Received 14 December 2023, accepted 22 January 2024, date of publication 25 January 2024, date of current version 14 February 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3358620

RESEARCH ARTICLE

Block-Wise Separable Convolutions: An Alternative Way to Factorize Standard Convolutions

YAN-JEN HUANG, HSIN-LUNG WU, AND CHING-CHEN
Department of Computer Science and Information Engineering, National Taipei University, New Taipei City 237, Taiwan

Corresponding author: Hsin-Lung Wu (hsinlung@mail.ntpu.edu.tw)

This work was supported in part by the National Science and Technology Council, Taiwan, under Grant NSTC-112-2221-E-305-004; and in part by National Taipei University under Grant 2023-NTPU-HHMS-01.

ABSTRACT In this paper, we introduce block-wise separable convolutions (BlkSConv) to replace the standard convolutions for compressing deep CNN models. First, BlkSConv expresses the standard convolutional kernel as an ordered set of block vectors each of which is a linear combination of fixed basis block vectors. Then it eliminates most basis block vectors and their corresponding coefficients to obtain an approximated convolutional kernel. Moreover, the proposed BlkSConv operation can be efficiently realized via a combination of pointwise and group-wise convolutions. Thus the constructed networks have smaller model size and fewer multiply-adds operations while keeping comparable prediction accuracy. We also develop a hyperparameter search framework based on principal component analysis (PCA) to determine a qualified hyperparameter setting of the block depth and number of basis block vectors. By this search framework, we construct networks which achieve nice prediction performance while simultaneously satisfying the constraints of model size and model efficiency. Our code, data, and models are available at https://github.com/yanjenhuang/blksconv.

INDEX TERMS Convolutional neural network, block-wise separable convolution, network architecture search.

I. INTRODUCTION

Deep Learning (DL) has been the basis of many successes in artificial intelligence, including a variety of applications in computer vision, reinforcement learning, and natural language processing. One of the most popular deep neural networks is Convolutional Neural Network (CNN). With the help of various techniques such as residual connections and batch normalization, it is easy to train deep CNNs with many layers on powerful GPUs. While large-scale CNN models have achieved great successes, they require huge computational complexity and massive storage. For example, VGG16 [1] has 138 million parameters and requires 154700 million multiply-adds operations (MAdds) to classify an image. It is a great challenge to deploy them in real-time applications, especially on devices with limited resources such as mobile phones and embedded systems. Thus, the prediction models are required to be compact and fast while keeping acceptable accuracy. The main approach to be compact is the model compression which aims at establishing a tradeoff between model efficiency and accuracy. In the area of model compression, methods to construct efficient and compact CNNs are mainly divided into two approaches: one approach is to compress trained CNNs and the other approach is to design new compact CNNs and train them from scratch. Many works based on the first approach suggested several techniques such as quantization [2], model pruning [3], [4], [5], [6], [7], Huffman coding [3], and low rank factorization [8].

Studies in the second approach explored many ways for factorizing convolutions. For instance, Szegedy et al. [9] improved GoogLeNet [10] through factorizing convolutions

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

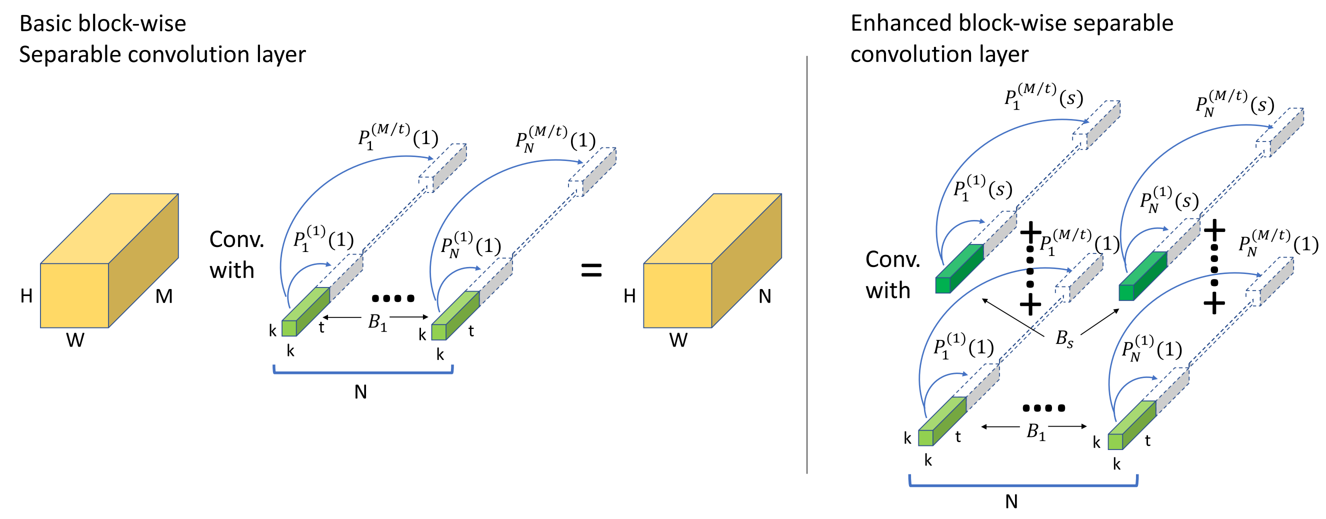

FIGURE 1. The proposed block-wise separable convolution and its enhanced version. (a) Basic block-wise Separable convolution layer. (b) Enhanced block-wise separable convolution layer.

with larger spatial filters by a two-layer convolutional architecture with smaller spatial filters. At present, most factorizing methods are usually performed via a combination of depthwise convolution, pointwise convolution, and groupwise convolution. For example, in [11], the depth-wise separable convolutions (DSCs) were proposed where the standard convolution is decomposed into a depth-wise convolution and a pointwise convolution. The ShuffleNets [12], [13] utilized pointwise group convolution with channel shuffle to decompose the standard convolution. Moreover, many lightweight models based on DSCs or groupwise convolutions such as MobileNets [14], [15], [16] and ShuffleNets [12], [13] were proposed to greatly reduce computation cost while maintaining accuracy.

In this paper, we follow the research path of the second approach and propose block-wise separable convolutions (*BlkSConv*) to replace standard convolutions. BlkSConv approximates a standard convolution as follows. A standard $k \times k \times M$ convolutional kernel can be represented as an ordered set of block vectors of size $k \times k \times t$. Since each block vector can be written as a linear combination of k^2t basis vectors of size $k \times k \times t$, this standard convolutional kernel can be viewed as an ordered set of block vectors each of which is a linear combination of k^2t basis block vectors. Then BlkSConv eliminates most basis block vectors and their corresponding coefficients to obtain an approximated convolutional kernel. As shown on the left of Figure 1, the extreme version of BlkSConv is called the basic BlkSConv where only one basis block vector is used. When carefully setting the depth of the block vector, that is the parameter t, an approximated convolution of fewer parameters can be obtained and the corresponding compact CNN has acceptable prediction performance compared to the standard convolutions. To increase the prediction accuracy of the basic BlkSConv, an enhanced version is proposed by increasing the number of basis block vectors, that is the parameter s, as shown on the right of Figure 1. However, adding too many basis block vectors will significantly increase the model size and computational cost. Thus there is a tradeoff between model efficiency/size and accuracy. To realize the full potential of the enhanced BlkSConv in trading-off model efficiency/size and accuracy, we propose a framework based on the principal component analysis to search for the hyperparameters t and s of each BlkSConv layer for the given standard convolutional network. The proposed search framework suggests a possible setting of parameters t and s such that the constructed model based on these selected hyperparameters may achieve high prediction accuracy while simultaneously satisfying the constraints of model size and model efficiency in terms of MAdds.

To summarize, our main contributions are as follows. First, we develop a new convolutional layer called *BlkSConv* to approximate the standard convolutional layer. To approximate a standard convolutional kernel, BlkSConv divides the kernel into blocks and approximates each block by a linear combination of several fixed basis block vectors. The constructed networks have small model size and fewer multiply-adds operations while maintaining acceptable prediction accuracy. Then, we also develop a search framework to determine the block depth and the number of basis block vectors such that the corresponding networks with selected hyperparameters achieve comparable prediction performance while simultaneously satisfying the constraints of model size and model efficiency. We also present experimental results to demonstrate the performance of selected BlkSConv-based CNNs based on our proposed hyperparameter search algorithm. Our results show that selected BlkSConv-based CNNs achieve competitive performance compared with the standard convolutional models for the datasets including ImageNet, CIFAR-10/100, Stanford Dogs, and Oxford Flowers.

For reader's comprehension, the major contributions of the present study are listed as follows:

- We develop BlkSConv which can approximate the standard convolution with minimal loss of accuracy.
- BlkSConv can be implemented via a combination of pointwise and group-wise convolutions.
- A PCA-based hyperparameter search algorithm is developed to determine the block depth and the number of basis block vectors when setting the BlkSConv hyperparameters.

II. RELATED WORK

Many efforts have been devoted to improve efficiency of CNNs which could be roughly divided into three categories. First, model pruning is a popular method to improve efficiency of CNNs. In [3] and [17], their methods remove redundancy in the trained CNN model by pruning connection. In [3], [18], [19], and [20], the calculation amount of the trained model is compressed via quantization. In [5], an accelerator-aware pruning algorithm is developed to generate a more regular non-zero weight pattern that fits accelerator architectures well. In [6], a saliency-adaptive sparsity learning approach is proposed to optimize weight pruning. In [7], the JointPruning method is designed for compressing point cloud neural networks. In [21], [22], [23], [24], and [25], model filters that have small contributions are removed and the corresponding trained model is fine-tuned to preserve the performance.

Second, many techniques are developed to factorize the standard convolutions. In [9], convolutions with larger spatial filters are factorized into two-layer convolutional architectures with smaller spatial filters. Through different combinations of depthwise convolution, pointwise convolution, and groupwise convolution, many well-known factorizing frameworks were developed. In [11], the depth-wise separable convolutions (DSCs) were proposed where the standard convolution is decomposed into a depth-wise convolution and a pointwise convolution. The ShuffleNets [12], [13] use pointwise group convolution with channel shuffle to decompose the standard convolution. Moreover, many lightweight models based on DSCs or groupwise convolutions such as MobileNets [14], [15], [16] and ShuffleNets [12], [13] were proposed to greatly reduce computation cost while maintaining accuracy.

Recently, neural architecture search-based methods [26], [27], [28], [29], [30] have been proposed to automatically construct network architectures. These methods search over a set of network hyperparameters including different types of convolutional layers and kernel sizes in order to find a network structure which satisfies optimization constraints such as inference speed. Major search frameworks include genetic-based methods [28] and reinforcement learning based methods [29]. These techniques were used in state-of-the-art CNN architectures such as MnasNet [26] and MobileNetV3 [14].

Convolution weights of trained CNNs are also analyzed in [31], [32], [33], and [34]. Following their analyses, several approaches toward reducing redundant weights were proposed. In [8], [35], and [36], the convolutional kernels are approximated via low-rank factorization. In [33], the kernels are analyzed via principal component analysis.

III. BLOCK-WISE SEPARABLE CONVOLUTIONS

For any natural number n, let $[n]$ denote the set $\{1, 2, \ldots, n\}$. In a standard CNN, each convolutional layer converts an input tensor I of size $M \times X \times Y$ into an output tensor O of size $N \times X \times Y$ by applying the filter kernels $F_1, F_2, \ldots, F_N$, each of size $M \times \ell \times \ell$ with odd ℓ such that, for any $x, y, j \in [X] \times [Y] \times [N]$,

$$O_{x,y,j} = \sum_{\substack{s_1,s_2 \in S \\ s_3 \in [M]}} I_{x+s_1,y+s_2,s_3} \cdot F_j(s_1, s_2, s_3). \tag{1}$$

where $S = \{z \in \mathbb{Z} : -(l-1)/2 \leq z \leq (l-1)/2\}$. During training, the weights of each kernel F_j are optimized via backpropagation. The total number of weight parameters to be optimized in each kernel F_j is $\ell^2 \cdot M$. In the subsequent work, we propose a framework to reduce the number of parameters of the standard convolutions while preserving its prediction performance. Then, in order to implement our new framework, we adopt a combination of pointwise and group-wise convolutions to efficiently realize the reduced convolutions. Combining these ideas, we introduce block-wise separable convolutions, denoted by *BlkSConv*. However, to generate a BlkSConv-based models, many hyperparameters should be determined for keeping prediction performance, model size, and model efficiency. Thus, we also propose an efficient hyperparameter search algorithm to select hyperparameters satisfying the given model constraints.

A. EXPRESSING A STANDARD CONVOLUTION VIA A LINEAR COMBINATION OF BLOCK VECTORS

In this section, we propose block-wise separable convolutions. First, each convolutional kernel F_j of size $M \times \ell \times \ell$ can be expressed as a concatenation of M/t blocks $Q_j^{(1)}, Q_j^{(2)}, \ldots, Q_j^{(M/t)}$ each of size $\ell \times \ell \times t$ where $Q_j^{(k)}(x, y, z) = F_j(x, y, z + (k-1)t)$ for any $x, y, z \in [X] \times [Y] \times [t]$. We call t the block depth. Let $\{B_1, B_2, \ldots, B_{t\ell^2}\}$ be a set of basis block vectors. Each $Q_j^{(k)}$ can be expressed uniquely as a linear combination of $B_1, B_2, \ldots, B_{t\ell^2}$, that is, there exist $t\ell^2$ values $P_j^{(k)}(i) \in \mathbb{R}$ such that $Q_j^{(k)} = \sum_{i=1}^{t\ell^2} P_j^{(k)}(i) \cdot B_i$. In practice, $t\ell^2$ may be large. In order to reduce the model size, we require that the number of basis block vectors is fewer than or equal to a fixed number s with $s < t\ell^2$. Now each $Q_j^{(k)}$ is replaced by the following linear combination of $B_1, \ldots, B_s$, that is $\widehat{Q}_j^{(k)} = \sum_{i=1}^{s} P_j^{(k)}(i) \cdot B_i$. The corresponding convolutional kernel $\widehat{F}_j$ is the concatenation of M/t blocks $\widehat{Q}_j^{(1)}, \ldots, \widehat{Q}_j^{(M/t)}$. Therefore,

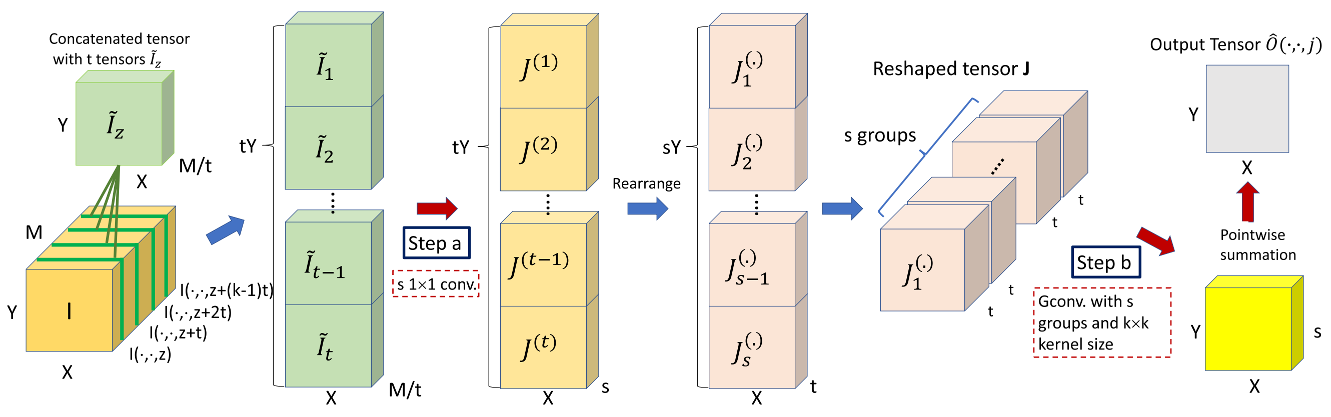

FIGURE 2. **Flowchart of the block-wise separable convolution where Gconv. means the group-wise convolution.**

the corresponding output tensor is

$$\widehat{O}_{x,y,j} = \sum_{\substack{s_1,s_2\in S\\ s_3\in[M]}} I_{x+s_1,y+s_2,s_3}\cdot \widehat{F}_j(s_1,s_2,s_3). \tag{2}$$

By Equation 2, the number of weight parameters in BlkSConv is $s\cdot(t\cdot\ell^2+\frac{M}{t})$. To significantly reduce model size, we set $s = 1$. The left of Figure 1 illustrates the operation of BlkSConv when $s = 1$. In order to achieve the minimal model size, t can be set as $\sqrt{M}/\ell$ and the number of parameters becomes $2\ell\sqrt{M}$ while the parameter number of the standard and 1×1 pointwise convolutions are $M\ell^2$ and M, respectively. Thus, the constructed BlkSConv-based CNNs have smaller model size than existing CNN models. Take the ResNet34 [37] as an example where, in the last stage of the ResNet-34, the convolutional kernel size is 3×3 and the channel size is 512, that is $\ell = 3$ and $M = 512$. In this case, the ratio between the parameter size of the BlkSConv-based convolutions and the parameter size of the standard convolutions is approximately 0.0295.

However, the prediction performance of the BlkSConv-based CNN with the smallest model size is usually worse than that of the standard CNNs. To increase accuracy, the number of basis block vectors should be increased, that is $s > 1$. The right of Figure 1 illustrates the operation of BlkSConv when $s > 1$. In this case, the number of parameters becomes $2s\ell\sqrt{M}$. Let us take convolutions in the last stage of ResNet-34 as examples. We set $t = 4$ in the BlkSConv. Now the ratio between the parameter size of the BlkSConv-based convolutions and the parameter size of the standard convolutions is approximately 0.0356. Thus we can add at least 5 basis block vectors to increase prediction accuracy. In this case, the ratio between the parameter size of the BlkSConv-based convolutions with 5 basis block vectors and the parameter size of the standard convolutions is approximately 0.178. In the experimental section, we demonstrate that the BlkSConv-based convolutions with few basis block vectors have prediction performance as well as the standard convolutions on ImageNet or even outperform the standard convolutions on several datasets when the backbone CNNs are ResNets.

The next problem is the computational efficiency of BlkSConv. If we compute the kernel $\widehat{F}_j$ first and perform a regular convolution according to the kernel $\widehat{F}_j$, then it is obvious that the computational cost is larger than the cost for just performing a standard convolution. We will address this problem in the subsequent section.

B. IMPLEMENTATION OF BLKSCONV VIA A COMBINATION OF POINTWISE AND GROUP-WISE CONVOLUTIONS

In this section, we propose an efficient implementation method to realize BlkSConv. The flowchart of the proposed implementation is illustrated in Figure 2. To derive an efficient implementation for BlkSConv operation, we rewrite Equation 2 as follows.

$$\begin{aligned}
\widehat{O}_{x,y,j} &= \sum_{\substack{s_1,s_2\in S\\ z\in[t]\\ k\in[M/t]}} I_{x+s_1,y+s_2,z+(k-1)t}\cdot \widehat{Q}_j^{(k)}(s_1,s_2,z)\\
&= \sum_{\substack{s_1,s_2\in S\\ z\in[t]\\ k\in[M/t]}} I_{x+s_1,y+s_2,z+(k-1)t}\cdot \sum_{i\in[s]} P_j^{(k)}(i)\cdot B_i(s_1,s_2,z)\\
&= \sum_{\substack{i\in[s]\\ s_1,s_2\in S\\ z\in[t]}} B_i(s_1,s_2,z)\\
&\quad\times \underbrace{\sum_{k\in[M/t]} P_j^{(k)}(i)\cdot \overbrace{I_{x+s_1,y+s_2,z+(k-1)t}}^{\widetilde{I}_z(x+s_1,y+s_2,k)}}_{J^{(z)}(x,y,i):\ \text{a pointwise convolution of } \widetilde{I}_z} .
\end{aligned} \tag{3}$$

Let $\widetilde{I}_z(x,y,k)$ be a tensor of size $X\times Y\times M/t$ defined by $\widetilde{I}_z(x,y,k) \triangleq I(x,y,z+(k-1)t)$. We define $J^{(z)}(x,y,i) \triangleq \sum_{k=1}^{M/t} P_j^{(k)}(i)\cdot \widetilde{I}_z(x+s_1,y+s_2,k)$ which is a pointwise convolution of $\widetilde{I}_z$. Next, we define $J_i(x,y,z) \triangleq J^{(z)}(x,y,i)$ and let J be the reshaped tensor which is the concatenation of $J_1, \ldots, J_s$, that is $J(x, y, z + (i-1)t) = J_i(x, y, z)$. Now Equation 3 can be rewritten as

$$\widehat{O}_{x,y,j} = \sum_{\substack{i\in[s] \\ s_1,s_2\in S \\ z\in[t]}} B_i(s_1, s_2, z) \times J(x + s_1, y + s_2, z + (i-1)t). \quad (4)$$

Finally, Equation 4 is just a group-wise convolution of the tensor J with s groups.

Let us compute the computational cost (MAdds) of the implementation for BlkSConv. By Equation 3 (Step a in Figure 2), the computational cost of s pointwise convolutions on the concatenation of $\widetilde{I}_1, \ldots, \widetilde{I}_t$ is $sXYM$. In addition, by Equation 4 (Step b in Figure 2), the computational cost of the group-wise convolution on the tensor J is $sXYt\ell^2$. Finally, the computational cost of the pointwise summation in the last step is sXY. The total MAdds of a BlkSConv operation is $sXY(M + t\ell^2 + 1)$ while the MAdds of a standard convolution is $XYM\ell^2$. Again, let us take convolutions in the last stage of ResNet-34 as examples. We set $s = 5$ and $t = 4$ as the hyperparameters of the BlkSConv-based convolution. Now the ratio between the MAdds of a BlkSConv-based convolution and the Madds of a standard convolution is approximately 0.595. Thus the proposed BlkSConv operation is much more efficient than the standard convolution in practical cases.

C. HYPERPARAMETER SEARCH VIA PRINCIPAL COMPONENT ANALYSIS

Designing a BlkSConv-based CNN involves hyperparameters including the block depth and the number of basis block vectors in each convolutional layer that affect the performance of the corresponding CNN model. To realize an efficient BlkSConv-based CNN, we conduct a hyperparameter search algorithm based on principal component analysis of trained CNNs. Given a trained CNN, the algorithm generates the block depth and the number of basis block vectors for each standard convolutional layer of the trained CNN in the following way. First, for each individual $\ell \times \ell \times M$ kernel K of the trained CNN where we assume that $M = 2^\alpha$ for some $\alpha \in \mathbb{N}$, the kernel K is partitioned into M/t block vectors $B_1, B_2, \ldots, B_{M/t}$ each of size $\ell \times \ell \times t$ with $t \in \{1, 2, \ldots, 2^\beta\}$ for some integer $\beta < \alpha$. Next, we perform principal component analysis (PCA) on the set $\{B_1, B_2, \ldots, B_{M/t}\}$. Then, for a fixed integer γ and for each $q \in \{1, 2, \ldots, \gamma\}$, the algorithm computes the variance $V_{t,q}$ of the kernel K which is explained by the first q principal components PC1,PC2,..,PCq. In addition, let $\mathrm{CC}_{t,q}$ and $\mathrm{MS}_{t,q}$ denote the MAdds and the model size of the BlkSConv under the setting that the block depth is t and the number of basis block vectors is q, respectively. Note that the MAdds and the model size of the standard convolution is exactly $\mathrm{CC}_{M,1}$ and $\mathrm{MS}_{M,1}$, respectively. After computing all $\mathrm{V}_{t,q}$, $\mathrm{CC}_{t,q}$, and $\mathrm{MS}_{t,q}$, the algorithm generates the feasible set:

$$H_{\alpha_v,\alpha_c,\alpha_s} = \{(t, q) : \substack{\alpha_v \le V_{t,q}, \\ \mathrm{CC}_{t,q} \le \alpha_c \mathrm{CC}_{M,1}, \\ \mathrm{MS}_{t,q} \le \alpha_s \mathrm{MS}_{M,1}}\} \quad (5)$$

for fixed positive constants $\alpha_v, \alpha_c, \alpha_s \in (0, 1)$. Finally, the algorithm chooses the hyperparameter (t, q) from $H_{\alpha_v,\alpha_c,\alpha_s}$ according to the computational cost or the model size.

On one hand, note that the goal of BlkSConv is to maintain the prediction performance of the trained standard CNN. In general, the prediction accuracy is proportional to the model size of the constructed CNN. Therefore, in this sense, we choose the hyperparameters $(\hat{t}, \hat{q})$ from $H_{\alpha_v,\alpha_c,\alpha_s}$ such that the constructed BlkSConv has the largest parameter size, that is

$$(\hat{t}, \hat{q}) = \arg \max_{(t,q)\in H_{\alpha_v,\alpha_c,\alpha_s}} \mathrm{MS}_{t,q}. \quad (6)$$

One can expect that the generated BlkSConv-based CNN has nice prediction performance compared to the original CNN with standard convolutions.

On the other hand, one of the advantage of BlkSConv operations is that BlkSConv can greatly reduce the model size of the original standard CNN. Thus, in this sense, we can select the hyperparameters $(\tilde{t}, \tilde{q})$ from $H_{\alpha_v,\alpha_c,\alpha_s}$ such that the constructed BlkSConv has the smallest parameter size, that is

$$(\tilde{t}, \tilde{q}) = \arg \min_{(t,q)\in H_{\alpha_v,\alpha_c,\alpha_s}} \mathrm{MS}_{t,q}. \quad (7)$$

However, the prediction performance may degrade when the parameter size of the BlkSConv-based model decreases. We will demonstrate in the experimental section that the BlkSConv-based CNNs generated according to Equation 7 also have acceptable prediction accuracy compared to the standard CNNs.

In summary, both Equation 6 and Equation 7 provide ways to determine hyperparameters from the feasible set $H_{\alpha_v,\alpha_c,\alpha_s}$ such that corresponding BlkSConv-based CNNs have smaller model size and fewer multiply-adds operations than the original CNN with standard convolutions.

Finally, let us consider the extreme case that two constants β and γ are set by $\beta = 0$ and $\gamma = 1$. Let us further set the search parameter $\alpha_v = 0$. Under this restricted search condition, the cardinality of the feasible set H_{0,α_c,α_s} is always 1. Thus the outputs of Equation 6 and Equation 7 are the same. In fact, the resulting BlkSConv-based CNN is exactly the same as the CNN where the standard convolutions are replaced by the blueprint separable convolutions previously developed in [33].

IV. EXPERIMENTS

In this section, we evaluate BlkSConv-based models and the proposed hyperparameter architecture search algorithm (HSA) on large-scale, small-scale, and fine-grained datasets. The experimental algorithms are implemented on NVIDIA Tesla V100 GPUs and in Pytorch and Scikit-learn.

TABLE 1. ResNet architectures used in the high resolution images, ImageNet, Stanford Dogs, and Oxford 102 Flowers. ResNet-10 (L=1), ResNet-18 (L=2), ResNet-26 (L=3).

Layer Name	Output Size	Modules	Apply HSA
conv1	(112, 112, 64)	7 × 7, 64, stride 2	No
max pool	(56, 56, 64)	3 × 3, stride 2	|
conv2_x	(56, 56, 64)	$\begin{bmatrix} 3\times3, & 64 \\ 3\times3, & 64 \end{bmatrix} \times L$	No
conv3_x	(28, 28, 128)	$\begin{bmatrix} 3\times3, & 128 \\ 3\times3, & 128 \end{bmatrix} \times L$	Yes
conv4_x	(14, 14, 256)	$\begin{bmatrix} 3\times3, & 256 \\ 3\times3, & 256 \end{bmatrix} \times L$	Yes
conv5_x	(7, 7, 512)	$\begin{bmatrix} 3\times3, & 512 \\ 3\times3, & 512 \end{bmatrix} \times L$	Yes
avg pool	(1, 1, 512)	7 × 7	|
fc layer	1000	512 × 1000 fc	|

A. HYPERPARAMETER SEARCH DETAILS

First, we apply the PCA-based hyperparameter search algorithm (HSA) developed in Section III-C on several variants of ResNet models [37]. In the first part, we consider the large-scale classification scenarios. Several standard ResNets are trained on ImageNet [38] first and their architectures are shown in Table 1. The HSA for searching BlkSConv architectures is only applied to conv3_x, conv4_x, conv5_x layers of these standard ResNets. Next, the search hyperparameters α_v, α_c, α_s are set as 0.5 or 0.75. It is possible that the feasible set $H_{\alpha_v,\alpha_c,\alpha_s}$ is empty. In this case, the corresponding standard convolutional layer is not replaced. Moreover, the proposed HSA has two selection strategies: one is based on the largest parameter size, denoted by $SS =$ max, and the other is based on the smallest parameter size, denoted by $SS =$ min as shown in Table 2. We use sitj to denote the architecture of the selected BlkSConv where i means the number of basis block vectors and j is the depth of the blocks. In the case that α_v is large, it often requires many principal components to accumulate enough explained variance and thus this causes large numbers of parameters or MAdds. Therefore, the feasible set $H_{\alpha_v,\alpha_c,\alpha_s}$ is probably empty when we further require small α_c and α_s. On the other hand, the parameter α_v cannot be too small because the prediction performance of the network is highly proportional to the amount of the accumulated variance as discussed in Section III-C where we will demonstrate it in the ablation study of this section. We present the results for ResNet-18 and ResNet-26 on ImageNet under the setting that $\alpha_v = 0.5$ which are shown in Table 2.

In the second part, we consider the small-scale classification on CIFAR10/100 [39]. The standard ResNet-20 and ResNet-56 are used as the experimental models where their BlkSConv-based convoluational architectures are shown in Table 3.

B. LARGE-SCALE CLASSIFICATION: IMAGENET

To evaluate the performance of BlkSConv-based models in large-scale recognition, we conduct experiments on ImageNet [38]. ImageNet contains nearly 1.3M training images and 50,000 testing images. We augment the images via random resized crop to 224px and random horizontal flip with 50% probability. As suggested in [37], the initial learning rate, the momentum, and the weight decay are set to 0.1, 0.9, and 10^{-4}, respectively. The training takes 100 epochs and the learning rate is scheduled to decay by a factor of 0.1 at epochs 30, 60, and 90. We use SGD optimizer with batch size 256 to train ResNet-10, ResNet-18, and ResNet-26. Besides ResNets, VGG and AlexNet are also used in our experiment. We use SGD optimizer with batch size 128 to train VGG11-BN, VGG13-BN, and AlexNet.

On one hand, let us focus the case that $\alpha_v = 0.5$ and $SS =$ max in Table 2. The prediction accuracies of the selected BlkSConv-based models and the standard model are close within 1%. It confirms our expectation that BlkSConv-based models have smaller parameter sizes and fewer MAdds than standard models while preserving prediction performance if the proposed HSA adopts a selection strategy based on the maximum parameter size.

On the other hand, let us consider the case that $\alpha_v = 0.5$ and $SS =$ min in Table 2. The parameters and MAdds of the BlkSConv-based models are only 12.6%

TABLE 2. The standard ResNet-18 and ResNet-26 have 10.8M parameters/ 1213.8M MAdds and 17.03M parameters / 1907.4M MAdds, respectively. The P-rat. and M-rat. denote ratios of the parameter size and MAdds of BlkSConvs compared with those of standard convolutions, respectively.

ImageNet	ResNet-18		ResNet-26	
$(\alpha_v, \alpha_c, \alpha_s, SS)$	Acc	P/M-rat.	Acc	P/M-rat.
(.5, .50, .50, max)	69.9	.40 / .46	72.0	.42 / .46
(.5, .75, .75, max)	69.7	.60 / .65	72.3	.63 / .67
(.5, .50, .50, min)	67.5	.12 / .27	69.9	.12 / .26
(.5, .75, .75, min)	67.5	.12 / .29	69.9	.12 / .29
Standard	**70.7**	|	**72.6**	|

TABLE 3. ResNet architectures used in the smale-scale classification, CIFAR10/100. ResNet-20 (L=3), ResNet-56 (L=9).

Layer Name	Output Size	Modules	Apply HSA
conv1	(32, 32, 16)	3 × 3, 16	No
conv2_x	(16, 16, 16)	$\begin{bmatrix} 3\times3, & 16 \\ 3\times3, & 16 \end{bmatrix} \times L$	No
conv3_x	(8, 8, 32)	$\begin{bmatrix} 3\times3, & 32 \\ 3\times3, & 32 \end{bmatrix} \times L$	No
conv4_x	(4, 4, 64)	$\begin{bmatrix} 3\times3, & 64 \\ 3\times3, & 64 \end{bmatrix} \times L$	Yes
avg pool	(1, 1, 64)	4 × 4	|
fc layer	10(or 100)	64×10 (or 100) fc	|

TABLE 4. Comparison among the BlkSConv-based and Standard ResNet on ImageNet and CIFAR. For brief expression, we denote (0.5, 0.5, 0.5, max / min) by (·, max / min).

Models on ImageNet	Acc	Params	MAdds
ResNet-10 standard	63.38	4.64M	520M
ResNet-18(·,max)	69.92	4.39M	560M
ResNet-26(·,min)	**69.97**	**2.13M**	**509M**
Models on CIFAR100	Acc	Params	MAdds
ResNet-20 standard	67.99	202K	12.97M
ResNet-20(·,max)	67.07	**72K**	**5.04M**
ResNet-56(·,min)	**69.99**	149K	10.61M

TABLE 5. Comparison among ResNets, VGG, and AlexNet on ImageNet.

Model	Acc	Params	MAdds
ResNet-18	**70.72**	10.8M	1.2G
ResNet-18(·,max)	69.92	**4.3M**	**560M**
ResNet-26	**72.60**	17.03M	1.9G
ResNet-26(·,max)	72.03	**7.2M**	**880M**
VGG11-BN	**70.11**	9.22M	7.5G
VGG11-BN(·,max)	65.25	**4.71M**	**5.5G**
VGG13-BN	**70.18**	9.41M	11.2G
VGG13-BN(·,max)	65.72	**4.42M**	**7.7G**
AlexNet	50.64	2.46M	656M
AlexNet(·,max)	**57.50**	**1.31M**	**461M**

TABLE 6. Performance results for BlkSConv-based ResNet-56 on CIFAR10/100. The standard ResNet-56 has 645K parameters and 41.28M MAdds.

ResNet-56	CIFAR 10		CIFAR 100	
$(\alpha_v, \alpha_c, \alpha_s, SS)$	Acc	P/M-rat.	Acc	P/M-rat.
(.5, .5, .5, max)	93.37	.37 / .38	70.63	.37 / .38
(.5, .5, .5, min)	93.32	.23 / .24	69.99	.23 / .25
Standard	93.21	|	70.99	|

TABLE 7. Performance comparison among BlkSConv-based and the standard ResNet-18 models.

ResNet-18	Stanford Dogs		Flowers	
$(\alpha_v, \alpha_c, \alpha_s, SS)$	Acc	P/M-rat.	Acc	P/M-rat.
(.5, .5, .5, max)	53.00	.53 / .53	65.54	.53 / .53
(.5, .5, .5, min)	53.15	.32 / .40	65.28	.48 / .51
Standard	52.43	|	62.23	|

and 29.8% of the standard model while the gap of their prediction accuracies is about 3%. We adopt an interesting way based on restricting the parameter size and MAdds to interpret the advantage of the generated BlkSConv-based models where the selection strategy *SS* is set as min. We also compare the standard ResNet-10, the BlkSConv-based ResNet-18, and the BlkSConv-based ResNet-26 in Table 4 where the parameter sizes or MAdds of three given models are similar. The BlkSConv-based ResNet-26 with parameter (0.5, 0.5, 0.5, min) and the BlkSConv-based ResNet-18 with parameter (0.5, 0.5, 0.5, max) greatly outperform the standard ResNet-10 where both the BlkSConv-based models lead to an accuracy gain of at least 6.5%. In addition, the BlkSConv-based ResNet-26 with parameter (0.5, 0.5, 0.5, min) only has half the parameter size of the BlkSConv-based ResNet-18 with parameter (0.5, 0.5, 0.5, max).

Table 5 shows comparison results for ResNets, VGG and AlexNet where their HSA parameters are all set as (0.5, 0.5.0.5, max). For VGGs, the parameters and MAdds of the BlkSConv-based models are 51% and 73% of the standard model while the gap of their prediction accuracies is about 5%. The large number of parameters and MAdds are caused since many feasible sets $H_{0.5,0.5,0.5}$ generated by HSA are empty. For AlexNet, the parameters and MAdds of the BlkSConv-based models are 53% and 70% of the standard model while the BlkSConv-based AlexNet has a great accuracy gain of 7%.

C. SMALL-SCALE CLASSIFICATION: CIFAR 10/100

Both CIFAR10 and CIFAR100 contain 50,000 training images and 10,000 testing images. They are all 32×32 colored images. We conduct experiments with modified ResNet-20 and ResNet-56 on small-scale image classification. The training images are first augmented by random horizontal flips and random shifts by up to 4px to prevent overfitting. As suggested in [37], the training takes 200 epochs. We use SGD as the optimizer and the batch size is 128. The initial learning rate, the momentum, and the weight decay are set to 0.1, 0.9, and 10^{-4}, respectively. The learning rate is scheduled to decay by a factor of 0.1 at epochs 100, 150, and 180.

The performance results on CIFAR 10/100 are shown in Table 6. The BlkSConv-based models have much smaller model sizes and fewer MAdds than the standard model while all BlkSConv-based variants outperform the standard model on CIFAR 10 and have comparable accuracy on CIFAR 100. In the bottom of Table 4, the BlkSConv-based ResNet-20 with (0.5, 0.5, 0.5, max) and the standard ResNet-20 both have a comparable accuracy while the BlkSConv-based ResNet-20 model is compressed 64% of the parameter size and MAdds is decreased 61% compared to the standard ResNet-20 model. Furthermore, the BlkSConv-based ResNet-56 with (0.5, 0.5, 0.5, min) and the standard ResNet-20 model both have similar parameter sizes and MAdds while the BlkSConv-based ResNet-56 model has an accuracy gain of 2%.

D. FINE-GRAINED CLASSIFICATION

We conduct experiments for fine-grained recognition on two datasets Stanford Dogs [40] and Oxford 102 Flowers [41].

For the experimental setup, the standard ResNet-18 and its BlkSConv-variants are all trained from scratch by augmenting data through random crops, horizontal flips, and random gamma transform. We use SGD as the optimizer and the initial learning rate, the momentum, and the weight decay are set to 0.1, 0.9, and 10^{-4}, respectively. The number of epochs is 200, and the learning rate is scheduled to decay at epochs 100, 150, and 200 by a factor of 0.1. The proposed BlkSConv-based ResNet-18 models significantly outperform the standard ResNet-18 model both on Stanford Dogs and Oxford 102 Flowers as shown in Table 7.

E. ABLATION STUDY I: NECESSITY TO HAVE LARGE EXPLAINED VARIANCE

Here, we demonstrate how the variance hyperparameter α_v affects the prediction accuracy of BlkSConv-based CNNs. We use ResNet-18 as the experimental model. After training the standard ResNet-18 on Stanford Dogs, the next goal is to find several BlkSConv-variants of ResNet-18 all of which have different explained variances such that their accuracies

TABLE 8. Results on Stanford Dogs for different explained variances, α_v with $\alpha_c = \alpha_s = 0.75$ and $SS = \min$.

α_v	Acc	P-rat.	M-rat.
0.0	50.91	0.0397	0.1781
0.1	50.49	0.0449	0.1777
0.2	51.04	0.0767	0.2458
0.3	52.83	0.1074	0.2377
0.4	52.03	0.2751	0.4684
0.5	**53.61**	0.3171	0.4611
Standard	52.43	10.8M	1213.8M

TABLE 9. Performance comparison among BlkSConv-based and the standard CNN models without using HSA. All models are trained from scratch.

	Stanford Dogs		Flowers	
Model	Standard	*s4t1*	Standard	*s4t1*
ResNet-18	52.43	53.00	62.23	65.54
VGG11-BN	38.67	52.26	48.17	67.47
VGG13-BN	44.81	51.86	39.95	62.49

can be compared. Note that $H_{a,0.75,0.75} \subseteq H_{b,0.75,0.75}$ for any a, b with $a \geq b$. Based on this observation, the model in $H_{b,0.75,0.75}$ which has the smallest parameter size is likely to have a small explained variance as well. Therefore, the selection strategy of the proposed HSA is set by $SS = \min$ in order to select several BlkSConv-based ResNet-18 models with different explained variances. Now we apply the proposed HSA to the trained ResNet-18 under six search hyperparameters $\{(\alpha_v, 0.75, 0.75, \min) : \alpha_v = 0.0, 0.1, 0.2, \ldots, 0.5\}$. The comparison result is shown in Table 8. It can be seen that the accuracy of the BlkSConv-based model is greater than that of the standard model only when the variance hyperparameter α_v is large enough, that is $\alpha_v \geq 0.5$.

F. ABLATION STUDY II: BLKSCONV-BASED MODELS WITHOUT HSA

The proposed HSA framework has a limitation where it has to start from acceptable pre-trained weights of standard convolutional layers. When standard CNN models are trained from scratch on a small dataset, the models with the trained convolutional weights often have bad performance. This makes the generated BlkSConv-based models have low prediction accuracy as well. To overcome it, we suggest to set each BlkSConv architecture by *s4t1*. This helps us construct BlkSConv-based CNN architectures without using HSA. Since both Stanford Dogs and Oxford 102 Flowers are small datasets, we conduct experiments to compare the prediction performance between the standard and the BlkSConv-based models trained on these two datasets. The results are shown in Table 9. After training from scratch, all BlkSConv-based models have better prediction performance than corresponding standard CNN models.

Next, we use three BlkSConv modules including *s1t1*, *s1t2*, and *s2t1* to replace the convolutional modules used in MobileNetV2 where we denote the modified models as BlkNet-sitj. The inverted residuals structure is used in MobileNetV2, which has three operators as shown in Figure 3(a). We replace the modules inside red-dashed box with our BlkSConv to construct our BlkNet-sitj. The BlkSConv-based inverted residuals structure is shown in Figure 3(b). We compare the generated BlkNets with MobileNetV2 on ImageNet and the comparison results are shown in Table 10. BlkNet-s1t1 slightly outperforms MobileNetV2 while the parameter size of BlkNet-s1t1 is slightly smaller than that of MobileNetV2. It can be seen that BlkNet-s1t2 preserves prediction performance and MAdds of BlkNet-s1t1 and has much less parameters than BlkNet-s1t1. Finally, it satisfies the expectation that BlkNet-s2t1 has much better prediction accuracy than other models but its parameter size and MAdds are larger than other models.

TABLE 10. Performance comparison among BlkNets and MobileNetV2 on ImageNet.

Models	Acc	Params	MAdds
MobileNetV2(x1.0)	65.84	3.50M	320.2M
BlkNet-s1t1	66.17	3.44M	354.4M
BlkNet-s1t2	65.48	2.97M	357.7M
BlkNet-s2t1	**68.03**	4.41M	522.7M

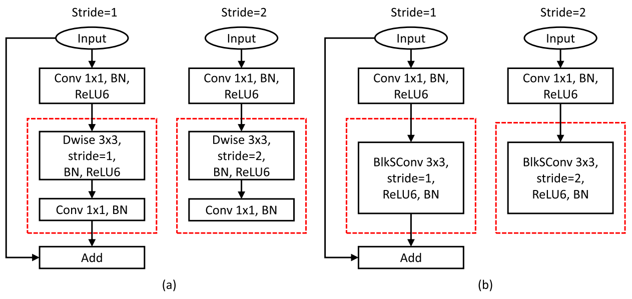

FIGURE 3. The inverted residuals structures used in MobileNetV2 and BlkNets. Modules of MobileNetV2 inside red-dashed box are replaced with our BlkSConv modules. (a) The inverted residuals structure used in MobileNetV2. (b) The BlkSConv-based inverted residuals structure used in BLKNet.

V. DISCUSSION AND FUTURE WORK

As mentioned in Section I, the main purpose of our proposed BlkSConv is to replace standard convolutions without sacrificing prediction accuracy and then to reduce model size and computational load of the standard CNNs simultaneously. Thus, for a specific deep learning task, it is possible to replace the network's standard convolutions with BlkSConvs when deploying a CNN with high task accuracy on constrained hardware. It allows for substantial memory savings with minimal loss of accuracy.

We take lightweight age estimation as an example. In [42], several standard CNNs are proposed for the task of age estimation. However, the model sizes of these standard CNNs are too large to be lightweight. In [42], the authors suggested to replace the standard convolutions of these standard CNNs by the separable convolutions in order to significantly reduce the number of parameters. Nevertheless, the prediction accuracy of their constructed lightweight CNN based on separable convolutions is not preserved compared to the standard CNN. Therefore, it is interesting to apply our proposed BlkSConvs to the standard CNNs constructed in [42] so as to preserve the prediction accuracy and reduce model size simultaneously. We leave it as our future work.

In addition, there is an important issue on the application of BlkSConv. As mentioned in Section IV-F, the PCA-based HSA framework has to start from pre-trained weights of standard convolutional layers. However, it is very time-consuming to train a BlkSConv network via the PCA-based HSA. This may restrict the application of BlkSConv. To address this problem, we suggest to set each BlkSConv architecture by several specific hyperparameters shown in Section IV-F and then to train the network from the scratch. This gives a practical solution for training a BlkSConv-based network without HSA. It makes BlkSConv more applicable.

VI. CONCLUSION

In this paper, we introduce the block-wise separable convolutions (BlkSConv) to replace standard convolutions. An efficient implementation of the BlkSConv operation via a combination of pointwise and group-wise convolutions is also given. Moreover, we also propose an efficient hyperparameter search algorithm based on principal component analysis in order to select an optimal BlkSConv-based convolutional network under certain constraints on model size and model efficiency. Finally, the experimental results demonstrate the advantage of the BlkSConv-based CNN models selected by the proposed hyperparameter search algorithm.

REFERENCES

[1] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014, *arXiv:1409.1556*.

[2] 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.*, Jun. 2017, pp. 4820–4828.

[3] 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*.

[4] A. See, M.-T. Luong, and C. D. Manning, "Compression of neural machine translation models via pruning," 2016, *arXiv:1606.09274*.

[5] H.-J. Kang, "Accelerator-aware pruning for convolutional neural networks," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 30, no. 7, pp. 2093–2103, Jul. 2020.

[6] J. Shi, J. Xu, K. Tasaka, and Z. Chen, "SASL: Saliency-adaptive sparsity learning for neural network acceleration," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 31, no. 5, pp. 2008–2019, May 2021.

[7] J. Guo, J. Liu, and D. Xu, "JointPruning: Pruning networks along multiple dimensions for efficient point cloud processing," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 32, no. 6, pp. 3659–3672, Jun. 2021.

[8] M. Jaderberg, A. Vedaldi, and A. Zisserman, "Speeding up convolutional neural networks with low rank expansions," 2014, *arXiv:1405.3866*.

[9] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2016, pp. 2818–2826.

[10] C. Szegedy, W Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2015, pp. 1–9.

[11] L. Sifre and S. Mallat, "Rigid-motion scattering for texture classification," *Int. J. Comput. Vis.*, vol. 19, no. 3559, pp. 501–515, Mar. 2014.

[12] N. Ma, X. Zhang, H.-T. Zheng, and J. Sun, "ShuffleNet V2: Practical guidelines for efficient CNN architecture design," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, Oct. 2018, pp. 116–131.

[13] 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.*, Jun. 2018, pp. 6848–6856.

[14] A. Howard, M. Sandler, G. Chu, LC. Chen, B. Chen, M. Tan, W. Wang, Y. Zhu, R. Pang, V. Vasudevan, and Q. V. Le, "Searching for MobileNetV3," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 1314–1324.

[15] 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 applications," 2017, *arXiv:1704.04861*.

[16] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "MobileNetV2: Inverted residuals and linear bottlenecks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 4510–4520.

[17] L. Zhu, R. Deng, M. Maire, Z. Deng, G. Mori, and P. Tan, "Sparsely aggregated convolutional networks," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2018, pp. 186–201.

[18] E. Park, S. Yoo, and P. Vajda, "Value-aware quantization for training and inference of neural networks," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2018, pp. 580–595.

[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.*, 2016, pp. 525–542.

[20] D. Zhang, J. Yang, D. Ye, and G. Hua, "LQ-Nets: Learned quantization for highly accurate and compact deep neural networks," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2018, pp. 365–382.

[21] Y. He, G. Kang, X. Dong, Y. Fu, and Y. Yang, "Soft filter pruning for accelerating deep convolutional neural networks," 2018, *arXiv:1808.06866*.

[22] Y. He, X. Zhang, and J. Sun, "Channel pruning for accelerating very deep neural networks," in *Proc. IEEE Int. Conf. Comput. Vis.*, Oct. 2017, pp. 1389–1397.

[23] H. Li, A. Kadav, I. Durdanovic, H. Samet, and H. P. Graf, "Pruning filters for efficient ConvNets," 2016, *arXiv:1608.08710*.

[24] 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. (ICCV)*, Oct. 2017, pp. 5058–5066.

[25] P. Singh, V. K. Verma, P. Rai, and V. P. Namboodiri, "Leveraging filter correlations for deep model compression," in *Proc. IEEE Winter Conf. Appl. Comput. Vis. (WACV)*, Mar. 2020, pp. 824–833.

[26] M. Tan, B. Chen, R. Pang, V. Vasudevan, M. Sandler, A. Howard, and Q. Le, "MnasNet: Platform-aware neural architecture search for mobile," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2019, pp. 2820–2828.

[27] B. Wu, X. Dai, P. Zhang, Y. Wang, F. Sun, Y. Wu, Y. Tian, P. Vajda, Y. Jia, and K. Keutzer, "FBNet: Hardware-aware efficient ConvNet design via differentiable neural architecture search," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 10734–10742.

[28] L. Xie and A. Yuille, "Genetic CNN," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 1379–1388.

[29] B. Zoph and Q. V. Le, "Neural architecture search with reinforcement learning," 2016, *arXiv:1611.01578*.

[30] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le, "Learning transferable architectures for scalable image recognition," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 8697–8710.

[31] M. Denil, B. Shakibi, L. Dinh, M. Ranzato, and N. De Freitas, "Predicting parameters in deep learning," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 26, 2013, pp. 148–2156.

[32] J. Guo, Y. Li, W. Lin, Y. Chen, and J. Li, "Network decoupling: From regular to depthwise separable convolutions," 2018, *arXiv:1808.05517*.

[33] D. Haase and M. Amthor, "Rethinking depthwise separable convolutions: How intra-kernel correlations lead to improved MobileNets," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 14600–14609.

[34] W. Shang, K. Sohn, D. Almeida, and H. Lee, "Understanding and improving convolutional neural networks via concatenated rectified linear units," in *Proc. Int. Conf. Mach. Learn.*, Jun. 2016, pp. 2217–2225.

[35] E. 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.*, vol. 27, 2014, pp. 1269–1277.

[36] J. Jin, A. Dundar, and E. Culurciello, "Flattened convolutional neural networks for feedforward acceleration," 2014, *arXiv:1412.5474*.

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

[38] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet large scale visual recognition challenge," *Int. J. Comput. Vis.*, vol. 115, no. 3, pp. 211–252, Dec. 2015.

[39] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," Univ. Toronto, Toronto, ON, Canada, Tech. Rep. TR-2009, 2009.

[40] A. Khosla, N. Jayadevaprakash, B. Yao, and F. Li, "Novel dataset for fine-grained image categorization: Stanford dogs," in *Proc. CVPR Workshop Fine-Grained Vis. Categorization (FGVC)*, vol. 2, 2011, pp. 806–813.

[41] M.-E. Nilsback and A. Zisserman, "Automated flower classification over a large number of classes," in *Proc. 6th Indian Conf. Comput. Vis., Graph. Image Process.*, Dec. 2008, pp. 722–729.

[42] Y.-J. Huang and H.-L. Wu, "New coarse-to-fine approaches for age estimation based on separable convolutions," *IEEE Access*, vol. 11, pp. 130306–130318, 2023.

YAN-JEN HUANG received the M.S. degree in computer science and information engineering from National Taipei University, Taiwan, in 2018, where he is currently pursuing the Ph.D. degree in electrical engineering and computer science. His research interests include algorithm design, machine learning, and deep learning.

HSIN-LUNG WU received the Ph.D. degree in computer science and information engineering from National Chiao Tung University, Taiwan, in 2008. He is currently a Professor with the Department of Computer Science and Information Engineering, National Taipei University, Taiwan. His main research interests include the design and analysis of algorithms, computational complexity, theory of machine learning, and deep learning.

CHING-CHEN received the bachelor's degree in computer science and information engineering from National Taipei University, Taiwan, in 2022, where she is currently pursuing the master's degree in computer science and information engineering. Her research interests include machine learning and deep learning.