DualConv: Dual Convolutional Kernels for Lightweight Deep Neural Networks

Jiachen Zhong, Junying Chen, *Member, IEEE*, and Ajmal Mian, *Senior Member, IEEE*

***Abstract*— Convolutional neural network (CNN) architectures are generally heavy on memory and computational requirements which make them infeasible for embedded systems with limited hardware resources. We propose dual convolutional kernels (DualConv) for constructing lightweight deep neural networks. DualConv combines 3 × 3 and 1 × 1 convolutional kernels to process the same input feature map channels simultaneously and exploits the group convolution technique to efficiently arrange convolutional filters. DualConv can be employed in any CNN model such as VGG-16 and ResNet-50 for image classification, you only look once (YOLO) and R-CNN for object detection, or fully convolutional network (FCN) for semantic segmentation. In this work, we extensively test DualConv for classification since these network architectures form the backbone for many other tasks. We also test DualConv for image detection on YOLO-V3. Experimental results show that, combined with our structural innovations, DualConv significantly reduces the computational cost and number of parameters of deep neural networks while surprisingly achieving slightly higher accuracy than the original models in some cases. We use DualConv to further reduce the number of parameters of the lightweight MobileNetV2 by 54% with only 0.68% drop in accuracy on CIFAR-100 dataset. When the number of parameters is not an issue, DualConv increases the accuracy of MobileNetV1 by 4.11% on the same dataset. Furthermore, DualConv significantly improves the YOLO-V3 object detection speed and improves its accuracy by 4.4% on PASCAL visual object classes (VOC) dataset.**

***Index Terms*— Dual convolution, lightweight deep neural network, parameter reduction, performance improvement.**

I. Introduction

Convolutional neural networks (CNNs) have achieved unmatched performance in many applications such as image classification, object detection, and semantic segmentation. Current research trend of improving and enhancing network performance makes the networks deeper and more complex, which eventually leads to a dramatic increase in the model size (number of parameters/weights) and the required computational resources. Due to these two reasons, modern CNN models can only run on servers equipped with high-performance GPUs. Although embedded devices and mobile platforms have a huge demand for deployment of deep models, current architectures are not suitable for these systems due to their limited memory, power, and computational resources. Therefore, designing lightweight yet accurate CNN models that can be deployed in embedded devices and mobile platforms has become an active research direction.

Manuscript received June 8, 2021; revised November 11, 2021 and February 3, 2022; accepted February 9, 2022. This work was supported in part by the National Natural Science Foundation of China under Grant 61802130 and in part by the Guangdong Natural Science Foundation under Grant 2019A1515012152 and Grant 2021A1515012651. Ajmal Mian was supported by the Australian Research Council Future Fellowship funded by the Australian Government under Project FT210100268. *(Corresponding author: Junying Chen.)*

Jiachen Zhong and Junying Chen are with the School of Software Engineering, South China University of Technology, Guangzhou 510006, China, and also with the Key Laboratory of Big Data and Intelligent Robot (SCUT), Ministry of Education, Guangzhou 510006, China (e-mail: 1173992770@qq.com; jychense@scut.edu.cn).

Ajmal Mian is with the Department of Computer Science, The University of Western Australia, Crawley, WA 6009, Australia (e-mail: ajmal.mian@uwa.edu.au).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TNNLS.2022.3151138.

Digital Object Identifier 10.1109/TNNLS.2022.3151138

In embedded devices and mobile platforms, the network accuracy, computational complexity, and number of parameters are all equally important factors for evaluating different network architectures. Hence, many methods have been proposed to increase the efficiency of neural network models. A general approach taken by these methods is to start from a standard CNN model and increase the model efficiency by reducing the number of parameters and floating-point operations (FLOPs) through model compression. Model compression can be divided into three broad categories: connection pruning [1], filter pruning [2]–[8], and model quantization [1], [9].

Network connection pruning not only reduces the network complexity but also prevents the network from overfitting. Hanson and Pratt [10] proposed a pruning method based on bias parameter attenuation. LeCun *et al.* [11] showed that a tradeoff can be made between network accuracy and complexity by using second-derivative information to remove unimportant weights from the network. Hassibi and Stork [12] further extended the idea and argued that retraining a highly pruned network (as in [11]) may lead to inferior generalization. However, both the works by LeCun *et al.* [11] and Hassibi and Stork [12] are based on the computation of the Hessian matrix which incurs a high computational cost. The idea of filter pruning is to prune the filter channels which contribute the least in the network model [4], [5]. Lin *et al.* [3] modeled network channelwise pruning as a Markov decision process and used reinforcement learning for training. They named this method as runtime neural pruning (RNP) as it pruned the deep neural network dynamically at runtime. Luo *et al.* [4] and Li *et al.* [7] focused on filter level pruning which pruned the whole filter if it was less important. He *et al.* [5] used the least absolute shrinkage and selection operator (LASSO) regression to select filter channels and least square reconstruction to rebuild the network. After pruning the network, the model usually needs fine-tuning to maintain its performance [2]. In model quantization, the idea is to reduce the parameters of the network or to reduce the storage bits of the feature maps [1], [9]. For example, Vanhoucke *et al.* [13] used 8-bit unsigned char to reduce the storage bits of the activation values. The reasoning behind this is that the accuracy/precision of the weights in the process of network inference does not need to be so high.

In practice, model compression is a costly and difficult process. Therefore, there is a need to design efficient networks right from the start. One such method is to design efficient network architectures that inherently have fewer parameters and lower complexity. An efficiently designed network architecture requires less data and time to train and is also easy to prune after training. However, designing a new network architecture that maintains high accuracy with minimal computational cost requires significant effort given the large number and space of hyperparameters. Moreover, this approach also does not take advantage of the many existing standard network architectures. Hence, a better approach is to design efficient convolutional filters for existing standard network architectures. The new convolutional filters can simply replace standard convolutional filters in existing CNN model architectures to reduce their computational complexity without sacrificing accuracy.

SqueezeNet proposed by Iandola *et al.* [14] significantly reduces parameters and computational complexity while maintaining network accuracy. Its network structure unit introduces 1 × 1 convolution

to reduce the computational complexity. The 1×1 convolutional kernels not only reduce the network parameters and computational complexity but also offer the flexibility to control the depth of the feature maps, achieve cross-channel information fusion, and provide an additional level of nonlinearity. Depthwise separable convolution was proposed in MobileNetV1 by Howard *et al.* [15]. The depthwise separable convolution is a form of factorized convolution that factorizes a standard convolution into a depthwise and a 1×1 pointwise convolution. When the 1×1 convolution is used to filter the input feature maps, it fuses the original information of each channel of input feature maps into the output feature maps. Hence, the original information of input images can pass to deeper convolutional layers. MobileNetV2 [16] proposed inverted residual blocks which first use 1×1 convolutions to increase the number of channels of the input feature maps and then use depthwise separable convolutions to filter the features. ShuffleNet proposed by Zhang *et al.* [17] uses a form of convolution called group convolution (GroupConv) to reduce the computational cost of the network. It also uses the channel shuffle operation to enhance the interaction between different groups of channels. Heterogeneous convolution (HetConv) [18] uses heterogeneous convolutional kernels with different sizes within a convolutional filter. Whereas the 3×3 convolution in HetConv extracts the spatial information of input feature map, the 1×1 convolution in HetConv reduces the computational cost of neural network allowing for information sharing between convolutional layers. Heterogeneous convolutional filters are applicable to existing standard network architectures to reduce the network complexity.

Inspired by GroupConv and HetConv, we propose dual convolution (DualConv), designing a new convolutional filter which integrates 3×3 group convolution with 1×1 pointwise convolution to deal with the same input feature map channels simultaneously. Because DualConv uses 1×1 convolution to preserve the original information of input feature maps, the 3×3 convolutional filters at deeper convolutional layers can still learn from the original information of input feature maps. DualConv is more efficient and more general compared to model compression methods, because it can be applied to all current and future CNN architectures.

The proposed DualConv is used to replace the standard convolution in VGG-16 [19] and ResNet-50 [20] to perform image classification experiments on CIFAR-10 [21], CIFAR-100 [21], and the large-scale ImageNet [22] datasets. Our results show that the proposed DualConv significantly reduces the cost of network computation and the number of parameters while surprisingly achieving slightly higher accuracy than the original models in some cases. For better comparison, we reproduce GroupConv and HetConv to replace the standard convolution in VGG-16 on CIFAR-10 dataset, and replace the standard convolution in ResNet-50 on ImageNet dataset. Quantitative results and visual analysis show that DualConv generally achieves higher accuracy than GroupConv and HetConv with slightly higher number of parameters. DualConv is further applied to modify the convolutional filters in MobileNetV1. Our DualConv-modified MobileNetV1 performs more accurately than the original MobileNetV1 as well as the GroupConv-modified (or HetConv-modified) MobileNetV1. DualConv is also applied to modify the convolutional filters in MobileNetV2, reducing its parameters by 54% with only 0.68% drop in accuracy on CIFAR-100 dataset.

The proposed DualConv is further tested on object detection task by replacing the 3×3 standard convolution in the you only look once model (YOLO-V3) [23]. Experiments on PASCAL visual object classes (VOC) dataset [24] show that DualConv-modified YOLO-V3 requires much less computations leading to faster detection speed. Moreover, DualConv-modified YOLO-V3 improves the mean average precision (mAP) value of each image with 4.4% higher accuracy.

II. Related Work

Efficient convolutional filters can effectively reduce the computational cost and parameters of a neural network, and eliminate the need for designing new convolutional network architectures from scratch. Three types of efficient convolutional filters have been proposed in the literature to replace standard convolutional filters in existing network architectures. We briefly describe these below.

A. Depthwise Separable Convolution

A standard convolution, shown in Fig. 1(a), simultaneously performs feature extraction and channel fusion on the input feature maps. Depthwise separable convolution in MobileNetV1 [15] decomposes the standard convolution into depthwise convolution and pointwise convolution as shown in Fig. 1(b). In depthwise convolution, a single convolutional kernel is applied to each input channel. Usually, a 3×3 convolution is used for such feature extraction. Pointwise convolution applies 1×1 convolution to the output feature map of depthwise convolution to perform channelwise fusion. Hence, by splitting the feature extraction and channel fusion, the depthwise separable convolution significantly reduces the number of parameters and consequently the computations performed by the network.

B. Group Convolution

The concept of GroupConv was first proposed in AlexNet [25]. Due to limited GPU performance, at that time, the model was divided into two GPUs for training. In GroupConv, the convolutional filters are divided into G groups and the input feature map channels are also divided into G groups as shown in Fig. 1(c). Each group of convolutional filters processes the corresponding group of input feature map channels. Since each group of convolutional filters is only applied to the corresponding input channel group, the computational cost of convolution is significantly reduced, but the channel information is not shared between different groups, i.e., different groups of output feature map channels only receive information from their corresponding groups of input channels. This hinders the flow of information between different groups of channels, reducing the feature extraction ability of GroupConv. To overcome this issue, ShuffleNet [17] performs a channel shuffle operation to enhance the information exchange between different groups of channels.

C. Heterogeneous Convolution

HetConv [18] contains both 3×3 convolution and 1×1 convolution in one convolutional filter, as shown in Fig. 1(d). Note that the heterogeneous filters are arranged in a shifted manner (see Fig. 3 in [18]). The M/P 3×3 convolutional kernels are discretely arranged, and the 3×3 and 1×1 kernels alternate within a convolutional filter. The computational complexity of the original 3×3 standard convolution can be reduced by three to eight times using heterogeneous convolutional filters, without sacrificing the accuracy of the network much. The heterogeneous design essentially breaks down the continuity of cross-channel information integration and negatively affects the preservation of complete information of input feature map. Therefore, such a strategy could reduce the network accuracy.

III. Proposed Dual Convolution

A. Design Scheme of Dual Convolution

We propose dual convolution which combines the strengths of group convolution and heterogeneous convolution. Whereas some convolutional kernels perform both 3×3 and 1×1 convolutional operations simultaneously, others only perform 1×1 convolutions,

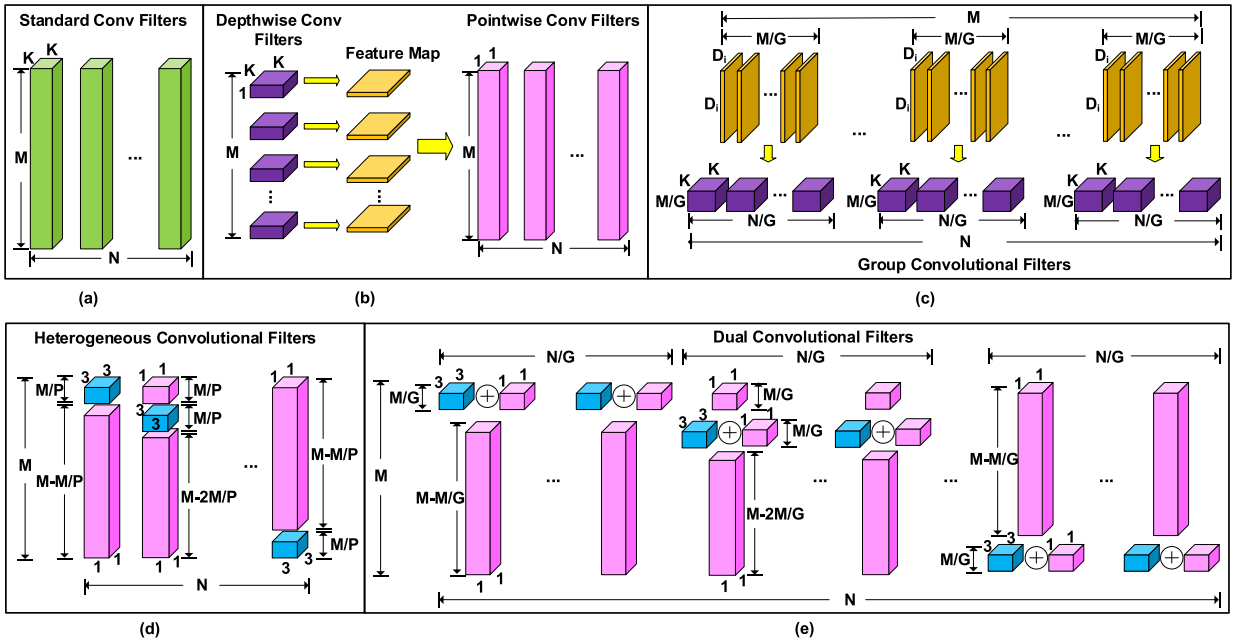

Fig. 1. Convolutional filter designs of (a) standard convolution, (b) depthwise separable convolution, (c) group convolution, (d) heterogeneous convolution, and (e) proposed dual convolution. M is the number of input channels (i.e., the depth of input feature map), N is the number of convolutional filters and also the number of output channels (i.e., the depth of output feature map), D_i is the width and height dimension of input feature map, $K \times K$ is the convolutional kernel size, G is the number of groups in group convolution and dual convolution, and $1/P$ is the ratio of 3×3 convolutional kernels in heterogeneous convolution. Note that the heterogeneous filters are arranged in a shifted manner [18].

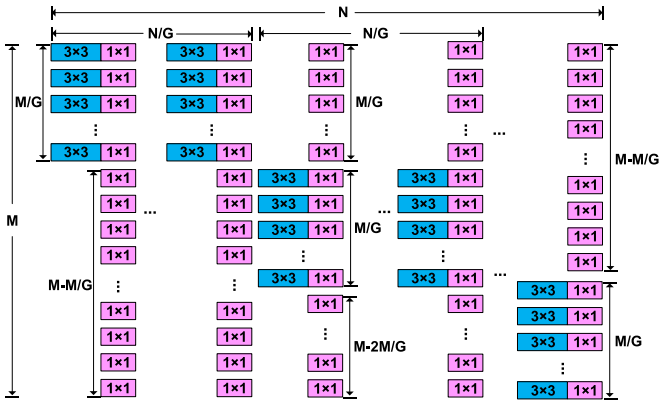

Fig. 2. Structural layout of dual convolution.

as shown in Fig. 1(e). The structural layout of dual convolution is shown in Fig. 2. Note how the 3×3 convolution moves in the feature map channel dimension and yet the 1×1 convolution is performed on all input channels. Our approach can be regarded as the combination of 3×3 group convolution and 1×1 pointwise convolution on the same input feature map, which makes it easy to integrate into existing network architectures. Because applying continuous 1×1 convolution on input feature maps can preserve the original information, it can help deeper convolutional layers to extract information more effectively.

DualConv not only solves the problem of poor communication of GroupConv but also improves the performance of deep neural networks compared to HetConv. In the GroupConv shown in Fig. 1(c), every N/G convolutional filters handle M/G input feature map channels, extracting information for N/G output feature map channels. As each convolutional filter extracts information from only $1/G$ input channels, the output feature map channel of such convolutional filter contains less information than that of the convolutional filter which handles the complete input feature map. Based on this observation, we add M 1×1 convolutional kernels to each convolutional filter so that it is able to handle the complete input feature map for better information extraction and sharing between convolutional layers. In the HetConv shown in Fig. 1(d), M/P kernels are 3×3 convolutional kernels, and the rest $(M - M/P)$ kernels are 1×1 convolutional kernels. Such alternative arrangement breaks down the continuity of cross-channel information integration and negatively affects the preservation of complete information of input feature map. Based on this observation, we design parallel 1×1 convolutional kernels for all the 3×3 convolutional kernels, so as to preserve the original information of input feature maps to help deeper convolutional layers to extract more effective features.

We combine the above two modifications together to design DualConv. We divide N convolutional filters into G groups, each group handles the complete input feature map where M/G input feature map channels are processed by 3×3 and 1×1 convolutional kernels simultaneously and the rest $(M - M/G)$ input channels are processed by 1×1 convolutional kernels solely. The results of simultaneous 3×3 and 1×1 convolutional kernels are summed up, as indicated by the $\oplus$ sign in Fig. 1(e). Because the filter group structure enforces a block-diagonal sparsity on the channel dimension, the filters with high correlation are learned in a more structured way [26]. As such, we do not arrange the convolutional filters in a shifted manner. The design of DualConv reduces the parameters of original backbone network models through group convolution strategy and promotes better information sharing between convolutional layers by preserving the original information of input feature maps and allowing for maximum cross-channel communication with M 1×1 convolutions. As a result, DualConv can be constructed without the need for channel shuffle operation.

Assume that the size of output feature map is $D_o \times D_o \times N$, where D_o is the width and height dimension of output feature map. In the standard convolution shown in Fig. 1(a), the input feature map is filtered by N convolutional filters with size of $K \times K \times M$ in

the convolutional layer, where $K \times K$ is the convolutional kernel size. Therefore, the total number of FLOPs performed in a standard convolutional layer $\mathrm{FL_{SC}}$ is

$$\mathrm{FL_{SC}} = D_o^2 \times K^2 \times M \times N. \tag{1}$$

In DualConv, the number of convolutional filter groups G is used to control the proportion of $K \times K$ convolutional kernels in a convolutional filter. For a given G, the proportion of combined simultaneous convolutional kernels with size of $(K \times K + 1 \times 1)$ is $1/G$ of all channels, while the proportion of the remaining 1×1 convolutional kernels is $(1 - 1/G)$. Therefore, in a dual convolutional layer composed of G convolutional filter groups, the number of FLOPs for the combined convolutional kernels is

$$\mathrm{FL_{CC}} = \left(D_o^2 \times K^2 \times M \times N + D_o^2 \times M \times N\right)/G \tag{2}$$

and the number of FLOPs for the remaining 1×1 pointwise convolutional kernels is

$$\mathrm{FL_{PC}} = \left(D_o^2 \times M \times N\right) \times (1 - 1/G). \tag{3}$$

The total number of FLOPs is

$$\begin{aligned} \mathrm{FL_{DC}} &= \mathrm{FL_{CC}} + \mathrm{FL_{PC}} \\ &= D_o^2 \times K^2 \times M \times N/G + D_o^2 \times M \times N. \end{aligned} \tag{4}$$

Comparing the computational cost (FLOPs) of dual convolutional layer with that of standard convolutional layer, the computational reduction ratio $R_{\mathrm{DC/SC}}$ is

$$R_{\mathrm{DC/SC}} = \frac{\mathrm{FL_{DC}}}{\mathrm{FL_{SC}}} = \frac{1}{G} + \frac{1}{K^2}. \tag{5}$$

As seen from (5), given that $K = 3$ in DualConv design, the speedup can reach eight to nine times when G is large.

B. Comparison With Previous Work

As shown in Fig. 1(b), a depthwise separable convolutional layer contains two convolutional layers, i.e., a depthwise convolutional layer followed by a pointwise convolutional layer, which increases the network complexity. On the contrary, the proposed DualConv does not add additional layers to the network. The number of FLOPs for a depthwise separable convolutional layer is

$$\mathrm{FL_{DSC}} = D_o^2 \times \left(K^2 \times M + M \times N\right). \tag{6}$$

The computational reduction ratio over the standard convolutional layer $R_{\mathrm{DSC/SC}}$ is

$$R_{\mathrm{DSC/SC}} = \frac{\mathrm{FL_{DSC}}}{\mathrm{FL_{SC}}} = \frac{1}{N} + \frac{1}{K^2}. \tag{7}$$

As mentioned in Section III-A, each convolutional filter in GroupConv extracts information from only $1/G$ input channels, while the convolutional filter in DualConv handles the complete input feature map. In a group convolutional layer, the number of FLOPs is

$$\mathrm{FL_{GC}} = \left(D_o^2 \times K^2 \times M \times N\right)/G \tag{8}$$

and the computational reduction ratio $R_{\mathrm{GC/SC}}$ is

$$R_{\mathrm{GC/SC}} = \frac{\mathrm{FL_{GC}}}{\mathrm{FL_{SC}}} = \frac{1}{G}. \tag{9}$$

Unlike HetConv where the 1×1 convolution is not applied to all input feature map channels, the proposed DualConv operates 1×1 convolution on the whole input feature map. It can retain and fuse the information of the original input features better than HetConv, with only a slight increase in the number of FLOPs and parameters. The number of FLOPs for a heterogeneous convolutional layer is

$$\mathrm{FL_{HC}} = \left(D_o^2 \times M \times N\right) \times \frac{K^2 + P - 1}{P} \tag{10}$$

and the computational reduction ratio $R_{\mathrm{HC/SC}}$ is

$$R_{\mathrm{HC/SC}} = \frac{\mathrm{FL_{HC}}}{\mathrm{FL_{SC}}} = \frac{1}{P} + \frac{1}{K^2} - \frac{1}{P \times K^2}. \tag{11}$$

As derived from (7) and (11), given $K = 3$, when N and P are large, the speedup of depthwise separable convolution and HetConv can reach eight to nine times, which is similar to the speedup of DualConv. However, the speedup of GroupConv is proportional to G.

IV. Experiments and Discussions

We perform extensive experiments using the proposed dual convolutional filters. The tradeoff between accuracy and computational cost of the network model is adjusted by the number of convolutional filter groups, G. When the value of G is large, the structure of DualConv becomes closer to the standard convolution consisting of all 1×1 convolutional kernels. In general, DualConv retains the accuracy of the original network and in some cases achieves slightly higher accuracy than the original convolutional filter. Moreover, compared to other efficient convolutions with similar computational cost, DualConv achieves higher network accuracy. Hence, DualConv makes it more feasible to deploy deep CNNs on mobile platforms or embedded devices.

A. VGG-16 and ResNet-50 on CIFAR-10

For VGG-16 network architecture, we replace the 3×3 standard convolutions in the last 12 layers with the proposed DualConv. The G values for all replaced layers are the same, and the number of convolutional kernels in each layer is kept the same as that of the original VGG-16 network. In ResNet-50 network structure, we use DualConv to replace all the 3×3 standard convolutions with stride 1 in the convolutional layers (except the first layer). For hyperparameters, we set the weight decay to 5e-4, the initial learning rate to 0.1 and multiply it by 0.1 after every 50 epochs. We use the stochastic gradient descent (SGD) optimizer and the multiply step learning rate decay strategy.

Table I shows the performance comparisons of DualConv, GroupConv, and HetConv on CIFAR-10 dataset using VGG-16 network architecture. Table I also illustrates the comparisons with several representative model compression methods applied to VGG-16, e.g., Li-pruned [7], structured Bayesian pruning (SBP) [8], and auto-balanced filter pruning (AFP) [6]. For a fair comparison between different convolutional filters under the same implementation framework, the proposed DualConv and the reproduced GroupConv and HetConv are all implemented in the PyTorch framework adopting the im2col method to flatten the feature maps and convolutional kernels. The implementation framework and the flattening method may be the reasons why the reproduced HetConv performs slightly (0.2%–0.76%) worse than the HetConv reported in [18].

In Table I, with the increase of G value, the number of FLOPs and parameters of network decrease significantly, whereas the network accuracy drops slightly. When $G = 4$, the accuracy of VGG-16 network is actually higher than that of standard VGG-16 network while the computations and the parameters are both reduced by over 60%. DualConv generally obtains higher accuracy than HetConv, demonstrating better feature learning ability than HetConv, since 1×1 convolution is applied to all channels. Moreover, as for ResNet-50, the accuracy of ResNet-50 with DualConv outperforms the standard ResNet-50 network and the computations and parameters are reduced significantly by over 25% when $G = 8$.

TABLE I

PERFORMANCE OF VGG-16 AND RESNET-50 WITH DUALCONV, GROUPCONV, OR HETCONV ON CIFAR-10 USING DIFFERENT SETTINGS[1]

Model	Acc (%)	FLOPs	Params
VGG-16	93.95	313.21M	14.73M
Li-pruned [7]	93.40	206.44M	5.32M
SBP [8]	92.50	136.41M	-
SBPa [8]	91.00	99.30M	-
AFP-E [6]	92.94	63.72M	-
AFP-F [6]	92.87	58.39M	-
VGG-16_GC_G2	92.48	157.49M	7.37M
VGG-16_GC_G4	90.23	79.64M	3.69M
VGG-16_GC_G8	88.46	40.71M	1.85M
VGG-16_GC_G16	85.78	21.24M	0.93M
VGG-16_GC_G32	82.62	11.51M	0.48M
VGG-16_HC_P2	93.69	175.23M	8.45M
VGG-16_HC_P4	93.67	105.98M	5.17M
VGG-16_HC_P8	93.52	71.35M	3.54M
VGG-16_HC_P16	93.34	54.04M	2.72M
VGG-16_HC_P32	92.97	45.38M	2.31M
VGG-16_G2	93.91	192.10M	9.00M
VGG-16_G4	**94.14**	114.24M	5.33M
VGG-16_G8	93.61	75.31M	3.49M
VGG-16_G16	93.55	55.85M	2.57M
VGG-16_G32	93.20	46.11M	2.11M
ResNet-50	94.08	1.30G	23.52M
ResNet-50_G2	93.81	1.11G	20.32M
ResNet-50_G4	94.15	984.33M	18.27M
ResNet-50_G8	**94.33**	922.99M	17.24M
ResNet-50_G16	93.68	892.32M	16.73M
ResNet-50_G32	93.08	876.98M	16.47M

[1]VGG-16(ResNet-50)_Gα represents DualConv-modified VGG-16 or ResNet-50, VGG-16_GC_Gα represents GroupConv-modified VGG-16, and VGG-16_HC_Pα represents HetConv-modified VGG-16, where α is the value of G in DualConv and GroupConv, or the value of P in HetConv.

Our results demonstrate that 1×1 pointwise convolution can transfer and fuse the information of input feature maps well, and the output feature maps can retain the information of input feature maps better. Hence, it can be performed without the need for channel shuffle operation.

B. MobileNetV1 and MobileNetV2 on CIFAR-10

In MobileNetV1 network architecture, we replace all the depthwise separable convolutions with our proposed dual convolutions. Since the images in CIFAR-10 and CIFAR-100 datasets are much smaller than the images in ImageNet dataset, the stride of the first depthwise separable convolutional layer in the original or DualConv-modified MobileNetV1 network is modified to 1 instead of 2. In MobileNetV2 network structure, we replace the inverted residual block with our proposed dual convolution while the convolution stride is kept as 1. The replacement strategy is to add batch normalization and ReLU6 operations after the proposed dual convolution when replacing the inverted residual block. The experimental settings are the same as those in Section IV-A.

Table II shows our experimental results. In MobileNetV1 network structure, although the proposed DualConv increases the parameters and computational cost of the original MobileNetV1 network, it improves the network accuracy by 1.23%. Even when $G = 32$, MobileNetV1 with DualConv still has higher accuracy than the original MobileNetV1 network with similar parameters and computational cost. In MobileNetV2 network structure, our proposed DualConv can decrease the network parameters and computational cost exceeding 60% with only 1.16% drop in accuracy when $G = 32$.

TABLE II

PERFORMANCE OF MOBILENETV1 AND MOBILENETV2 WITH DUALCONV ON CIFAR-10 USING DIFFERENT SETTINGS

Model	Acc (%)	FLOPs	Params
MobileNetV1	91.91	46.37M	3.22M
MobileNetV1_HC_P32 [18]	92.17	56.91M	-
MobileNetV1_G2	93.09	243.12M	17.29M
MobileNetV1_G4	**93.14**	144.03M	10.23M
MobileNetV1_G8	92.92	94.49M	6.69M
MobileNetV1_G16	92.64	69.71M	4.93M
MobileNetV1_G32	91.92	57.3M	4.04M
MobileNetV2	91.99	64.96M	2.37M
MobileNetV2_G2	90.71	41.84M	1.45M
MobileNetV2_G4	90.56	31.07M	1.09M
MobileNetV2_G8	90.56	25.68M	916.92K
MobileNetV2_G16	90.31	23.50M	829.94K
MobileNetV2_G32	**90.83**	22.42M	786.45K

TABLE III

PERFORMANCE OF VGG-16 AND RESNET-50 WITH DUALCONV ON CIFAR-100 USING DIFFERENT SETTINGS

Model	Acc (%)	FLOPs	Params
VGG-16	72.41	332.48M	34.01M
VGG-16_G2	**73.04**	211.37M	28.29M
VGG-16_G4	72.82	133.52M	24.61M
VGG-16_G8	72.52	94.59M	22.78M
VGG-16_G16	72.07	75.12M	21.86M
VGG-16_G32	71.52	65.39M	21.40M
ResNet-50	78.55	1.30G	23.70M
ResNet-50_G2	78.46	1.11G	20.50M
ResNet-50_G4	**78.53**	984.51M	18.45M
ResNet-50_G8	78.01	923.17M	17.43M
ResNet-50_G16	77.77	892.50M	16.91M
ResNet-50_G32	77.50	877.17M	16.65M

C. VGG-16 and ResNet-50 on CIFAR-100

In this experiment, we use DualConv to modify VGG-16 and ResNet-50 network structures to perform image classification on a larger dataset CIFAR-100. We replace the 3×3 standard convolutions with dual convolutions. Note that, the VGG-16 architecture used for CIFAR-100 dataset has three fully connected layers, while the VGG-16 architecture used for CIFAR-10 dataset has only one fully connected layer. The values of the hyperparameters are set as: weight decay = 5e-4, and initial learning rate = 0.1 which is multiplied by 0.2 after every 60 epochs. Moreover, we use SGD optimizer and multiply step learning rate decay strategy.

The Top-1 accuracy of the networks are recorded in Table III. As shown in Table III, when we replace the 3×3 standard convolutions of VGG-16 with the proposed dual convolutions, the accuracy improves when G increases to 8. The computational cost is reduced by more than 70% when $G = 8$. When G increases further, the accuracy of network drops slightly, but the number of parameters and computational cost are further reduced. On the other hand, the best accuracy of ResNet-50 with DualConv is achieved when $G = 4$, which is only 0.02% lower than the original ResNet-50, but the number of parameters and computational cost are reduced by more than 20%. These results demonstrate the strong generalization ability of the proposed DualConv.

D. MobileNetV1 and MobileNetV2 on CIFAR-100

In this experiment, we test the proposed DualConv in MobileNetV1 and MobileNetV2 network structures to perform image classification

TABLE IV
PERFORMANCE OF MOBILENETV1 AND MOBILENETV2 WITH DUALCONV ON CIFAR-100 USING DIFFERENT SETTINGS

Model	Acc (%)	FLOPs	Params
MobileNetV1	68.13	46.46M	3.32M
MobileNetV1_G2	72.05	243.21M	17.38M
MobileNetV1_G4	**72.24**	144.12M	10.32M
MobileNetV1_G8	71.79	94.58M	6.79M
MobileNetV1_G16	71.01	69.81M	5.02M
MobileNetV1_G32	70.28	57.42M	4.14M
MobileNetV2	68.54	64.96M	2.37M
MobileNetV2_G2	**67.86**	41.84M	1.45M
MobileNetV2_G4	**67.86**	31.07M	1.09M
MobileNetV2_G8	67.64	25.68M	916.92K
MobileNetV2_G16	67.19	23.50M	829.94K
MobileNetV2_G32	67.22	22.42M	786.45K

on CIFAR-100 dataset. The experimental settings are the same as those in Section IV-C.

As shown in Table IV, when we replace the depthwise separable convolutions of MobileNetV1 network with the proposed dual convolutions, the accuracy increases by 4.11% when $G = 4$. Even when the G value increases to 32, the network accuracy still outperforms the standard MobileNetV1 architecture and has similar computational cost. In MobileNetV2 network structure, the proposed dual convolution greatly reduces the network parameters and computational cost with only a slight drop in accuracy (0.68% when $G = 2$ or $G = 4$). These experimental results also confirm the strong generalization ability of the proposed DualConv.

E. VGG-16 and ResNet-50 on ImageNet

We experiment with the DualConv-modified VGG-16 and ResNet-50 network architectures on the large-scale ImageNet dataset. The modification strategy of VGG-16 is the same as that described in Section IV-A, and VGG-16 for ImageNet also has three fully connected layers as that for CIFAR-100. For ResNet-50 network, we use DualConv to replace all the 3×3 standard convolutions with stride 1 and 2 in the convolutional layers (except the first layer). Moreover, the values of the hyperparamters are set as: weight decay = 1e-4, batch size = 128, initial learning rate = 0.01 in VGG-16 and 0.1 in ResNet-50, and the learning rate is multiplied by 0.1 after every 30 epochs. Both network architectures are trained for 90 epochs.

Table V presents the performance of VGG-16 and ResNet-50 networks with DualConv on ImageNet. Some representative model compression methods (including RNP (3X) [3], CP 2X [5], ThiNet [4], and neuron importance score propagation (NISP) [2]), HetConv-modified VGG-16 and ResNet-50 (achieving the best performance when $P = 4$), and GroupConv-modified ResNet-50 (achieving the best performance when $G = 2$) are also compared.

As illustrated in Table V, when $G = 2$, the computational cost of VGG-16 with DualConv decreases by about 38% compared to the original VGG-16 network with only a slight drop in accuracy (0.48% in Top-1 accuracy and 0.17% in Top-5 accuracy). Note that, the number of parameters of VGG-16 with DualConv does not change much on the ImageNet dataset. This is because the last fully connected (nonconvolutional) layers occupy most of the parameters, i.e., about 102 M on ImageNet dataset. In the case for ResNet-50, the model with DualConv significantly decreases the computational cost and parameters of the original ResNet-50 model with a slight drop in accuracy (0.18% in Top-1 accuracy and 0.29% in Top-5 accuracy) when $G = 2$. Furthermore, from Table V, we can see that our proposed DualConv achieves better accuracy than model compression methods and other efficient convolutional filters (i.e., GroupConv and HetConv) with similar parameters and computational cost.

TABLE V
PERFORMANCE OF VGG-16 AND RESNET-50 WITH DUALCONV ON IMAGENET USING DIFFERENT SETTINGS[1]

Model	Top-1 Acc(%)	Top-5 Acc(%)	FLOPs	Params	GTime (ms)	CTime (ms)
VGG-16	72.69	90.91	15.47G	138.36M	1.74	307.96
RNP (3X) [3]	-	87.57	5.16G	-	-	-
ThiNet-70 [4]	69.8	89.53	4.79G	131.44M	76.71*	-
CP 2X [5]	-	89.90	7.74G	-	-	-
VGG-16_HC_P4 [18]	71.2	90.20	5.29G	-	-	-
VGG-16_G2	**72.21**	**90.74**	9.54G	132.64M	1.70	239.28
VGG-16_G4	70.74	89.87	5.72G	128.96M	1.54	185.05
VGG-16_G8	69.74	89.18	3.81G	127.13M	1.50	161.86
VGG-16_G16	68.79	88.75	2.86G	126.21M	1.54	**148.61**
VGG-16_G32	68.22	88.06	2.38G	125.75M	1.52	161.78
ResNet-50	74.27	92.09	4.09G	25.56M	1.34	99.21
ThiNet-50 [4]	71.01	90.02	3.41G	12.38M	153.60*	-
NISP [2]	72.67	-	2.97G	14.36M	-	-
ResNet-50_GC_G2	73.85	91.77	3.16G	19.90M	1.22	80.07
ResNet-50_HC_P4	73.16	91.27	2.86G	18.02M	1.36	92.83
ResNet-50_G2	**74.09**	**91.80**	3.37G	21.16M	1.34	87.46
ResNet-50_G4	74.03	91.76	2.91G	18.33M	1.32	79.47
ResNet-50_G8	73.83	91.77	2.68G	16.91M	1.32	**75.19**
ResNet-50_G16	73.49	91.31	2.56G	16.20M	1.34	75.88
ResNet-50_G32	72.98	91.05	2.50G	15.85M	1.32	76.10

*Recorded on a single M40 GPU with batch size of 32.
[1]GTime represents GPU time, and CTime represents CPU time.

F. MobileNetV1 and MobileNetV2 on ImageNet

We also experiment with the DualConv-modified MobileNetV1 and MobileNetV2 network architectures on the large-scale ImageNet dataset. The stride of the first depthwise separable convolutional layer in the original or DualConv-modified MobileNetV1 network is changed back to 2 for ImageNet dataset. The hyperparameter settings in MobileNetV1 are the same as those in ResNet-50 on ImageNet (as described in Section IV-E). The values of the hyperparamters in MobileNetV2 are set as: weight decay = 4e-5 and initial learning rate = 0.05. Besides, Cosine learning rate decay strategy is used, and the network is trained for 150 epochs.

Table VI shows that although DualConv increases the accuracy of MobileNetV1 network (up to 1.64% in Top-1 accuracy and 1.11% in Top-5 accuracy), it also increases the network parameters and computational cost. As for MobileNetV2, the model parameters and computational cost are reduced by applying DualConv. However, the accuracy of MobileNetV2 with DualConv is lower than the original MobileNetV2. A possible reason for this is that we use DualConv to replace the entire inverted residual block in MobileNetV2 network, activating the feature maps once, while the inverted residual block activates the feature maps twice.

Our results show that DualConv can be integrated in both standard and lightweight network architectures to increase the network accuracy and reduce the network parameters and computational cost. Our experiments also demonstrate that the proposed DualConv can fit to various image classification datasets well and has a strong generalization capability.

G. Classification Time on ImageNet

As pointed out in [27], FLOPs is an indirect network efficiency metric but speed is a direct metric. Therefore, not only the computational cost (FLOPs) and number of parameters are important for the evaluation of efficient convolutional filters, but also the inference time

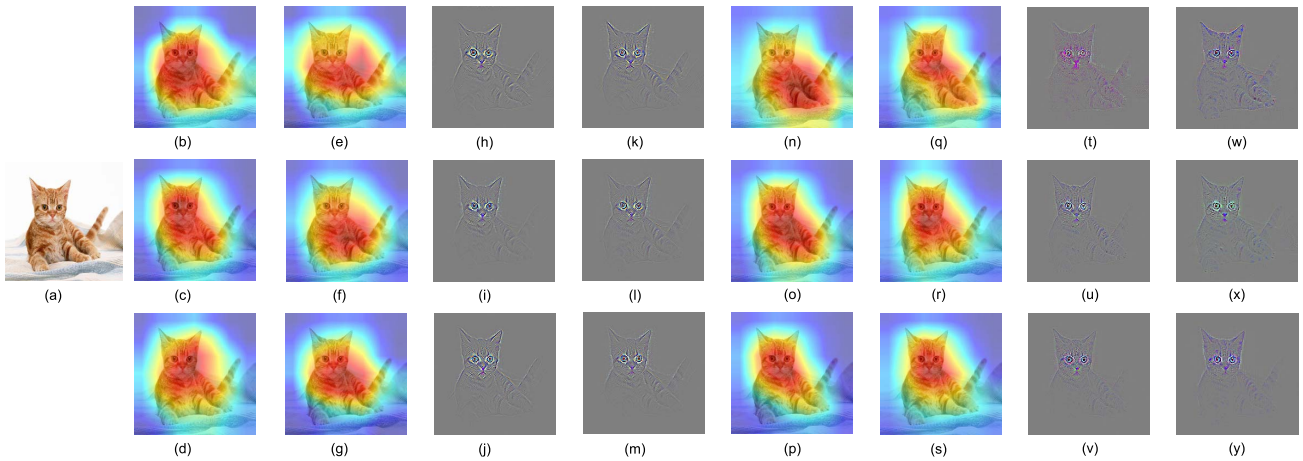

Fig. 3. Visualizations of an example image for ResNet-50 and MobileNetV1 networks on ImageNet dataset. (a) Original input image. (b)–(g) Heatmaps obtained by Grad-CAM method on ResNet-50 networks. (h)–(m) Guided Grad-CAM visualizations integrating guided backpropagation with Grad-CAM on ResNet-50 networks. (n)–(s) Heatmaps obtained by Grad-CAM method on MobileNetV1 networks. (t)–(y) Guided Grad-CAM visualizations integrating guided backpropagation with Grad-CAM on MobileNetV1 networks. (a) Input Image. (b) ResNet-50. (c) ResNet-50_G2. (d) ResNet-50_G4. (e) ResNet-50_G8. (f) ResNet-50_GC_G2. (g) ResNet-50_HC_P4. (h) ResNet-50. (i) ResNet-50_G2. (j) ResNet-50_G4. (k) ResNet-50_G8. (l) ResNet-50_GC_G2. (m) ResNet-50_HC_P4. (n) MobileNetV1. (o) MobileNetV1_G2. (p) MobileNetV1_G4. (q) MobileNetV1_G8. (r) MobileNetV1_GC_G2. (s) MobileNetV1_HC_P4. (t) MobileNetV1. (u) MobileNetV1_G2. (v) MobileNetV1_G4. (w) MobileNetV1_G8. (x) MobileNetV1_GC_G2. (y) MobileNetV1_HC_P4.

TABLE VI

PERFORMANCE OF MOBILENETV1 AND MOBILENETV2 WITH DUALCONV ON IMAGENET USING DIFFERENT SETTINGS

Model	Top-1 Acc(%)	Top-5 Acc(%)	FLOPs	Params	GTime (ms)	CTime (ms)
MobileNetV1	70.65	89.69	568M	4.23M	0.80	21.73
MobileNetV1_GC_G2	67.90	88.02	2.44G	15.17M	0.77	53.96
MobileNetV1_HC_P4	70.47	89.61	1.63G	10.46M	0.97	55.33
MobileNetV1_G2	**72.29**	**90.80**	2.98G	18.31M	0.93	70.76
MobileNetV1_G4	72.13	90.64	1.77G	11.24M	0.84	46.27
MobileNetV1_G8	71.15	89.91	1.16G	7.71M	0.84	34.76
MobileNetV1_G16	70.47	89.48	854.82M	5.94M	0.84	30.80
MobileNetV1_G32	68.58	88.25	703.09M	5.06M	0.82	**30.77**
MobileNetV2	69.22	88.93	300.79M	3.50M	0.80	20.62
MobileNetV2_G2	**65.45**	**84.40**	221.46M	2.67M	0.80	**16.52**
MobileNetV2_G4	62.78	84.39	171.79M	2.35M	0.79	16.57
MobileNetV2_G8	62.20	84.12	146.95M	2.19M	0.79	17.30
MobileNetV2_G16	61.60	83.67	135.55M	2.11M	0.80	17.40
MobileNetV2_G32	55.97	79.20	135.26M	2.07M	0.79	17.36

is an important evaluation metric. We measure the inference time per image for each network model. The GPU inference time is recorded on a single NVIDIA Tesla V100 GPU, and the CPU time is recorded using a single thread on an Intel Core i7-8700 CPU. Since the image classification models generally run fast on V100 GPU and there exists synchronization overhead between GPU threads, we cannot see an obvious difference in GPU inference time between different models. Hence, we discuss about the CPU inference time for the evaluated models. It is demonstrated in Table V that the proposed DualConv not only reduces the parameters and computational cost of VGG-16 and ResNet-50 but also reduces their CPU inference time by 52% and 24%, respectively. Moreover, Table VI shows that although DualConv increases the inference time of MobileNetV1, but it reduces the model parameters, computational cost and inference time of MobileNetV2.

Because GroupConv leads to higher memory access cost when the number of groups (G) is larger [27], we can see from the CPU inference time in Table V that the DualConv-modified VGG-16 and ResNet-50 networks achieve faster speeds when $G = 16$ and $G = 8$, but not when $G = 32$.

About the inference time comparisons with GroupConv and HetConv, we can see from Tables V and VI that when the values of G (or P) are the same, GroupConv-modified models run faster than DualConv-modified models but HetConv-modified models run slower than DualConv-modified models (e.g., HetConv-modified ResNet-50 runs for 92.83 ms while DualConv-modified ResNet-50 runs for 79.47 ms). The former phenomenon is consistent with the number of FLOPs and parameters, but the latter is inconsistent because the HetConv filters are arranged in a shifted manner [18], which decreases the inference speed. This reflects the efficiency of the proposed DualConv.

H. Visual Analysis

To better illustrate the benefit of DualConv, we apply gradient-weighted class activation mapping (Grad-CAM) [28] and guided backpropagation [29] methods to visualize the ResNet-50 and MobileNetV1 networks on ImageNet dataset to obtain high-resolution class-discriminative visualizations. The resulting heatmaps and guided Grad-CAM visualizations of an example image are shown in Fig. 3. From Fig. 3, we can see that when DualConv ($G = 2$) is applied to ResNet-50, the localization shown in Grad-CAM heatmap is more centered than other ResNet-50 networks, and the fine-grained details in its guided Grad-CAM visualization are clearer than other ResNet-50 networks except the original ResNet-50. Besides, DualConv-modified MobileNetV1 ($G = 2$) presents the best localization and clearest fine-grained details among the compared MobileNetV1 networks.

I. YOLO-V3 on PASCAL VOC

To show that the proposed DualConv can generalize to different tasks, we apply DualConv to object detection model. YOLO-V3 [23] is one of the common one-stage object detection frameworks using a single CNN to predict multiple bounding boxes and class probabilities. Since the first convolutional layer of YOLO-V3 network is important for the low-level information extraction, it is not modified. All the remaining convolutional layers are modified with DualConv. The 3×3 standard convolutions with stride 1 in YOLO-V3 are replaced by dual convolutions. However, we do not replace the 3×3 convolutions with stride 2 in YOLO-V3 because the 1×1 convolutions with stride 2 would harm the information preservation and channel fusion of input feature maps.

TABLE VII
PERFORMANCE OF YOLO-V3 WITH DUALCONV ON PASCAL VOC 2007 TEST SET

Model	mAP(%)	FLOPs	Parameters	Time(ms)
YOLO-V3	41.24	32.71G	61.63M	26.66
YOLO-V3_G2	44.04	22.79G	42.41M	**18.17**
YOLO-V3_G4	**45.64**	16.41G	30.05M	19.79
YOLO-V3_G8	42.40	13.22G	23.88M	23.22
YOLO-V3_G16	40.64	11.63G	20.79M	21.81
YOLO-V3_G32	41.08	10.83G	19.24M	23.63

The original YOLO-V3 and DualConv-modified YOLO-V3 models are all trained from scratch. All the network models are trained for 100 epochs. We resize the input images to 416×416 and use all 16551 images in the training and validation sets of PASCAL VOC $2007 + 2012$ for training, and use the test set of PASCAL VOC 2007 for computing the mAP. First, the average precision (AP) value of each class in PASCAL VOC 2007 is obtained by calculating the area under the precision-recall curve, and then these are averaged to get the mAP value. The inference time is recorded using one NVIDIA Tesla V100 GPU.

From Table VII, we can see that the YOLO-V3_G4 model which uses the proposed DualConv not only reduces the computational cost and the number of parameters by about 50% but also improves the accuracy (mAP) by 4.4% compared to the original YOLO-V3 model which uses 3×3 standard convolutions. The inference time also improves from 26.66 to 19.79 ms (6.87 ms faster). Therefore, DualConv not only compresses the model but also improves the inference speed making it possible for object detection to be deployed on mobile platforms or embedded devices.

Note that the results in Table VII are obtained for networks using exactly the same settings, i.e., training from scratch on the same training set with the same number of epochs. Hence, our comparison is fair. These results are not comparable to the YOLO-V3 model that is pretrained on the ImageNet dataset which is much bigger.

V. CONCLUSION

We propose DualConv that combines 3×3 group convolution with 1×1 pointwise convolution solving the problem of cross-channel communication and preservation of the information in the original input feature maps. Compared to HetConv, DualConv improves network performance by adding minimal parameters. DualConv is applied to common network structures to perform image classification and object detection. By comparing the experimental results of standard convolution and DualConv, the effectiveness and efficiency of the proposed DualConv is demonstrated. As seen from the experimental results, DualConv can be integrated in both standard and lightweight network architectures to increase the network accuracy and reduce the network parameters, computational cost, and inference time. We also demonstrate that DualConv can fit to various image datasets well and has a strong generalization capability. Future research work will focus on deployment on embedded devices to further prove the efficiency of DualConv in practical applications.

REFERENCES

[1] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding," in *Proc. Int. Conf. Learn. Represent.*, 2016, pp. 1–14.

[2] R. Yu *et al.*, "NISP: Pruning networks using neuron importance score propagation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 9194–9203.

[3] J. Lin, Y. Rao, J. Lu, and J. Zhou, "Runtime neural pruning," in *Proc. Annu. Conf. Neural Inf. Process. Syst.*, 2017, pp. 2178–2188.

[4] 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.

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

[6] X. Ding, G. Ding, J. Han, and S. Tang, "Auto-balanced filter pruning for efficient convolutional neural networks," in *Proc. AAAI Conf. Artif. Intell.*, 2018, pp. 6797–6804.

[7] H. Li, A. Kadav, I. Durdanovic, H. Samet, and H. P. Graf, "Pruning filters for efficient ConvNets," in *Proc. Int. Conf. Learn. Represent.*, 2017, pp. 1–13.

[8] K. Neklyudov, D. Molchanov, A. Ashukha, and D. Vetrov, "Structured Bayesian pruning via log-normal multiplicative noise," in *Proc. Annu. Conf. Neural Inf. Process. Syst.*, 2017, pp. 6778–6787.

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

[10] S. J. Hanson and L. Y. Pratt, "Comparing biases for minimal network construction with back-propagation," in *Proc. Annu. Conf. Neural Inf. Process. Syst.*, 1989, pp. 177–185.

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

[12] B. Hassibi and D. G. Stork, "Second order derivatives for network pruning: Optimal brain surgeon," in *Proc. Annu. Conf. Neural Inf. Process. Syst.*, 1993, pp. 164–171.

[13] V. Vanhoucke, A. Senior, and M. Z. Mao, "Improving the speed of neural networks on CPUs," in *Proc. Deep Learn. Unsupervised Feature Learn. Workshop*, 2011, pp. 1–8.

[14] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer, "SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5 MB model size," 2016, *arXiv:1602.07360*.

[15] A. G. Howard *et al.*, "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] X. Zhang, X. Zhou, M. Lin, and J. Sun, "ShuffleNet: An extremely efficient convolutional neural network for mobile devices," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 6848–6856.

[18] P. Singh, V. K. Verma, P. Rai, and V. P. Namboodiri, "HetConv: Heterogeneous kernel-based convolutions for deep CNNs," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 4835–4844.

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

[20] 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.

[21] A. Krizhevsky, "Learning multiple layers of features from tiny images," University of Toronto, Tech. Rep., 2009.

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

[23] J. Redmon and A. Farhadi, "YOLOv3: An incremental improvement," 2018, *arXiv:1804.02767*.

[24] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and W. Zisserman, "The PASCAL visual object classes (VOC) challenge," *Int. J. Comput. Vis.*, vol. 88, no. 2, pp. 303–338, Jun. 2010.

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

[26] Y. Ioannou, D. Robertson, R. Cipolla, and A. Criminisi, "Deep roots: Improving CNN efficiency with hierarchical filter groups," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 5977–5986.

[27] 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.*, 2018, pp. 116–131.

[28] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, "Grad-CAM: Visual explanations from deep networks via gradient-based localization," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 618–626.

[29] J. T. Springenberg, A. Dosovitskiy, T. Brox, and M. Riedmiller, "Striving for simplicity: The all convolutional net," in *Proc. Int. Conf. Learn. Represent. Workshop*, 2015, pp. 1–14.