D. Tao, "GoDec+: Fast and robust low-rank matrix decomposition based on maximum correntropy," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 6, pp. 2323–2336, Jun. 2018.

[11] X. Zhang, J. Zou, K. He, and J. Sun, "Accelerating very deep convolutional networks for classification and detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 38, no. 10, pp. 1943–1955, Oct. 2016.

[12] S. Lin, R. Ji, X. Guo, and X. Li, "Towards convolutional neural networks compression via global error reconstruction," in *Proc. Int. Joint Conf. Artif. Intell.*, New York, NY, USA, Jul. 2016, pp. 1753–1759.

[13] S. Lin, R. Ji, C. Chen, D. Tao, and J. Luo, "Holistic CNN compression via low-rank decomposition with knowledge transfer," *IEEE Trans. Pattern Anal. Mach. Intell.*, to be published.

[14] S. Han, J. Pool, J. Tran, and W. Dally, "Learning both weights and connections for efficient neural network," in *Proc. Adv. Neural Inf. Process. Syst.*, Montréal, QC, Canada, Dec. 2015, pp. 1135–1143.

[15] Y. He, X. Zhang, and J. Sun, "Channel pruning for accelerating very deep neural networks," in *Proc. IEEE Int. Conf. Comput. Vis.*, Venice, Italy, Oct. 2017, pp. 1398–1406.

[16] J.-H. Luo, J. Wu, and W. Lin, "ThiNet: A filter level pruning method for deep neural network compression," in *Proc. IEEE Int. Conf. Comput. Vis.*, Venice, Italy, Oct. 2017, pp. 5068–5076.

[17] R. Yu, A. Li, C.-F. Chen, J.-H. Lai, V. I. Morariu, X. Han, M. Gao, C.-Y. Lin, and L. S. Davis, "NISP: Pruning networks using neuron importance score propagation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Salt Lake City, UT, USA, Jun. 2018, pp. 9194–9203.

[18] Z. Zhuang, M. Tan, B. Zhuang, J. Liu, Y. Guo, Q. Wu, J. Huang, and J. Zhu, "Discrimination-aware channel pruning for deep neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, Montréal, QC, Canada, Dec. 2018, pp. 875–886.

[19] J. M. Alvarez and M. Salzmann, "Compression-aware training of deep networks," in *Proc. Adv. Neural Inf. Process. Syst.*, Long Beach, CA, USA, Dec. 2017, pp. 856–867.

[20] M. A. Carreira-Perpinan and Y. Idelbayev, "'Learning-compression,' algorithms for neural net pruning," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Salt Lake City, UT, USA, Jun. 2018, pp. 8532–8541.

[21] T. Zhang, S. Ye, K. Zhang, J. Tang, W. Wen, M. Fardad, and Y. Wang, "A systematic DNN weight pruning framework using alternating direction method of multipliers," in *Proc. Eur. Conf. Comput. Vis.*, Munich, Germany, Sep. 2018, pp. 184–199.

[22] X. Yu, T. Liu, X. Wang, and D. Tao, "On compressing deep models by low rank and sparse decomposition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Honolulu, HI, USA, Jul. 2017, pp. 7370–7379.

[23] M. Jaderberg, A. Vedaldi, and A. Zisserman, "Speeding up convolutional neural networks with low rank expansions," in *Proc. Brit. Mach. Vis. Conf.*, Nottingham, U.K., Sep. 2014, pp. 1–13.

[24] 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.*, Montréal, QC, Canada, Dec. 2014, pp. 1269–1277.

[25] Y. LeCun, J. S. Denker, and S. A. Solla, "Optimal brain damage," in *Proc. Adv. Neural Inf. Process. Syst.*, 1990, pp. 598–605.

[26] H. Hu, R. Peng, Y.-W. Tai, and C.-K. Tang, "Network trimming: A data-driven neuron pruning approach towards efficient deep architectures," 2016, *arXiv:1607.03250*. [Online]. Available: https://arxiv.org/abs/1607.03250

[27] 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.*, Toulon, France, Apr. 2017, pp. 1–17.

[28] 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.*, Venice, Italy, Oct. 2017, pp. 2736–2744.

[29] J. Wu, C. Leng, Y. Wang, Q. Hu, and J. Cheng, "Quantized convolutional neural networks for mobile devices," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Las Vegas, NV, USA, Jun. 2016, pp. 4820–4828.

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

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

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

[33] G. E. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," 2015, *arXiv:1503.02531*. [Online]. Available: https://arxiv.org/abs/1503.02531

[34] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "MobileNets: Efficient convolutional neural networks for mobile vision applicationas," 2017, *arXiv:1704.04861*. [Online] Avaiable: https://arxiv.org/abs/1704.04861

[35] X. Zhang, X. Zhou, M. Lin, and J. Sun, "ShuffleNet: An extremely efficient convolutional neural network for mobile devices," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Salt Lake City, UT, USA, Jun. 2018, pp. 6848–6856.

[36] M. Fazel, "Matrix rank minimization with applications," Ph.D dissertation, Dept. Electr. Eng., Stanford Univ., Stanford, CA, USA, 2002.

[37] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in PyTorch," in *Proc. Adv. Neural Inf. Process. Syst.*, W. Autodiff, Ed., Long Beach, CA, USA, Dec. 2017, pp. 1–4.

[38] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," Univ. Toronto, Toronto, ON, Canada, Tech. Rep., 2009, vol. 1, no. 4.

[39] K. He, X. Zhang, S. Ren, and J. Sun, "Identity mappings in deep residual networks," in *Proc. Eur. Conf. Comput. Vis.*, Amsterdam, The Netherlands, Oct. 2016, pp. 630–645.

KAILING GUO received the B.S. and Ph.D. degrees from the School of Electronic and Information Engineering, South China University of Technology, Guangzhou, China, in 2011 and 2017, respectively. From January 2015 to January 2017, he was a Visiting Ph.D. Student with the Center for Quantum Computation and Intelligent Systems, Faculty of Engineering and Information Technology, University of Technology, Sydney. He is currently an Assistant Professor with the School of Electronics and Information and a Postdoctoral Fellow with the School of Computer Science and Engineering, South China University of Technology. His current research interests include computer vision and machine learning.

XIAONA XIE was born in Shantou, Guangdong, China, in 1997. She received the B.E. degree in information engineering from the South China University of Technology, China, in 2019. She is currently pursuing the M.E. degree with the South China University of Technology in September 2019. From 2015 to 2019, she won three scholarships, including national scholarship, school scholarship and enterprise scholarship. In 2018, she was awarded Honorable Mention in Interdisciplinary contest in Modeling as a member of the team. Her undergraduate design thesis was rated as an excellent graduation thesis by the university. Her research interest includes neural network compression.

XIANGMIN XU (M'13–SM'19) received the Ph.D. degree from the South China University of Technology, Guangzhou, China. He is currently a Full Professor with the School of Electronic and Information Engineering, South China University of Technology. His current research interests include image/video processing, human computer interaction, computer vision, and machine learning.

XIAOFEN XING received the B.S., M.S., and Ph.D. degrees from the South China University of Technology, China, in 2001, 2004, and 2013, respectively. She has been an Associate Professor with the School of Electronic and Information Engineering, South China University of Technology, since 2007. Her current research interests include image/video processing, human computer interaction, and video surveillance.