Deep Neural Network Acceleration Based on Low-Rank Approximated Channel Pruning

Zhen Chen, *Student Member, IEEE*, Zhibo Chen, *Senior Member, IEEE*, Jianxin Lin, Sen Liu, and Weiping Li, *Fellow, IEEE*

***Abstract*— Acceleration and compression on deep Convolutional Neural Networks (CNNs) have become a critical problem to develop intelligence on resource-constrained devices. Previous channel pruning can be easily deployed and accelerated without specialized hardware and software. However, weight-level redundancy is not well explored in channel pruning, which results in a relatively low compression ratio. In this work, we propose a Low-rank Approximated channel Pruning (LAP) framework to tackle this problem with two targeted steps. First, we utilize low-rank approximation to eliminate the redundancy within filter. This step achieves acceleration, especially in shallow layers, and also converts filters into smaller compact ones. Then, we apply channel pruning on the approximated network in a global way and obtain further benefits, especially in deep layers. In addition, we propose a spectral norm based indicator to coordinate these two steps better. Moreover, inspired by the integral idea adopted in video coding, we propose an evaluator based on Integral of Decay Curve (IDC) to judge the efficiency of various acceleration and compression algorithms. Ablation experiments and IDC evaluator prove that LAP can significantly improve channel pruning. To further demonstrate the hardware compatibility, the network produced by LAP obtains impressive speedup efficiency on the FPGA.**

***Index Terms*— Deep learning, network acceleration, channel pruning, low-rank approximation, efficiency evaluation, hardware resources.**

I. Introduction

In recent decades, Internet of Things (IoT) has been applied in various fields, including but not limited to industry, health, and transport [1]–[3]. In the meantime, deep learning techniques, especially Convolutional Neural Networks (CNNs), have shown superior performance on a series of artificial intelligence tasks, e.g., image classification [4] and object detection [5]. The trend is obvious that the intelligence of deep learning is indispensable for the development of IoT [6]. However, the high requirements on hardware resources hinder the prevailing of deep learning on IoT applications. In time-sensitive tasks, e.g., intelligent surveillance, autonomous driving, and health monitoring, uploading data to cloud servers would induce intolerable latency. To achieve real-time reactions, efficient and hardware-friendly CNNs are essential to edge computing and embedded devices.

The hardware implementation of CNNs is mainly limited by model storage, computation cost and memory footprint. For example, the widely-used VGG-16 model occupies more than 500MB storage space [7]. When computing a single image with resolution 224×224, VGG-16 requires 30.9 billion float point-operations (FLOPs) and 61MB runtime memory to keep the intermediate activations, which are unaffordable to resource-constrained devices. There are many works attempting to reduce the resource usage of CNNs with acceptable performance decay. But most of these approaches could only achieve one or two aforementioned goals [8].

In order to obtain the benefit of practical speedup on general platforms, it is required to ensure the compact structure of the processed convolutional layers [9], [10]. Many approaches based on low-rank approximation and channel pruning have tried to satisfy the constraint on structural compactness. However, there still exist some limitations on these works.

Low-rank approximation generates compact and efficient networks by abandoning unimportant components in a layer-oriented way [11], [12]. These methods provide a layer-wise optimal solution under a given metric, e.g., the minimum of approximation error in L_2-norm. However, the impact of approximation errors is accumulated layer by layer. Due to the lack of global consideration, small approximation error in each layer may bring serious damage to the whole network. Besides, burdensome trials are required to find the satisfying rank of each layer.

To guarantee the compact structure, channel pruning utilizes the filter-level sparsity as an aggressive approach [13]. But at a price, the lack of flexibility brings some shortcomings. (1) Channel pruning cannot make use of weight-level sparsity, and results in relatively low acceleration efficiency. (2) Since each filter is either entirely deleted or kept, pruning wrong filters may discard lots of important weights and bring unrecoverable damage to the network performance [14].

To tackle these bottlenecks of low-rank approximation and channel pruning, we find that the combination of these two methods would achieve acceleration and compression more efficiently. We first conduct low-rank approximation on CNNs

Manuscript received May 15, 2019; revised August 27, 2019 and November 3, 2019; accepted December 3, 2019. Date of publication January 1, 2020; date of current version April 1, 2020. This work was supported in part by NSFC under Grant U1908209, Grant 61571413, and Grant 61632001, and in part by the National Key Research and Development Program of China under Grant 2018AAA0101400. This article was recommended by Associate Editor Y. Uwate. *(Corresponding author: Zhibo Chen.)*

Z. Chen is with the City University of Hong Kong (CityU), Hong Kong, and also with the University of Science and Technology of China (USTC), Hefei 230026, China (e-mail: zchen.ee@my.cityu.edu.hk).

Z. Chen, J. Lin, S. Liu, and W. Li are with the CAS Key Laboratory of Technology in Geo-spatial Information Processing and Application System, University of Science and Technology of China, Hefei 230026, China (e-mail: chenzhibo@ustc.edu.cn).

Color versions of one or more of the figures in this article are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCSI.2019.2958937

to eliminate redundancy within the filter. Then we apply channel pruning to continue to accelerate and compress the network in a global way. On the one hand, the smaller but compact filters of the approximated network alleviate the damage of misjudging in the following channel pruning. On the other hand, combining with channel pruning helps low-rank approximation to handle the redundancy from a cross-layer perspective.

Another problem in this field is that a rational evaluation method is urgent to assess the efficiency of various acceleration and compression algorithms. Under the current evaluation scheme, for two compression algorithms A and B processing the same network, if A could provide a higher compression ratio and the performance decay induced by A is not higher than B simultaneously, the algorithm A is considered to be more efficient than B, and vice versa. However, this method compares algorithms at specific states, rather than globally. Moreover, it needs troublesome trials to produce the states that meet the rigorous judging condition. Given an acceleration and compression algorithm, we are able to produce a polynomial curve by fitting several samples at different compression ratios. In video coding, Bjøntegaard Delta PSNR (BD-PSNR) is a measurement of the coding efficiency using the relationship between rate and distortion [15]. Inspired by the idea of BD-PSNR, the area under the fitting curve within a specified interval represents the average performance decay, which provides a rational measurement of the efficiency. In this way, we could judge various acceleration and compression algorithms by comparing corresponding areas within the specified integral interval.

A preliminary version of this work has been presented in a conference [16], and this manuscript extends the initial version from several aspects to strengthen our contribution.

- We introduce a Low-rank Approximated channel Pruning (LAP) to pursue a higher acceleration ratio and compression ratio on CNNs with the constraint of negligible performance decay. The low-rank approximation step and the channel pruning step in the LAP framework are utilized to eliminate the redundancy at weight-level and filter-level respectively. The LAP framework also maintains the structural compactness of the processed CNNs.
- We propose a spectral norm based indicator for low-rank approximation. This indicator helps us to achieve a better balance between the two steps in the LAP framework.
- To objectively evaluate the efficiency of various acceleration and compression algorithms, we propose an evaluator for resource-constrained scenarios, named Integral of Decay Curve (IDC)[1]. To the best of our knowledge, our IDC represents the first attempt to evaluate the efficiency of acceleration and compression algorithms from a global perspective.
- We introduce a mathematical model on misjudging in channel pruning, which theoretically reveals the effectiveness of the LAP. Besides, ablation experiments and hardware implementations proves that our LAP effectively accomplishes reduction on the number of FLOPs and parameters as well as the latency on hardware platform simultaneously.

[1]Source code is available at https://github.com/franciszchen/IDC_Evaluation.

The remainder of this paper is organized as follows. In Section II, we introduce the related works. Then, our mathematical analysis and LAP framework, as well as the necessary introduction to the baseline low-rank approximation and Taylor pruning, are presented in Section III. In Section IV, we introduce the IDC evaluator in detail. In Section V, we present and analyze experimental results. Finally, we draw the conclusions in Section VI.

II. Related Work

Network compression mainly concentrates on shrinking model storage, and network acceleration focuses on reducing FLOPs. These two goals are consistent in most cases. There are many existing works aiming at deep neural network compression and acceleration from different perspectives, such as low-rank approximation, weight quantization, distill learning, pruning, and so on [17].

Weight quantization methods reduce the number of bits required for weight expression. Chen *et al.* hash weights to different groups and share the value for each group [18]. Once the weights are quantized into binary/ternary, the computation could be accelerated with bitwise operation libraries [19]. In addition, a straightforward quantization method is to convert the weights from float-point number into fixed-point format by various mappings, including linear quantization, min max quantization and log min max quantization [20]. Distill learning is used to extract knowledge from trained deep networks and transfers the learned knowledge to a smaller network with a reduction in storage and computation [21]–[23].

To obtain practical speedup, low-rank approximation based approaches factorize each convolutional layer into several efficient ones [24]. To keep network performance, they approximate the weight matrix or tensor with low-rank techniques, e.g., truncated Singular Value Decomposition (SVD) [25], CP-decomposition [26] and Tucker decomposition [27]. Tai *et al.* provide a closed-form solution for SVD-based approximation [28]. For IoT scenarios, Maji *et al.* propose ADaPT method that represents the original matrix by the sum of several low-rank substitutes [29].

Pruning can be categorized by different granularity levels, including weight-level, kernel-level, filter-level and layer-level [9], [30]. Weight pruning has higher flexibility [31], and obtains better storage compression by representing the sparse pruned network in Compressed Sparse Column (CSC) [32], but it cannot reduce the inference time and memory footprint without specialized hardware [10]. On the contrary, layer-level pruning is impracticable unless the network depth is sufficient [33].

Compared to these two extremes, channel pruning provides a trade-off between flexibility and ease of implementation. Channel pruning regards each filter as the basic unit and discards the filters that are considered to be insignificant, and finally produces a smaller network with compact structure. The main challenge hinges on employing the appropriate pruning criterion. Some works treat filter selection of each layer as

an iterative optimization problem, and it needs empirical trials to balance pruning ratio and accuracy decay layer by layer [34]. Other works propose heuristic criteria to estimate the importance of filters, e.g., various norms of weights or activations [35], the entropy of filters [36], the percentage of zeros in output feature maps [37], scale factors in batch normalization layers [38] and even to pick out unimportant filters by reinforcement learning [39]. Taylor pruning regards the network loss as a differentiable function of intermediate activation, and the retained first-order term of Taylor expansion indicates the importance of feature map and corresponding filter [40]. Their experiments had proved that Taylor pruning outperforms other channel pruning methods.

In recent years, Field Programmable Gate Array (FPGA) becomes popular to deploy deep learning applications because of its high parallelism, high flexibility and low energy consumption [8]. There are many works to optimize the deployment from the RTL-level [41]–[43]. However, when implementing large-scale CNNs, the deployment designed by hand becomes burdensome. More and more works implement acceleration and compression algorithms on FPGA platforms with the help of Xilinx Vivado High Level Synthesis (HLS) software [44]–[46].

III. Combination of Low-Rank Approximation and Channel Pruning

In this section, we first introduce SVD-based low-rank approximation and Taylor pruning in brief. Then, with mathematical analysis, we conclude that smaller filters would improve the performance of channel pruning. By applying low-rank approximation as a pre-processing properly, we propose a novel acceleration and compression framework, named Low-rank Approximated channel Pruning (LAP). In order to coordinate the two steps in LAP, we propose a spectral norm based indicator to achieve better rank selection.

A. SVD-Based Low-Rank Approximation

Low-rank approximation is implemented for each convolutional layer. For a convolutional layer with parameters $\mathcal{W} \in \mathbb{R}^{N\times d\times d\times C}$ as a 4D tensor, where N and C are the number of output channel and input channel respectively, and d is the spatial size of filters. The weight tensor $\mathcal{W}$ is reshaped into a 2D matrix $W \in \mathbb{R}^{Nd\times dC}$, and the singular values of W are denoted as $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_n \geq 0$. Then the weight matrix W is implemented with singular value decomposition as:

$$W = USV, \tag{1}$$

where $S \in \mathbb{R}^{n\times n}$ is the diagonal matrix with singular values $\sigma_1, \sigma_2, \cdots, \sigma_n$, $U \in \mathbb{R}^{Nd\times n}$ and $V \in \mathbb{R}^{n\times dC}$.

In order to approximate W, we truncate S with the k-largest singular values, and the processed weight matrix $\tilde{W}_k$ is computed as:

$$\tilde{W}_k = \tilde{U}_k \tilde{S}_k \tilde{V}_k, \tag{2}$$

where $\tilde{S}_k \in \mathbb{R}^{k\times k}$ preserves k diagonal values of S, $\tilde{U}_k \in \mathbb{R}^{Nd\times k}$ keeps corresponding k column vectors of U and $\tilde{V}_k \in \mathbb{R}^{k\times dC}$ contains k row vectors of V.

To construct approximated convolutional layers with $\tilde{W}_k = \tilde{U}'\tilde{V}'$, the preserved singular components could multiply $\tilde{U}_k$ and $\tilde{V}_k$ respectively:

$$u'_i = u_i\sqrt{\sigma_i}, \tag{3}$$

$$v'_j = v_j\sqrt{\sigma_j}, \tag{4}$$

where u'_i is the i-th column vector of $\tilde{U}'$ and v'_j is the j-th row vector of $\tilde{V}'$. Finally, the approximated matrices, $\tilde{U}'$ and $\tilde{V}'$, are reshaped into 4D tensors, $\tilde{\mathcal{U}}'$ and $\tilde{\mathcal{V}}'$, respectively. In this way, the original convolutional layer is replaced with two low-rank convolutional layers with weights $\tilde{\mathcal{U}}'$ and $\tilde{\mathcal{V}}'$. With appropriate rank k, the $\tilde{\mathcal{U}}'$ and $\tilde{\mathcal{V}}'$ demand less resources than the original matrix $\mathcal{W}$, and the low-rank approximation achieves a reduction on both parameters and FLOPs.

B. Taylor Pruning

Channel pruning attempts to find and discard the least necessary filters while maintaining the cost function as close as possible to that of the original network. The naive solution requires to check the cost change with each filter kept or pruned, which is exponential with the number of filters in the network. Obviously, the complexity of this method is unaffordable, even for small networks.

Molchanov *et al.* utilized the first-order term of Taylor expansion to estimate the change of cost function, which is regarded as the importance score Θ_{TE} of each filter [40]. The score Θ_{TE} is computed as the difference between the cost function of the network with parameter h_i pruned and the original network:

$$\Theta_{TE}(h_i) = \left|\mathcal{C}(h_i = 0) - \mathcal{C}(h_i)\right|, \tag{5}$$

where $\mathcal{C}(h_i = 0)$ represents the cost value with parameter h_i pruned, while $\mathcal{C}(h_i)$ denotes the cost if h_i kept.

For a function $f(x)$, the Taylor expansion at $x = a$ is:

$$f(x) = \sum_{p=0}^{P} \frac{f^{(p)}(a)}{p!}(x-a)^p + R_p(x), \tag{6}$$

where $f^{(p)}(a)$ is the p-th derivative of f computed at $x = a$, and $R_p(x)$ is the p-th order remainder. $\mathcal{C}(h_i = 0)$ is approximated with a first-order Taylor polynomial near $h_i = 0$:

$$\mathcal{C}(h_i = 0) = \mathcal{C}(h_i) - \frac{\partial \mathcal{C}}{\partial h_i}h_i + R_1(h_i = 0). \tag{7}$$

Particularly, the remainder $R_1(h_i = 0)$ can be calculated through the Lagrange form:

$$R_1(h_i = 0) = \frac{\mathcal{C}^{(2)}(\xi)}{2}h_i^2, \tag{8}$$

where ξ is a real number between 0 and h_i. Since the widely-used ReLU activation function encourages a smaller second order term, the reminder R_1 could be neglected in order to reduce the burdensome calculation.

By substituting Eq. (7) into Eq. (5) and ignoring the reminder R_1, the importance score Θ_{TE} is computed as:

$$\Theta_{TE}(h_i) = \left| \mathcal{C}(h_i) - \frac{\partial \mathcal{C}}{\partial h_i} h_i - \mathcal{C}(h_i) \right| = \left| \frac{\partial \mathcal{C}}{\partial h_i} h_i \right|. \quad (9)$$

In practice, Θ_{TE} is computed for multi-variate output, such as a feature map:

$$\Theta_{TE}\left(z_l^{(q)}\right) = \left| \frac{1}{M} \sum_m \frac{\partial \mathcal{C}}{\partial z_{l,m}^{(q)}} z_{l,m}^{(q)} \right|, \quad (10)$$

where M is the length of the vectorized feature map, and $z_l^{(q)} \in \mathbb{R}^{H_\ell \times W_\ell}$ is the q-th feature map in layer l. For a mini-batch with $T > 1$ samples, the score is computed for each sample separately and averaged over T.

In order to implement Taylor pruning across layers instead of layer-wise, the score Θ_{TE} is normalized within each layer:

$$\hat{\Theta}_{TE}\left(z_l^{(q)}\right) = \frac{\Theta_{TE}\left(z_l^{(q)}\right)}{\sqrt{\sum_j \left(\Theta_{TE}\left(z_l^{(j)}\right)\right)^2}}. \quad (11)$$

C. Analysis for Channel Pruning Misjudging

Previous works found that the network would suffer severe performance decay once some filters are pruned, while other filters seem inessential. For a clearer statement, these two kinds of filters are called significant filters and insignificant ones respectively. In each iteration, channel pruning algorithms attempt to prune the least important filter, but cannot avoid mistakes [40]. Given a certain pruning criterion, the probability that the pruning algorithm misjudges is assumed as a constant during pruning iterations, which is denoted as p. Noted that misjudging means picking any filter rather than the least important one, including any other insignificant filter and significant one.

According to the principle of maximum entropy, pruning which wrong filter can be regarded as an equal probability event when pruning misjudges [47]. As shown in Fig. 1, for a network with m_0 significant filters and the rest m_1 insignificant ones, the probability P_1 of pruning a significant filter is:

$$P_1 = \frac{m_0}{m_0 + m_1 - 1} p. \quad (12)$$

In order to reduce the granularity of filters, each filter of the original network is decomposed into s smaller filters. As the decomposition methods, e.g., low-rank approximation, reserve most of the information, it is reasonable to assume the ratio between significant and insignificant filters doesn't change in this procedure. For the decomposed network at the bottom of Fig. 1, the probability P_1' of pruning a significant filter is:

$$P_1' = \frac{sm_0}{s(m_0 + m_1) - 1} p, \quad (13)$$

and P_1' is slightly smaller than P_1. More importantly, as the proportion of weights in each filter to the network is reduced by s times, severe mistakes are alleviated into recoverable.

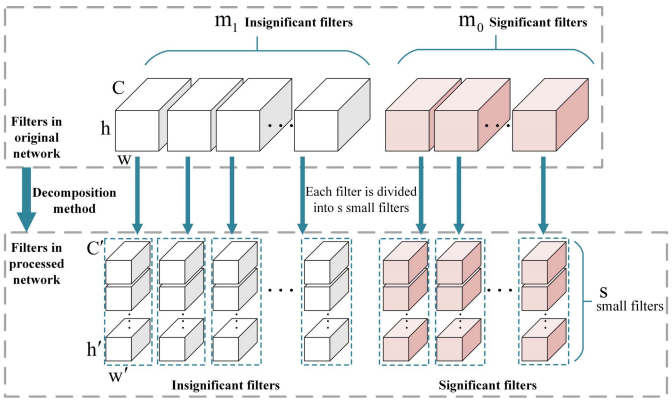


Fig. 1. Schema on reducing the granularity of filters. The original network at the top consists of m_0 significant filters (red) and m_1 insignificant ones (white). Each filter in the original network is represented as a $h \times w \times C$ cuboid. In order to alleviate the damage of channel pruning, each filter is decomposed into s smaller filters, and the processed network is presented at the bottom.

Compared with pruning an important filter in the original network, pruning s corresponding important filters induces the equivalent degree of damage. But the probability P_s' that pruning s important filters in the decomposed network is computed as follows:

$$P_s' = \frac{C_{sm_0}^s}{C_{s(m_0+m_1)-1}^s} p^s, \quad (14)$$

where $s \ll m_0$, thus P_s' is approximate to $(\frac{m_0}{m_0+m_1})^s p^s$. With P_s' much smaller than P_1, we conclude that the serious damage is less likely to happen. To sum it up, smaller filters relieve the damage in channel pruning and reduce the probability that severe mistake happens.

D. Low-Rank Approximated Channel Pruning

The aforementioned shortcomings limit the efficiency of channel pruning and low-rank approximation. Fortunately, low-rank approximation and channel pruning can compensate each other. Besides, they both keep network compact, which is necessary for practical speedup.

We propose a hardware-friendly acceleration and compression framework, LAP, which improves channel pruning with the help of low-rank approximation. These two steps of our LAP have clear functions and eliminate the redundancy at weight-level and filter-level sequentially. In this way, LAP can avoid processing the network in an aggressive way. The whole framework is illustrated in Fig. 2.

Step 1: Low-rank approximation method is implemented to diminish the redundancy within the filter. As a result, the approximated network consists of smaller filters, which improves the flexibility for the following channel pruning. By reducing the parameters, especially in the shallow layers, we obtain impressive acceleration benefits in this step.

Step 2: We implement channel pruning on the approximated network in the first step. According to the previous analysis, the smaller filters will help channel pruning generate a more efficient network with constraint on performance decay. This step is employed to obtain global compression benefits by eliminating the leftover redundancy, especially in deep layers.

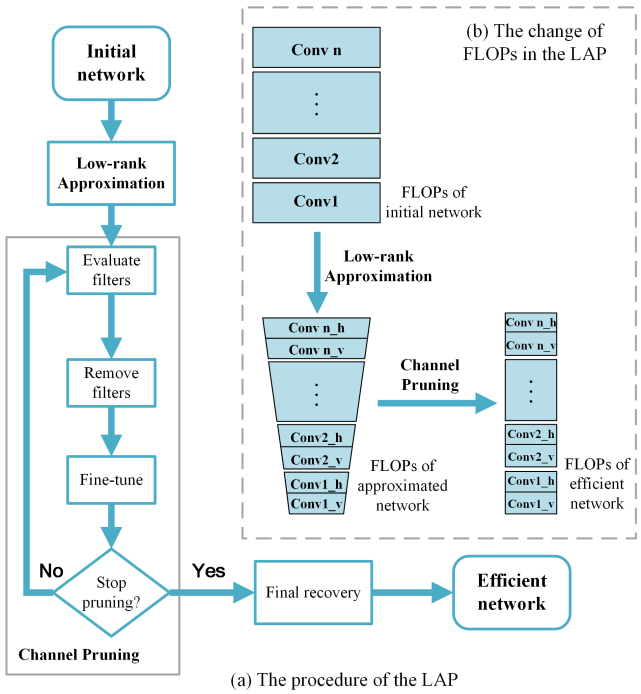


Fig. 2. The LAP framework. (a) The procedure of LAP. The LAP first conducts low-rank approximation on the initial network, then implement channel pruning on the approximated network. After the final recovery, the LAP would produce an efficient and compact network. (b) The change of FLOPs in the LAP. The area of the blue box stands for the FLOPs of the corresponding convolutional layer. The first step reduces most of the computation in shallow layers, and the second step diminishes parameters globally, especially the computation in deep layers. The LAP impressively reduces the FLOPs while maintaining the compact structure of the network.

We integrate these two steps in accordance with the fact that the computation reduction is more efficient in shallow layers. For a convolutional layer with weight tensor $\mathcal{W} \in \mathbb{R}^{N\times d\times d\times C}$, the amounts of parameters and computation are calculated as follows:

$$\text{Param} = (d^2C + 1)N, \tag{15}$$

$$\text{FLOPs} = 2HW(d^2C + 1)N, \tag{16}$$

where H, W and N are the height, the width and the number of channels of output feature maps, C is the number of channels of input feature maps, and d is the spatial size of the filter.

From Eq. (15) and Eq. (16), the FLOPs of a convolutional layer is proportional to the resolution of the output feature maps. As the resolution of feature maps generally decreases when the layer deepens, the computation in shallow layers is more sensitive to parameter reduction. Thus, the Step 1 in LAP pays more attention to shallow layers.

Previous works pointed out that channel pruning is likely to prune more filters in deeper layers, because deep layers produce relatively lower importance scores under a global pruning metric [40]. Compared with deep layers, channel pruning cannot accomplish efficient acceleration on shallow layers. Fortunately, with the help of low-rank approximation in the Step 1, we are able to restrict smaller ranks in shallow layers to obtain satisfying acceleration benefits. Meanwhile, we retain moderate redundancy in deep layers, which is required for further performance recovery. On the other hand, the redundancy left in deep layers could be easily diminished in the following channel pruning. In this way, the acceleration and compression benefits of the LAP are maximized. More details on how to balance these two steps in LAP are introduced in the subsection *F*.

E. A Better Indicator for Rank Selection

In low-rank approximation, the retained rank determines the benefits of acceleration and the induced performance decay. However, it is impossible to evaluate the network performance with every possible rank. Thus, an indicator is needed for the rank selection. When the indicator approaches a certain value, the retained rank is expected to achieve a nice trade-off between the acceleration and performance of the approximated network.

Previous works employed a Frobenius norm based indicator for low-rank approximation [48]. Denote that a weight matrix $W \in \mathbb{R}^{m\times n}(m > n)$ contains n singular values as $\sigma_1 \geq \cdots \geq \sigma_n \geq 0$. When the approximated matrix $\tilde{W}_k$ retains the k-largest singular values, this indicator α_F is computed as the normalized error between $\tilde{W}_k$ and W:

$$\begin{aligned}\alpha_F &= \frac{\| W - \tilde{W}_k \|_{\mathrm{F}}}{\| W \|_{\mathrm{F}}} \\ &= \frac{\sqrt{\sum_{i=k+1}^{n} \sigma_i^2}}{\sqrt{\sum_{j=1}^{n} \sigma_j^2}},\end{aligned} \tag{17}$$

where $0 \leq k < n$. In practice, we find that the spectral norm performs more robust than the Frobenius one. As for the same approximated matrix $\tilde{W}_k$, the spectral norm of the approximation error is computed as follows:

$$\| W - \tilde{W}_k \|_{\text{spec}} = \sigma_{k+1}. \tag{18}$$

Particularly, the spectral norm of the original matrix W is $\| W \|_{\text{spec}} = \sigma_1$. Our spectral norm based indicator α is defined as the relative magnitude of the largest abandoned component:

$$\begin{aligned}\alpha &= \frac{\| W - \tilde{W}_k \|_{\text{spec}}}{\| W \|_{\text{spec}}} \\ &= \frac{\sigma_{k+1}}{\sigma_1}.\end{aligned} \tag{19}$$

We find that when the spectral norm based indicator α is between 0.20 and 0.25, the network on ImageNet dataset reaches a nice trade-off between performance and acceleration, as shown in Table I. At the same time, the Frobenius norm based indicator α_F changes dramatically from 0.02 to 0.23 in different layers. In our analysis, Frobenius norm describes the cumulative energy of singular values, and numerous small singular values may interfere with rank selection. In fact, these components with small magnitudes provide negligible contribution to the inference of the network, and some of them can be regarded as noise. But the spectral norm based indicator α has non-maximum suppression over the noisy components. Once the abandoned component with a large singular value is

TABLE I
SPECTRAL NORM INDICATOR VS. FROBENIUS NORM INDICATOR

AlexNet	conv1	conv2	conv3	conv4	conv5
Rank	15	70	135	130	65
Acc. decay	-1.4%	-1.8%	-1.8%	-1.7%	-2.1%
F-norm ratio [49]	0.02	0.08	0.13	0.20	0.23
Spec-norm ratio	0.25	0.22	0.20	0.20	0.20
VGG-16	**conv1_2**	**conv2_1**	**conv3_2**	**conv4_2**	**conv4_3**
Rank	21	35	128	192	170
Acc. decay	-0.3%	-0.5%	-0.6%	-0.6%	-1.0%
F-norm ratio [49]	0.05	0.07	0.11	0.16	0.21
Spec-norm ratio	0.25	0.23	0.24	0.25	0.20

considered to be unimportant to the network, the other components with smaller singular values are reasonably negligible.

F. Interaction Between Low-Rank Approximation and Channel Pruning

For the low-rank approximation step of LAP, we introduce a novel spectral norm based indicator to pick the retained rank in Step 1. And in the channel pruning step, pruning ratio describes the percentage of the removed filters. In this way, the spectral norm based indicator and pruning ratio determine the trade-off between the performance decay and the benefit of acceleration and compression in LAP.

We present the effects of changing the spectral norm based indicator and pruning ratio on CIFAR-10 network respectively. In the Fig. 3(a), the pruning ratio is fixed at 0.5, and the accuracy and the benefit of LAP both vary with the indicator value. With the increase of the indicator value, the acceleration ratio of LAP ascends steadily. At a price, the accuracy drops, and the decline of accuracy is getting more and more intense. The Fig. 3(b) shows the similar impact of changes in pruning ratio when the indicator value is fixed.

As the benefits of LAP increase in a relatively stable way, we recommend to choose the indicator value or pruning ratio before the accuracy drops sorely in each step. It is suitable to pick a spectral norm based indicator around 0.7 in Fig. 3(a) and keep pruning ratio between 0.45 and 0.55 in Fig. 3(b).

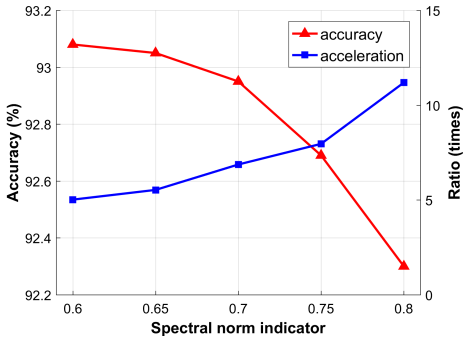
G. Compression, Acceleration and Speedup Analysis

Compression ratio is utilized to measure the degree to which the network is compressed, and computed as the parameter amount of the original network dividing that of the compressed network. Similarly, acceleration ratio is computed by the division in terms of the FLOPs, which represents the reduction in theoretical time consuming. The compression ratio r_c and acceleration ratio r_a are computed respectively:

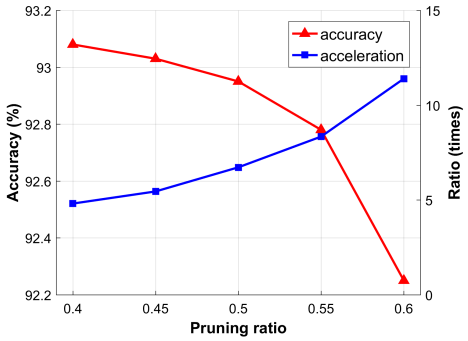
$$r_c = \frac{\text{orig Param}}{\text{compressed Param}}, \tag{20}$$

$$r_a = \frac{\text{orig FLOPs}}{\text{accelerated FLOPs}}. \tag{21}$$

For a convolutional layer with parameters $\mathcal{W} \in \mathbb{R}^{N\times d\times d\times C}$, it contains d^2NC parameters and $2HW(d^2C+1)N$ FLOPs. When the low-rank approximation truncates k singular values,



(a) The impact of low-rank approximation on the LAP as spectral norm indicator changes. The pruning ratio in Step 2 is fixed at 0.5.



(b) The impact of channel pruning on the LAP as pruning ratio changes. The spectral norm indicator in Step 1 is fixed at 0.7.

Fig. 3. The impact of low-rank approximation and channel pruning on LAP. The red line indicates the accuracy of the network, referring to the vertical axis on the left. The blue line indicates the acceleration ratio brought by the LAP, referring to the vertical axis on the right.

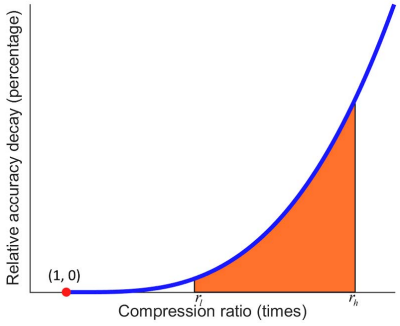


Fig. 4. Illustration on IDC calculation. The orange area under the curve indicates the accumulation of relative performance decay over the specified interval $[r_l, r_h]$.

the new structure has $kd(N+C)$ parameters and $2HW(dC+1)k + 2HW(dk+1)N$ FLOPs. The compression ratio is computed as:

$$r_c = \frac{dNC}{k(N+C)}. \tag{22}$$

And the acceleration ratio is computed as:

$$r_a = \frac{(d^2C+1)N}{kd(N+C)+N+k}. \tag{23}$$

Considering $k \ll min(N, C)$ in practice, the acceleration ratio and the compression ratio of low-rank approximation are usually satisfying. Noted that the Eq.(22) and the Eq.(23) are proposed for a single layer. The acceleration ratio or the compression ratio of the entire network needs to be weighted according to the amount of the parameters or the amount of the FLOPs in each layer.

When channel pruning keeps M filters out of original N ones, the compression ratio and the acceleration ratio for this layer are both $\frac{N}{M}$. Particularly, the acceleration ratio of the entire network needs to be weighted and accumulated according to the resolution of each layer. Thus, the acceleration ratio of the network is different from the compression ratio.

Different from acceleration focusing on theoretical benefits, speedup measures the practical time consuming, and is computed by clock cycles of hardware platforms. High speedup ratio is crucial for the deployment of CNNs.

IV. Evaluation on Compression and Acceleration Efficiency

As mentioned before, the current evaluation scheme has obvious limitations. Since acceleration and compression has similar characteristics, and we take compression for specific analysis in the following discussion. The rate-distortion relationship of BD-PSNR inspires us to consider the compression ratio and network performance at the same time. To overcome the disadvantages of the current evaluation scheme, we attempt to model the relationship between performance decay and compression ratio from a global perspective. In this section, we first analyze the relationship between performance decay and compression ratio. Then, we introduce the evaluator IDC and calculation procedures. Finally, we exhibit examples to show the rationality and necessity of IDC. Detailed experimental results are presented in Section V.

A. Relationship Between Performance Decay and Compression Ratio

Assume that there exists a function between the performance decay of the compressed network and the compression ratio. Here the decay D is defined as the relative decrease of network performance in terms of a specific metric, e.g., accuracy for classification tasks and mean square error for regression tasks. For the uncompressed network, the compression ratio r is 1 and corresponding decay is zero as follows:

$$D(r = 1) = 0. \tag{24}$$

With slight compression on the network, there is little performance decay as:

$$D(r)|_{r\to 1} \to 0. \tag{25}$$

As the compression ratio increases, the performance decay also grows, and the growth becomes faster and faster. The LAP accuracy curves in Fig. 3 also confirm this fact. Ideally, the function could be regarded as convex, with the second derivative larger than 0:

$$\frac{d^2D}{dr^2} > 0. \tag{26}$$

Algorithm 1 IDC-Based Evaluation

Input : The compression algorithms $\Theta := \{\theta_i\}$;
The network W;
The compression ratio r;
The performance decay D;
Output: The IDC set $S := \{s_j\}$;

for *each* $\theta \in \Theta$ **do**
  Conduct θ with different settings on W, and record at least 4 (r, D) sample pairs;
  Fit the third-order polynomial curve $D(r)$ with sample points by Eq.(28);
  Compute IDC score s within the intervel $[r_l, r_h]$ by Eq.(29);
  Add s into the IDC set S;
end
Compare IDC set S. The algorithm with minimum IDC is the most efficient.

Once compressed with a ratio larger than a specific threshold, the network would lose its ability. Before approaching this threshold, the convex function $D(r)$ could be regarded as smooth, as shown in Fig. 5. The smoothness guarantees that the function $D(r)$ could be fitted by low-order polynomials.

B. Integral of Decay Curve

Since the function is smooth and convex, the performance decay-compression ratio relationship could be fitted by an m-order polynomial curve:

$$D(r) = a_m r^m + \cdots + a_1 r + a_0. \tag{27}$$

We empirically find that the third-order polynomial achieves a nice balance between fitting precision and complexity, as shown in Fig. 7. Besides, the fitted third-order polynomial could accurately predict unknown performance decay at assigned compression ratio, as shown in Fig. 9 and Fig. 10. In contrast, the second-order polynomial lacks the ability to fit the relationship, and polynomials higher than the fourth-order bring overfitting. The fourth-order polynomial performs similarly as the third-order fitting, but it needs more sample points. In this case, we rewrite the fitting function Eq. (27) with third-order:

$$D(r) = a_3 r^3 + a_2 r^2 + a_1 r + a_0. \tag{28}$$

The area under the fitting curve $D(r)$ stands for the accumulation of performance decay, as shown in Fig. 4. We propose the IDC by the integral area dividing the specified range:

$$\text{IDC} = \frac{1}{r_h - r_l} \int_{r_l}^{r_h} D(r) dr, \tag{29}$$

where r_l and r_h are the lower bound and upper bound of the integral range respectively. The IDC represents the averaged performance decay induced by compression algorithms on the specified range of compression ratio. Under the same comparison conditions, a smaller IDC value means a better compression algorithm. The procedure of IDC calculation is summarized in Algorithm 1.

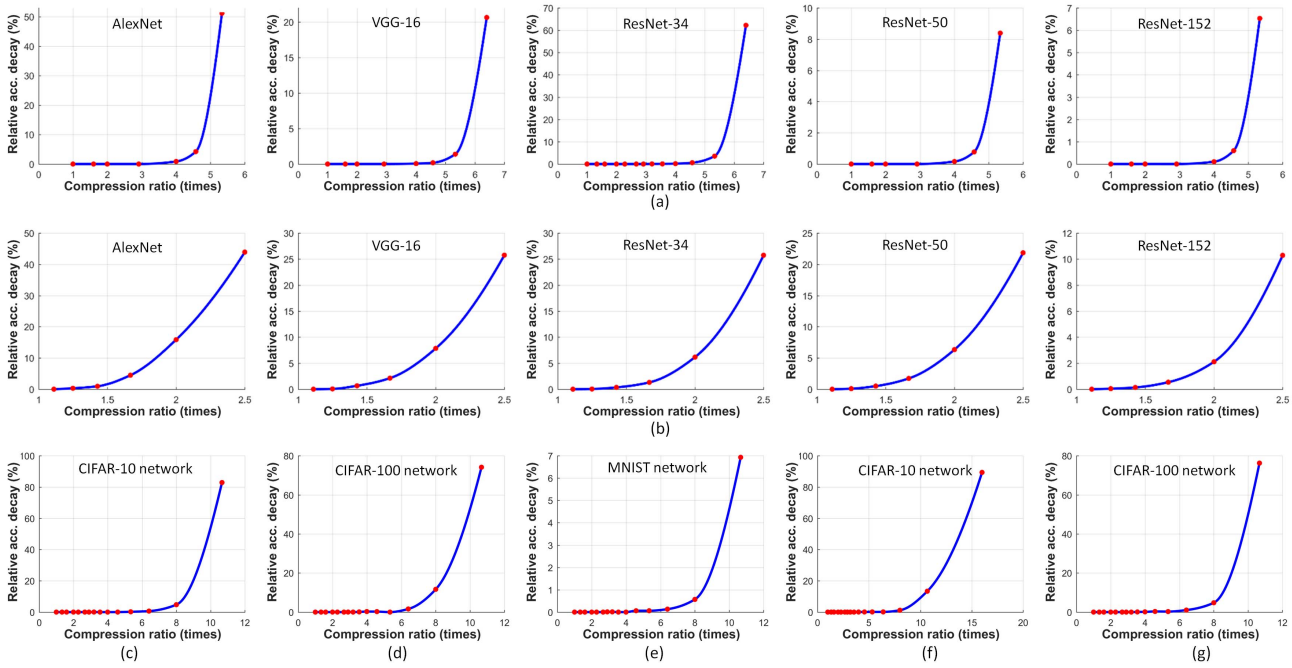
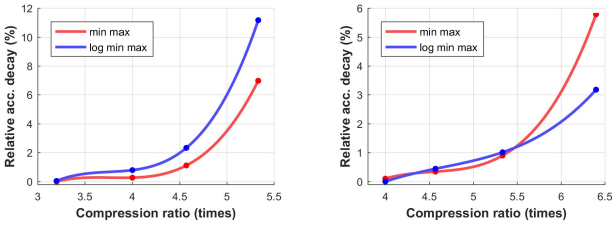


Fig. 5. The cubic interpolation curves of compression samples on various networks and datasets confirm the conjecture of smooth convex functions. In each subgraph, the horizontal axis represents the compression ratio of the network, and the vertical axis represents the relative performance decay induced by compression. (a) This row is implemented with linear quantization on various ImageNet-based networks with different bit widths, and the performance is Top-5 accuracy. (b) This row is implemented with weight pruning on various ImageNet-based networks with different pruning ratios, and the performance is Top-5 accuracy. (c), (d) and (e) are implemented with linear quantization on networks with CIFAR-10, CIFAR-100 [49] and MNIST [50] datasets respectively. (f) and (g) are implemented with min max quantization on networks with CIFAR-10 and CIFAR-100 datasets respectively.



(a) Quatization on ResNet-50 (b) Quatization on VGG-16

Fig. 6. The examples reveal the rationality and necessity of IDC. The dots are samples by different quantization methods, and the curves are fitted with third-order polynomials.

In practice, the selection of the integral interval can be determined by the needs of resource-constrained scenarios. We empirically do not recommend fitting curves with the initial point ($r = 1, D = 0$), and 4 sample points are required. In addition to the theoretical metrics like acceleration ratio and compression ratio, IDC can also be extended to analyze the practical efficiency of algorithms by replacing r with hardware measurements, e.g., the speedup ratio or the reduction on energy consumption.

C. Examples of IDC

We present two examples to demonstrate the rationality and necessity of IDC, as shown in Fig. 6. We implement two kinds of quantization methods on ResNet-50 [51] and VGG-16 [7], respectively. On ResNet-50, the fitting curve of min max quantization is always better than log min max quantization. And Table IV shows that min max quantization has lower IDC value. This example shows that the proposed IDC is consistent with the existing evaluation scheme.

But the two fitting curves of VGG-16 intersect. Under the existing evaluation scheme, the first and the fourth sample pairs show that log min max quantization is better, while the second and the third sample pairs show that min max quantization is better. These ambiguous cases prevent us from making accurate judgments about the compression algorithms. However, by calculating the integral of the curve, IDC can give a definitive judgment that log min max quantization generally performs better. Although min max quantization is slightly better in interval 4.5× to 5.5×, log min max quantization is obviously better in interval 5.5× to 6.5×. In this way, the judgment provided by IDC is reasonable.

V. Experiment

In this section, experiments are performed to show that our LAP framework can effectively boost the performance of channel pruning and low-rank approximation, and we also present the latency to further show our advantages on hardware deployment. We implement experiments with a VGG-like network [20] on CIFAR-10 dataset and CIFAR-100 dataset as well as AlexNet [4] on ImageNet [52] dataset. Widely-used SVD-based low-rank approximation method [28] and Taylor pruning method [40] are adopted in our LAP framework. Furthermore, we conduct different compression algorithms on various networks and datasets to confirm the IDC, and evaluate the efficiency of LAP with the help of IDC.

A. Experiment on Acceleration Framework

We implement low-rank approximation, channel pruning and our LAP framework on a CIFAR-10 network and AlexNet

TABLE II
PERFORMANCE COMPARISON ON CIFAR-10 NETWORK

	Accuracy	Param ($\times 10^6$)	Compression Ratio	FLOPs ($\times 10^8$)	Acceleration Ratio	Latency ($\times 10^8$)	Speedup Ratio
Original Network	93.16%	9.29	1	12.5	1	34.6	1
SVD Low-rank [28]	91.83%	1.39	6.69	2.01	6.25	4.21	8.22
Taylor Pruning [40]	92.03%	1.14	8.15	2.29	5.47	1.91	18.12
LAP	**92.25%**	**0.26**	**34.89**	**0.75**	**16.65**	**0.72**	**48.73**

TABLE III
PERFORMANCE COMPARISON ON ALEXNET

	Accuracy	Param ($\times 10^6$)	Compression Ratio	FLOPs ($\times 10^8$)	Acceleration Ratio	Latency ($\times 10^9$)	Speedup Ratio
Original Network	78.64%	2.47	1	13.1	1	3.54	1
SVD Low-rank [28]	76.68%	1.13	2.18	5.71	2.30	1.43	2.48
Taylor Pruning [40]	75.59%	1.53	1.61	8.65	1.51	2.01	1.76
LAP	**77.18%**	**0.51**	**4.84**	**3.08**	**4.25**	**0.43**	**8.23**

TABLE IV
IDC FOR QUANTIZATION METHODS ON RESNET-50 AND VGG-16

	ResNet-50	VGG-16
min max quantization	**1.45**	1.45
log min max quantization	2.65	**1.12**

TABLE V
GENERALIZATION OF LAP ON CIFAR-100 NETWORK USING THE SETTING OF CIFAR-10 NETWORK

	Accuracy	Compression Ratio	Acceleration Ratio
Original Network	72.47%	1	1
SVD Low-rank [28]	63.95%	25.61	17.81
Taylor Pruning [40]	63.41%	20.42	24.08
LAP	**64.15%**	**31.97**	**40.98**

TABLE VI
COMPARISON IN EXTREMELY RESOURCE-CONSTRAINED SCENARIOS

	Taylor Pruning [40]	ADaPT [29]	LAP
Accuracy	84.33%	86.00%	**87.44%**
Param Num	4.1×10^4	3.9×10^4	$\mathbf{3.7\times10^4}$

respectively. Our experiments are consistent with the training setting in [40]. We demonstrate the acceleration and compression performance on convolutional layers. In Table II, LAP achieves 16.65× reduction on computation and 34.89× compression simultaneously with 0.91% accuracy decay. In the ImageNet based experiments, LAP also outperforms either low-rank decomposition or channel pruning on both computation and model size, as shown in Table III. The experiments confirm that LAP effectively improves the performance of channel pruning. Noted that our experiments focus on reduction of FLOPs and parameters of convolutional layers in CNNs, and this task is much more difficult than processing fully-connected layers.

To demonstrate the generalization of settings in our LAP, we implement the LAP on CIFAR-100 network with the same setting of CIFAR-10 network in Table II. For a better comparison, low-rank approximation and pruning are conducted for more reduction on FLOPs and the amount of parameters. As shown in Table V, due to the much more difficult classification task on CIFAR-100, the performance degradation on this dataset is more obvious. However, our LAP using the predefined setting of CIFAR-10 task still outperforms two baseline methods impressively. This elaborates that our LAP is able to generalize across networks and datasets.

B. Experiment for Extremely Resource-Constrained Scenarios

The ADaPT method is designed for extremely efficient and tiny CNNs under IoT scenarios [29]. We conduct LAP and generate models with similar model size reported in the paper. Without the help of low-rank approximation, Taylor pruning performs worse than ADaPT. But the model produced by LAP outperforms ADaPT by 1.44 percent rise in network performance with fewer parameters, as shown in Table VI. This proves that our LAP is quite suitable for IoT scenarios.

C. Hardware Latency Efficiency

We deploy networks on Xilinx Zynq ZCU102 platform. The operations, including convolution operations and activation functions, are written in C++ and compiled by Vivado High Level Synthesis (HLS). We keep clock frequency at 100.0 MHz. In Table II and Table III, our LAP framework induces dramatic improvement on hardware latency, and also obtains the best latency reduction per FLOPs, as shown in Fig. 8. In our analysis, LAP generates more efficient filters with better flexibility, and obtains the most benefits from parallel computing. The consistency between FLOPs and practical latency is crucial to CNNs deployment on hardware platforms.

To further investigate the generalization of the LAP on hardware platform, we also deploy the same processed CIFAR-10 network at 500.0 MHz. The results in Table VII show that our LAP obtains more hardware speedup than two baseline methods at different clock frequencies. Besides, we also implement the CIFAR-100 networks at 100.0 MHz. This proves that our

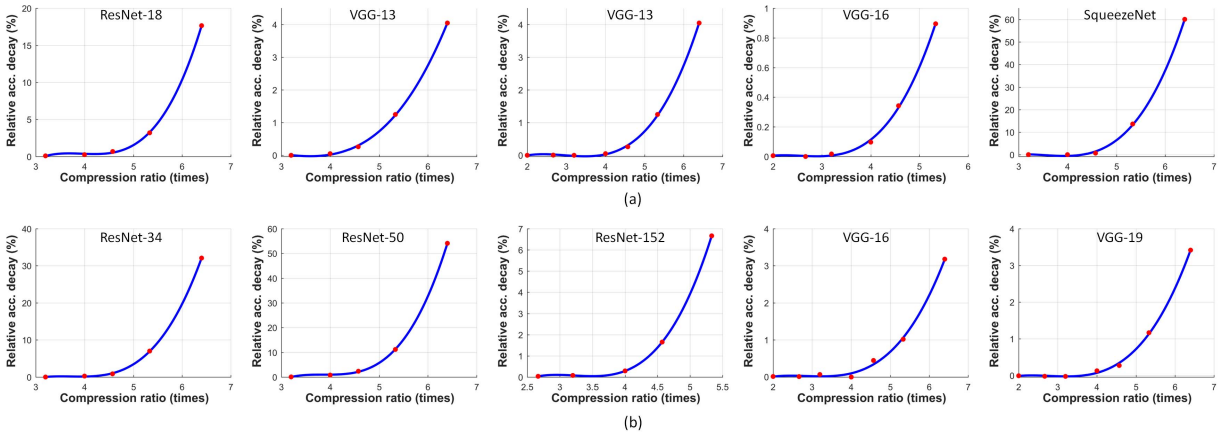


Fig. 7. Generate the fitting curves with more than 4 compression samples (red dots). The experiment is conducted with various networks and different quantization methods. (a) Subfigures are implemented with min max quantization. The R^2 is 0.9998, 0.9998, 0.9997, 0.9991 and 0.9996 from left to right. (b) Subfigures are implemented with log min max quantization. The R^2 is 0.9999, 1.0000, 1.0000, 0.9968 and 0.9990 from left to right.

TABLE VII
GENERALIZATION OF LAP IN HARDWARE DEPLOYMENT ACROSS CLOCK FREQUENCY, NETWORKS AND DATASETS

	CIFAR-10 @ 100 MHz		CIFAR-10 @ 500 MHz		CIFAR-100 @ 100 MHz	
	Acc. decay (%)	Speedup ratio	Acc. decay (%)	Speedup ratio	Acc. decay (%)	Speedup ratio
SVD Low-rank [28]	1.33	8.22	1.33	9.55	8.52	21.78
Taylor Pruning [40]	1.13	18.12	1.13	12.84	9.06	87.07
LAP	**0.91**	**48.73**	**0.91**	**50.02**	**8.32**	**99.53**

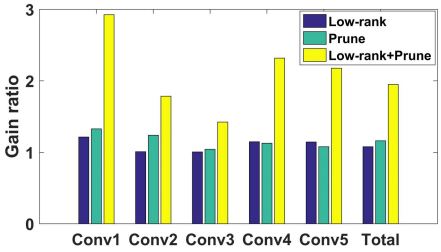


Fig. 8. The gain ratio is computed as the speedup ratio/acceleration ratio in AlexNet. It represents the efficiency of converting theoretical acceleration into practical speedup. This shows that the LAP (yellow) is more suitable for hardware deployment.

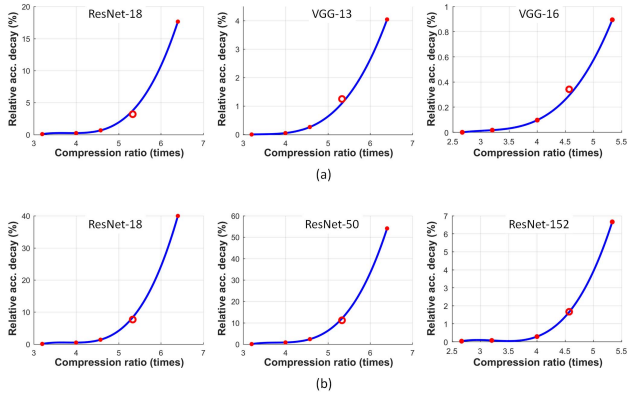


Fig. 9. Generate the fitting curves with 4 compression samples (red dots), and predict an unknown sample (red circles). The experiment is conducted with various networks and different quantization methods. (a) Subfigures are implemented with min max quantization, and the relative prediction errors are 0.1791, 0.1208 and 0.1346 from left to right. (b) Subfigures are implemented with log min max quantization, and the relative prediction errors are 0.0885, 0.0775 and 0.0023 from left to right.

LAP has higher hardware efficiency regardless of networks and datasets.

D. Experiment on Rank Selection Indicators

We compare the spectral norm based indicator with the Frobenius norm based one. We implement low-rank approximation on each single layer in AlexNet and VGG-16, and compute the Frobenius norm based indicator and the spectral norm based indicator. Here we record the Top-5 accuracy decay, which is more robust for analysis. Considering that VGG-16 has more layers, we limit its accuracy decay more strictly for the approximation on each layer. Detailed information is shown in Table I. The variation range of Frobenius norm based indicator spans orders of magnitude from 0.02 to 0.23, while our indicator is between 0.20 and 0.25 on different layers and different networks, which proves that our spectral norm based indicator is more robust for rank selection.

E. Verification of IDC Modeling

To confirm the justifiability of the third-order polynomial fitting for IDC, we observe the Goodness of Fit, R^2. The R^2 closer to 1 represents the fitting is more accurate. To ensure the generalization of the experiment, we implement different quantization methods on various networks with different num-

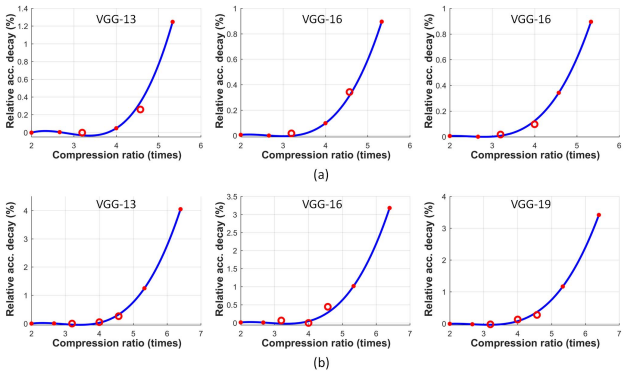


Fig. 10. Generate the fitting curves with 4 compression samples (red dots), and predict multiple unknown samples (red circles). The experiment is conducted with various networks and different quantization methods. (a) Subfigures predict 2 unknown samples and are implemented with min max quantization. The relative prediction errors are 0.1095, 0.0349 and 0.0934 from left to right. (b) Subfigures predict 3 unknown samples and are implemented with log min max quantization, and the relative prediction errors are 0.0518, 0.0740 and 0.0640 from left to right.

TABLE VIII

THE IDC COMPARISON ON CIFAR-10 DATASET FOR ACCELERATION AND COMPRESSION OF SVD-BASED LOW-RANK APPROXIMATION, TAYLOR PRUNING AND LAP WITH RECORDS AT 4 SETTINGS

	Acc. (%)	Relative Acc. Decay(%)	Compression		Acceleration	
			Ratio (times)	IDC	Ratio (times)	IDC
SVD Low-rank [28]	91.95	1.60	5.33	4.03	5.58	2.31
	91.36	2.24	8.15		7.11	
	91.09	2.52	9.24		8.42	
	90.15	3.53	28.18		22.07	
Taylor Pruning [40]	92.49	1.03	5.11	1.76	3.44	1.64
	92.08	1.47	8.15		5.47	
	91.78	1.79	16.33		10.54	
	91.65	1.93	20.74		13.36	
LAP	93.45	0	7.54	**0.36**	4.34	**0.37**
	93.28	0.18	9.27		5.34	
	93.10	0.37	13.03		7.12	
	92.25	1.28	34.89		16.65	

ber of samples. In Fig. 7, the R^2 always stays greater than 0.99. This means that a third-order polynomial model is able to fit the relationship well.

To further verify the generalization ability of the third-order polynomial, we predict the unknown performance decay with fitting curves. This experiment is conducted with different quantization methods on various networks. We compute the relative prediction error for each curve. As shown in Fig. 9 and Fig. 10, the relative prediction error is usually smaller than 0.15. This experiment proves that the fitted third-order polynomial curve has a strong generalization ability for the relationship between performance decay and compression ratio.

F. Evaluating LAP via IDC

In order to verify the efficiency of our proposed LAP, we implement the LAP on CIFAR-10 network and CIFAR-100 network with 4 different settings and compute the IDC score of the LAP respectively. The same processing is also conducted with SVD-based low-rank approximation [28] and Taylor pruning [40]. The CIFAR-10 network has the baseline accuracy 93.45% and the CIFAR-100 network obtains the baseline accuracy 72.47%. We set the integral interval as $10\times$ to $20\times$ for compression and $5\times$ to $10\times$ for acceleration. As shown in Table VIII and Table IX, our LAP achieves the minimum of IDC score for both compression and acceleration on CIFAR-10 and CIFAR-100. This means that our LAP brings the minimum performance decay over the specified interval. With the help of the proposed IDC evaluator, the experiment proves that our LAP helps low-rank approximation and channel pruning with improved efficiency of acceleration and compression.

TABLE IX

THE IDC COMPARISON ON CIFAR-100 DATASET FOR ACCELERATION AND COMPRESSION OF SVD-BASED LOW-RANK APPROXIMATION, TAYLOR PRUNING AND LAP WITH RECORDS AT 4 SETTINGS

	Acc. (%)	Relative Acc. Decay(%)	Compression		Acceleration	
			Ratio (times)	IDC	Ratio (times)	IDC
SVD Low-rank [28]	71.92	0.76	2.24	5.72	3.21	4.05
	69.93	3.50	5.74		6.18	
	68.73	5.16	11.22		10.56	
	63.95	11.76	25.61		17.81	
Taylor Pruning [40]	71.61	1.19	2.61	8.90	3.07	4.88
	68.13	5.99	7.87		9.48	
	66.89	7.70	12.38		14.90	
	63.41	12.50	20.42		24.08	
LAP	72.06	0.56	2.90	**4.59**	4.33	**1.43**
	71.20	1.75	5.62		8.67	
	68.91	4.91	16.21		20.94	
	64.15	11.48	31.97		40.98	

VI. CONCLUSION

In this work, we propose a two-step LAP framework to boost channel pruning with the help of low-rank approximation. In our LAP framework, convolutional layers are first decomposed by low-rank approximation, which converts the network into smaller filters. Then we implement channel pruning to further compress and accelerate the low-rank network. In addition, we propose a spectral norm based indicator to balance these two steps. Moreover, we introduce the IDC evaluator to measure the efficiency of various acceleration and compression algorithms. With the help of the IDC, experiments show that our LAP provides impressive improvement on both acceleration and compression. Implementations on FPGA further proves that our LAP is more suitable for hardware deployment with lower latency. In the future, we are going to employ LAP to develop real-time applications for IoT devices.

REFERENCES

[1] L. Da Xu, W. He, and S. Li, "Internet of Things in industries: A survey," *IEEE Trans. Ind. Informat.*, vol. 10, no. 4, pp. 2233–2243, Nov. 2014.
[2] S. M. Riazul Islam, D. Kwak, M. H. Kabir, M. Hossain, and K.-S. Kwak, "The Internet of Things for health care: A comprehensive survey," *IEEE Access*, vol. 3, pp. 678–708, Jun. 2015.
[3] G. Kortuem, F. Kawsar, D. Fitton, and V. Sundramoorthy, "Smart objects as building blocks for the Internet of Things," *IEEE Internet Comput.*, vol. 14, no. 1, pp. 44–51, Jan./Feb. 2010.

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

[5] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," in *Proc. Adv. neural Inf. Process. Syst.*, 2015, pp. 91–99.

[6] H. Li, K. Ota, and M. Dong, "Learning IoT in edge: Deep learning for the Internet of Things with edge computing," *IEEE Netw.*, vol. 32, no. 1, pp. 96–101, Jan./Feb. 2018.

[7] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014, *arXiv:1409.1556*. [Online]. Available: https://arxiv.org/abs/1409.1556

[8] Q. Zhang, M. Zhang, T. Chen, Z. Sun, Y. Ma, and B. Yu, "Recent advances in convolutional neural network acceleration," *Neurocomputing*, vol. 323, pp. 37–51, Jan. 2019.

[9] W. Wen, C. Wu, Y. Wang, Y. Chen, and H. Li, "Learning structured sparsity in deep neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, pp. 2074–2082.

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

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

[12] R. Rigamonti, A. Sironi, V. Lepetit, and P. Fua, "Learning separable filters," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2013, pp. 2754–2761.

[13] 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.*, Oct. 2017, pp. 5058–5066.

[14] H. Wang, Q. Zhang, Y. Wang, and H. Hu, "Structured probabilistic pruning for convolutional neural network acceleration," in *Proc. Brit. Mach. Vis. Conf. (BMVC)*, 2018.

[15] T. Wiegand, G. J. Sullivan, G. Bjontegaard, and A. Luthra, "Overview of the H. 264/AVC video coding standard," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 13, no. 7, pp. 560–576, 2003.

[16] Z. Chen *et al.*, "Exploiting weight-level sparsity in channel pruning with low-rank approximation," in *Proc. Int. Symp. Circuits Syst.*, 2019.

[17] Y. Cheng, D. Wang, P. Zhou, and T. Zhang, "Model compression and acceleration for deep neural networks: The principles, progress, and challenges," *IEEE Signal Process. Mag.*, vol. 35, no. 1, pp. 126–136, Jan. 2018.

[18] W. Chen, J. Wilson, S. Tyree, K. Weinberger, and Y. Chen, "Compressing neural networks with the hashing trick," in *Proc. Int. Conf. Mach. Learn.*, 2015, pp. 2285–2294.

[19] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, "Xnor-net: ImageNet classification using binary convolutional neural networks," in *Proc. Eur. Conf. Comput. Vis.* Cham, Switzerland: Springer, 2016, pp. 525–542.

[20] X. Chen. *Pytorch-Playground*. Accessed: Dec. 6, 2019. [Online]. Available: https://github.com/aaron-xichen/pytorch-playground

[21] G. 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

[22] A. Romero, N. Ballas, S. E. Kahou, A. Chassang, C. Gatta, and Y. Bengio, "Fitnets: Hints for thin deep nets," 2014, *arXiv:1412.6550*. [Online]. Available: https://arxiv.org/abs/1412.6550

[23] J. Yim, D. Joo, J. Bae, and J. Kim, "A gift from knowledge distillation: Fast optimization, network minimization and transfer learning," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jul. 2017, pp. 4133–4141.

[24] 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.

[25] 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.

[26] V. Lebedev, Y. Ganin, M. Rakhuba, I. Oseledets, and V. Lempitsky, "Speeding-up convolutional neural networks using fine-tuned cp-decomposition," 2014, *arXiv:1412.6553*. [Online]. Available: https://arxiv.org/abs/1412.6553

[27] Y.-D. Kim, E. Park, S. Yoo, T. Choi, L. Yang, and D. Shin, "Compression of deep convolutional neural networks for fast and low power mobile applications," 2015, *arXiv:1511.06530*. [Online]. Available: https://arxiv.org/abs/1511.06530

[28] C. Tai *et al.*, "Convolutional neural networks with low-rank regularization," 2015, *arXiv:1511.06067*. [Online]. Available: https://arxiv.org/abs/1511.06067

[29] P. Maji, D. Bates, A. Chadwick, and R. Mullins, "Adapt: Optimizing cnn inference on IoT and mobile devices using approximately separable 1-d kernels," in *Proc. 1st Int. Conf. Internet Things Mach. Learn.*, 2017, p. 43.

[30] S. Anwar, K. Hwang, and W. Sung, "Structured pruning of deep convolutional neural networks," *ACM J. Emerg. Technol. Comput. Syst.*, vol. 13, no. 3, p. 32, 2017.

[31] 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.*, 2015, pp. 1135–1143.

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

[33] G. Huang, Y. Sun, Z. Liu, D. Sedra, and K. Q. Weinberger, "Deep networks with stochastic depth," in *Proc. Eur. Conf. Comput. Vis.* Cham, Switzerland: Springer, 2016, pp. 646–661.

[34] Y. He, X. Zhang, and J. Sun, "Channel pruning for accelerating very deep neural networks," in *Proc. Int. Conf. Comput. Vis. (ICCV)*, vol. 2, 2017, p. 6.

[35] H. Li, A. Kadav, I. Durdanovic, H. Samet, and H. P. Graf, "Pruning filters for efficient convnets," 2016, *arXiv:1608.08710*. [Online]. Available: https://arxiv.org/abs/1608.08710

[36] B. Han *et al.*, "Deep face model compression using entropy-based filter selection," in *Proc. Int. Conf. Image Anal. Process.* Cham, Switzerland: Springer, 2017, pp. 127–136.

[37] 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

[38] 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. (ICCV)*, Oct. 2017, pp. 2755–2763.

[39] A. Ashok, N. Rhinehart, F. Beainy, and K. M. Kitani, "N2N learning: Network to network compression via policy gradient reinforcement learning," 2017, *arXiv:1709.06030*. [Online]. Available: https://arxiv.org/abs/1709.06030

[40] P. Molchanov, S. Tyree, T. Karras, T. Aila, and J. Kautz, "Pruning convolutional neural networks for resource efficient inference," 2016, *arXiv:1611.06440*. [Online]. Available: https://arxiv.org/abs/1611.06440

[41] S. Hong and Y. Park, "A FPGA-based neural accelerator for small IoT devices," in *Proc. Int. SoC Design Conf. (ISOCC)*, 2017, pp. 294–295.

[42] L. Mauch and B. Yang, "A novel layerwise pruning method for model reduction of fully connected deep neural networks," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, Mar. 2017, pp. 2382–2386.

[43] J. Zhu, Z. Qian, and C.-Y. Tsui, "Lradnn: High-throughput and energy-efficient deep neural network accelerator using low rank approximation," in *Proc. 21st Asia South Pacific Design Autom. Conf. (ASP-DAC)*, 2016, pp. 581–586.

[44] G. Lacey, G. W. Taylor, and S. Areibi, "Deep learning on FPGAs: Past, present, and future," 2016, *arXiv:1602.04283*. [Online]. Available: https://arxiv.org/abs/1602.04283

[45] A. Shawahna, S. M. Sait, and A. El-Maleh, "FPGA-based accelerators of deep learning networks for learning and classification: A review," *IEEE Access*, vol. 7, pp. 7823–7859, 2019.

[46] A. Dundar, J. Jin, B. Martini, and E. Culurciello, "Embedded streaming deep neural networks accelerator with applications," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 28, no. 7, pp. 1572–1583, Jul. 2017.

[47] E. T. Jaynes, "Information theory and statistical mechanics," *Phys. Rev.*, vol. 106, no. 4, p. 620, 1957.

[48] K. Osawa and R. Yokota, "Evaluating the compression efficiency of the filters in convolutional neural networks," in *Proc. Int. Conf. Artif. Neural Netw.* Cham, Switzerland: Springer, 2017, pp. 459–466.

[49] 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.

[50] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.

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

[52] O. Russakovsky *et al.*, "ImageNet large scale visual recognition challenge," *Int. J. Comput. Vis.*, vol. 115, no. 3, pp. 211–252, Dec. 2015.

Zhen Chen (S'18) received the B.S. degree in information engineering from Xi'an Jiaotong University (XJTU) in 2016 and the M.S. degree in electronic engineering and information science from the University of Science and Technology of China (USTC) in 2019. He is currently pursuing the Ph.D. degree in electrical engineering with the City University of Hong Kong (CityU). His research interests include medical image analysis, neural network acceleration, and machine learning. He served as a TPC Member in IEEE WoWMoM CCNCPS workshop 2019 and a Reviewer for the IEEE GLOBECOM 2018.

Zhibo Chen (M'01–SM'11) received the B.Sc. and Ph.D. degrees from the Department of Electrical Engineering, Tsinghua University in 1998 and 2003, respectively. He is currently a Professor with the University of Science and Technology of China. His research interests include image and video compression, visual quality of experience assessment, immersive media computing and intelligent media computing. He has more than 100 publications and more than 50 granted EU and U.S. patent applications. He was the TPC Chair of the IEEE PCS 2019 and an Organization Committee Member of ICIP 2017 and ICME 2013, a TPC Member in IEEE ISCAS and IEEE VCIP. He is also a member of the IEEE Visual Signal Processing and Communications Committee and the IEEE Multimedia System and Applications Committee.

Jianxin Lin received the B.S. degree in electronic engineering from the University of Science and Technology of China in 2015. He is currently pursuing the Ph.D. degree with the University of Science and Technology of China. His research interests include computer vision and machine learning. He has published several articles on CVPR, TPAMI, AAAI, IJCAI, and so on. He also serves as a Reviewer for ICML-2019, NeurIPS-2019, AAAI-2020, IJCV, and so on.

Sen Liu received the B.S. degree in computer science from the Beijing University of Posts and Telecommunications, Beijing, China, in 2013. He is currently an Associate Researcher with the Institute of Advanced Technology, University of Science and Technology of China. His area of interests includes artificial intelligence, deep learning, video coding, computer vision and pattern recognition, and reinforcement learning.

Weiping Li (S'84–M'87–SM'97–F'00) received the B.S. degree in electrical engineering from the University of Science and Technology of China (USTC), Hefei, China, in 1982, and the M.S. and Ph.D. degrees in electrical engineering from Stanford University, Stanford, CA, USA, in 1983 and 1988, respectively.

In 1987, he joined Lehigh University, Bethlehem, PA, USA, as an Assistant Professor, with the Department of Electrical Engineering and Computer Science. In 1993, he was promoted to an Associate Professor with tenure. In 1998, he was promoted to a Full Professor. From 1998 to 2010, he worked in several high-tech companies with the Silicon Valley (from 1998 to 2000 Optivision Inc., Palo Alto, from 2000 to 2002, Webcast Technologies, Mountain View, from 2002 to 2008, Amity Systems, Milpitas, from 2008 to 2010, Bada Networks, Santa Clara, CA,, USA). In March 2010, he returned to USTC to serve as the Dean of the School of Information Science and Technology until July 2014. He is currently a Professor with the School of Information Science and Technology, USTC. He served as a Member of the Moving Picture Experts Group (MPEG) of the International Standard Organization (ISO) and an Editor of the MPEG-4 International Standard. He served as a Founding Member of the Board of Directors of MPEG-4 Industry Forum. He was a recipient of the Certificate of Appreciation from ISO/IEC as a Project Editor in development of an international standard in 2004, the Spira Award for Excellence in Teaching in 1992 at Lehigh University, and the First Guo Mo-Ruo Prize for Outstanding Student in 1980 at USTC. He has served as the Editor-in-Chief for the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY and a Guest Editor for the PROCEEDINGS OF THE IEEE. He was the Chair of several Technical Committees in the IEEE Circuits and Systems Society and the IEEE International Conferences and the Chair of the Best Student Paper Award Committee for SPIE Visual Communications and Image Processing Conference. He has made many contributions to international standards. His inventions on fine granularity scalable video coding and shape adaptive wavelet coding have been included in the MPEG-4 international standard. As a Technical Advisor, he also made contributions to the Chinese audio video coding standard and its applications.