View metadata, citation and similar papers at core.ac.uk

brought to you by CORE

provided by Concordia University Research Repository

# SdcNet: A Computation-Efficient CNN for Object Recognition

Yunlong Ma
Department of Electrical and Computer Engineering
Concordia University
Montreal, Canada
m_yunlon@encs.concordia.ca

Chunyan Wang
Department of Electrical and Computer Engineering
Concordia University
Montreal, Canada
chunyan@ece.concordia.ca

***Abstract*—Extracting features from a huge amount of data for object recognition is a challenging task. Convolution neural network can be used to meet the challenge, but it often requires a large amount of computation resources. In this paper, a computation-efficient convolutional module, named SdcBlock, is proposed and based on it, the convolution network SdcNet is introduced for object recognition tasks. In the proposed module, optimized successive depthwise convolutions, supported by appropriate data management, are applied in order to generate vectors containing higher density and more varieties of feature information. The hyperparameters can be easily adjusted to suit varieties of tasks under different computation restrictions without significantly jeopardizing the performance. The experiments have shown that SdcNet achieved an error rate of 5.60% in CIFAR-10 with only 55M Flops and also reduced further the error rate to 5.24% using a moderate volume of 103M Flops. The expected computation efficiency of the SdcNet has been confirmed.**

***Index Terms*—Convolution Neural Network, Object Recognition, Feature Extraction, Successive Depthwise Convolutions, Data Flow Control**

## I. Introduction

Object recognition is widely used in various applications such as autopilot [1] and security systems [2]. Extracting various features related to the objects from diverse backgrounds is a critical challenge. The normal procedure of object recognition contains three steps, pre-processing, feature extraction and classification.

The feature extraction can be done by applying filtering-based methods, such as Wavelet [3] and SIFT [4]. SVM [5] and Adaboost [6] are often used for classification. Such processing methods are usually computation-efficient, however, they have limitations in handling a huge number of variations in object features.

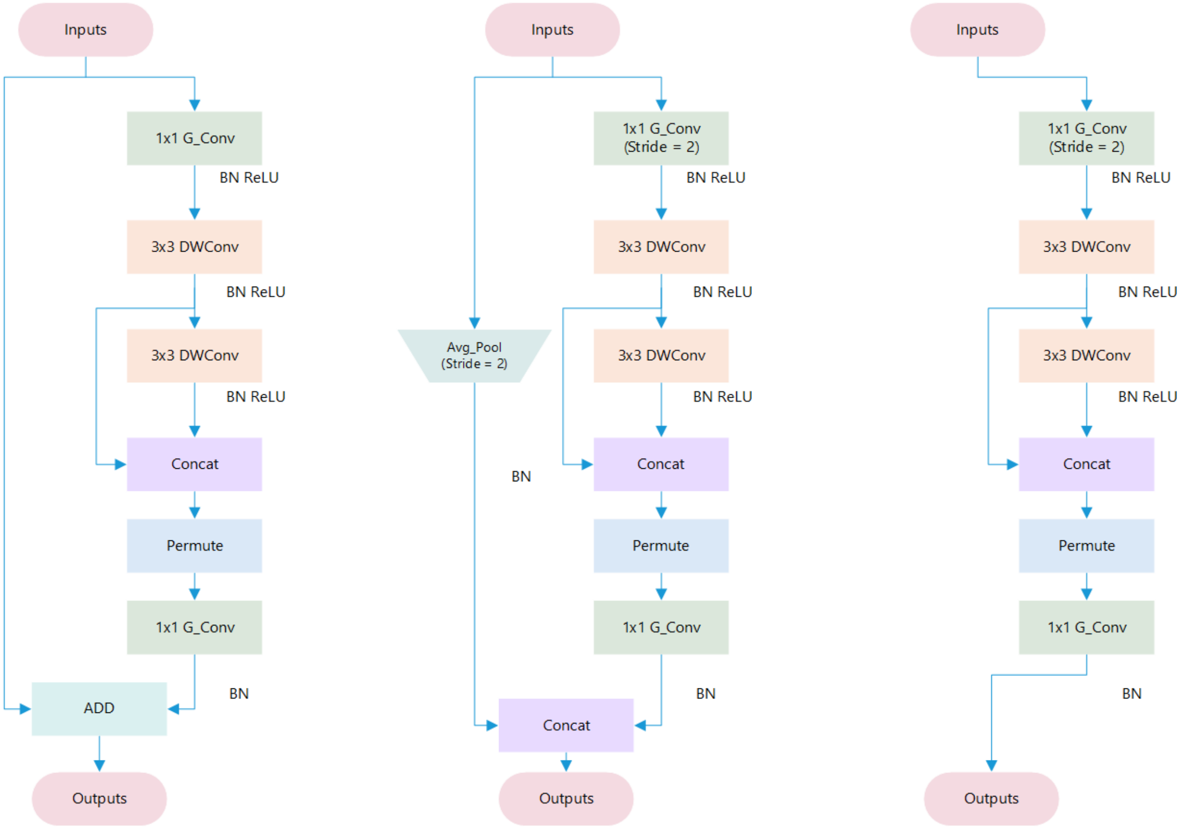
To deal with the situation, machine learning approaches, in particular convolution neural network(CNN), have noticeable advantages. It uses a large number of samples to progressively determine the system parameters in order to detect various object features. The networks such as VGG [7] and ResNet [8] have been reported to solve complex object recognition problems. Normaly, CNN requires a large number of layers, which, in consequence, needs a large number of parameters and a huge computation volume to achieve a good performance. Improving the computation efficiency of CNNs requires critical research effort. Some network pruning methods to reduce computation complexity are reported in [9–11]. In MobileNetV2 [12] and ShuffleNet [13], depthwise convolutions are used in their modules in an attempt to make the computation more efficient. Architecture Xception [14], a linear stack of depthwise convolution layers with residual connections, resulted in some gains in classification performance on the ImageNet dataset.

In convolution neural networks, different modes of convolutions transform the properties of the input data in different ways. It's important to control various data of different nature for appropriate modes of convolutions to extract features of different orders. Based on this idea, we propose, in this paper, a convolution module, named SdcBlock, and a CNN architecture, named SdcNet, with a view to reducing significantly the computation volume without sacrificing the processing quality. The SdcBlock, in which successive depthwise convolutions supported by appropriate data management are applied, is specifically designed for the computation with different types of data. The block is modularized to facilitate its applications in varieties of networks.

## II. Proposed Method

Feature extraction by CNN is performed by means of progressive filtering through a good number of convolution layers. In each of the layers, new feature vectors are generated, based on a large volume of the input data, in a way that the information relevant to the object features is extracted, composed, strengthened, and/or concentrated, while filtering out those irrelevant. Because of rich variations in the features, a large number of filtering kernels are often used in a single layer to increase the chance of extracting different features, which will certainly increase computation complexity, but not necessarily the concentration of the relevant feature information in the generated vectors.

To build effective convolution layers with a maximized capacity of extracting critical feature information, it's important to look into the different convolution modes and to direct the data to the appropriate convolution layers. In general, an input of $N_I$ channels can be transformed to an output of $K$ channels by a convolution with $K$ kernels. The following modes are the most commonly used.



(a) SdcBlock(Stride=1)

(b) SdcBlock-S2(Stride=2)

(c) SdcBlock-S2-F(Stride=2)

Fig. 1. SdcBlock Modules. G_Conv represents group convolution and DWConv represents dethwise convolution.

- *Standard convolution* [15]. In this mode, each of the $K$ convolution kernels is applied to all the $N_I$ input channels to generate one output channel.
- *Group convolution (G-Conv)* [16]. The $N_I$ input channels is divided into $g$ groups, and $K$ convolution kernels are also divided into $g$ sets. Each group of the input data is convolved with a set of $K/g$ kernels. The standard convolution can be seen as the specific case, with $g = 1$, of group convolution.
- *Depthwise convolution (DW-Conv)* [14]. It is, in fact, the conventional 2-D spatial convolution. In the context of CNN, it can be seen as another special case of group convolution, in which $g = N_I$, *i.e.*, one channel per group, and each of the $K$ kernels is applied only to one input channel, requiring $g = N_I = K$.

With given $N_I$ and $K$, the standard convolution is the most computation-demanding, as it generates each of the output vectors based on the data sampled from all the input channels. On the contrary, the depthwise mode is the least computation-demanding, and each of its output vectors is produced exclusively based on the data of a single channel. This exclusivity can facilitate the control of the data flow of individual channels. By using a preceding 1x1 convolution to organize its input data channels, the depthwise convolution can also process data of multiple channels.

The convolution module and architecture proposed in this paper have been designed to make the best use of the input data. It is done by means of a data management designed to optimize successive depthwise convolution results. The purpose is to generate feature vectors having a higher density and more varieties of the information critical for the classification.

### A. Modules

The proposed convolution module, named SdcBlock (Successive Depthwise Convolution Block), is illustrated in Fig. 1. If an input signal is composed of a large amount of data, its features can be represented not only by its original data form but also by the data resulting from filtering operations of different orders. Successive convolutions performed to the same signal can generate such feature information. Hence, the pivotal part of this module is the successive depthwise convolutions to implement the principle of multiple order processing in each channel of the input data.

The module is mainly composed of three parts for the three functions, Successive Depthwise Convolutions (Sdc), data preparation, and arrangement of convoluted data, respectively.

1) *Successive depthwise convolutions (Sdc).* They are performed by two boxes, indicated by 3x3 DWconv found in Fig. 1(a), used to generate the data of the first

TABLE I
DETAILS OF SDCNET CONFIGURATIONS

| Layer | Image Output Size | Stride | Repeat Times | Output Channels | |
|---|---|---|---|---|---|
| | | | | SdcNet-G4-L($g = 4$) | SdcNet-G3-S($g = 3$) |
| Input image data sized 32x32, 3 channels. | | | | | |
| G-Conv($g = 3$)* | 32x32 | 1 | 1 | 36 | 36 |
| Stages 1 | 32x32 | 1 | 1 | 24 | 24 |
| Stages 2 | 32x32 | 1 | 2 | 36 | 24 |
| Stages 3 | 16x16 | 2 | 1 | 72 | 36 |
| | 16x16 | 1 | 2 | 72 | 36 |
| Stages 4 | 8x8 | 2 | 1 | 96 | 72 |
| | 8x8 | 1 | 3 | 96 | 72 |
| Stages 5 | 8x8 | 1 | 3 | 144 | 96 |
| Stages 6 | 4x4 | 2 | 1 | 300 | 150 |
| | 4x4 | 1 | 2 | 300 | 150 |
| Stages 7 | 4x4 | 1 | 1 | 600 | 300 |
| Avg Pool** | 1x1 | 2 | 1 | 600 | 300 |
| FC*** | 1x1 | | 1 | 10 | 10 |
| Complexity**** | | | | 106.1M | 56.55M |

* G-Conv stands for group convolution. The kernel size of the group convolution is 3x3.
** The kernel size of the average pooling is 4x4.
*** 10 is for CIFAR-10 dataset and 100 is for CIFAR-100 dataset.
**** The complexity is evaluated with FLOPs for the dataset CIFAR-10, *i.e.* the number of floating-point multiplication-adds.

and second order filtering operations. They must be applied solely to the data of the same channel, for which only depthwise convolution is suitable. If the module is placed in the entry part of a CNN to process raw image data, the successive depthwise convolutions will generate the first and second order gradient maps in order to obtain various low-level features. If the module is placed in the middle or final parts of the convolution stages, these convolution operations will produce vectors of more dimensions and levels containing high order feature information.

The extracted feature information of each of the two convolutions needs to be carefully preserved. Hence, the two sets of the convolution results are concatenated, instead of being summed up.

In the current version of SdcBlock illustrated in Fig. 1(a), the kernel size of the two succisive depthwsise convolutions is 3x3. Batch normalization [17] and non-linear function ReLU [18] are applied after each of the convolutions.

2) *Data preparation for the successive depthwise convolutions.* Since the pivotal part in the proposed module is the depthwise convolutions performed in the individual input channels, it is important to prepare the input data in order to optimize the convolution results. In SdcBlock, a set of 1x1 convolution kernels are applied to the input data, as illustrated in Fig. 1(a), for the preparation. By doing so, the data can be scaled to suit the succeeding convolution and, meanwhile, the number of the input data channels are expanded to match that required in the successive depthwise convolutions. If $N_I$ input channels are expanded to $E * N_I$ channels, $E$ is the expansion number, used as one of the hyperparameters.

In the current version of the SdcNet, the group convolution mode is used in the first set of 1x1 convolution in each block, as illustrated in Fig. 1(a), in the data preparation so that the input channels can be grouped according to their nature. Moreover, the group convolutions can reduce the computation complexity significantly, with respect to the standard convolution. It should also be mentioned that batch normalization and non-linear function ReLU are applied after the first 1x1 group convolution.

3) *Data arrangement after the successive depthwise convolutions.* As mentioned previously, the results from the two successive depthwise convolution operations are concatenated. The data produced by the different depthwise convolutions are placed separately in different sections of the vectors generated by the concatenation. Rearranging the vector elements so that every segment in the vectors has elements randomly taken from the two convolutions results may benefit the following operations. Thus, the results of the concatenation are permuted to mingle the data produced by the two depthwise convolutions. Another group convolution, the second 1x1 kernel convolution illustrated in Fig. 1(a), is applied to combine the data from 2x$E$x$N_I$ channels to $N_O$ output channels. A batch normalization is followed after the second 1x1 group convolution.

The proposed module can be varied by a choice of the hyper parameter *stride*. In the basic version of the proposed SdcBlock illustrated in Fig. 1(a), it takes *stride=1*. A residual operation [8] is applied to the inputs and the rearranged convolved results. There are three ReLUs in each SdcBlock to ensure the non-linear ability of the module and there is no ReLU after the addition. In some cases, convolution layers of *stride=2* can be used in order to reduce computation cost. The SdcBlock with *stride* = *2*, named SdcBlock-S2, is shown in Fig. 1(b). In order to compensate for the eventual information lost due to the *stride* = *2* convolution, a concatenation of the average-pooled input data and the rearranged convolved data is performed. The output of SdcBlock-S2 contains both the sampled input information and the successive depthwise convolution results. A variation of the SdcBlock-S2, named SdcBlock-S2-F (Feature), has also been proposed and the procedure is shown in Fig. 1(c). Compared to SdcBlock-S2, the final output data result exclusively from the successive depthwise convolutions without being combined with the input.

### B. Network architecture

A convolution neural network architecture mainly composed of a stack of SdcBlocks is named SdcNet. Two SdcNets have been designed for the CIFAR image classification and the details of the designs are presented in Table I. Each of these SdcNets is composed of seventeen SdcBlocks grouped in seven stages. The hyperparameters in each stage are made the same, except *stride*.

In a SdcBlock, the hyperparameters $g$, the number of groups, and $E$, the expansion number, are used to control the computation volume. Furthermore, the volume is also closely related to the number of output channels in each stage. The two SdcNet networks, specified in Table I, differ in the number of groups and the number of output channels in stages. In SdcNet-G4-L, ”G4” indicates $g = 4$ is applied to all the stages whereas $g = 3$ is used in SdcNet-G3-S. As the former has a larger number of output channels than the latter, the name of the former ends with ”L”, standing for ”larger”, and that of the latter with ”S”, standing for ”smaller”. Besides, the basic SdcBlock is used in case of *stride=1* and SdcBlock-S2 is used in case of *stride=2*, unless otherwise specified.

## III. Performance Evaluation

To evaluate the performance of SdcNet, a set of experiments have been performed with CIFAR-10 and CIFAR-100 image classification datasets.

### A. Experiment Conditions

*1) Datasets.* The CIFAR dataset [19] (Canadian Institute For Advanced Research) is a collection of images that are commonly used to train machine learning and computer vision algorithms. CIFAR-10 and CIFAR-100 datasets contain 60000 RGB images of 32x32 pixels in 10 classes and 100 classes, respectively. For the CIFAR-10 dataset, the training set has 5000 images per class and the testing set contains 1000 randomly-selected images from each class. In the CIFAR-100 dataset, there are 500 training images per class and 100 testing images per class.

*2) Network Configurations.* Four different versions of SdcNets have been tested. The details of the network configurations are specified in Table I and the explanation of the network names is found in Section II B. These SdcNets differ in the number of groups in each stage, the number of output channels and types of used blocks. In cases of SdcNet-G4-L-F and SdcNet-G3-S-F, SdcBlock-S2-F are used for all the blocks in case of *stride=2*. In all the four networks, $E$ is equal to 6.

TABLE II
Comparison of classification error rate on CIFAR- 10 (C-10) and CIFAR-100 (C-100) with existing CNNs

| Model | FLOPs | Params | C-10 | C-100 |
|---|---|---|---|---|
| VGG-16-pruned[9] | 206M | 5.40M | 6.60% | 25.28% |
| VGG-19-pruned[10] | 195M | 2.30M | 6.20% | - |
| VGG-19-pruned[10] | 250M | 5.00M | - | 26.20% |
| ResNet-56-pruned[11] | 62M | - | 8.20% | - |
| ResNet-56-pruned[9] | 90M | 0.73M | 6.94% | - |
| ResNet-110-pruned[9] | 213M | 1.68M | 6.45% | - |
| ResNet-164-B-pruned[10] | 124M | 1.21M | 5.27% | 25.28% |
| SdcNet-G3-S-F | 56.55M | 1.09M | 5.79% | 25.83% |
| SdcNet-G3-S | 55.12M | 1.04M | 5.60% | 25.01% |
| SdcNet-G4-L-F | 106.1M | 2.61M | 5.27% | 23.52% |
| SdcNet-G4-L | 103.3M | 2.53M | 5.24% | 23.12% |

*3) Training Details.* The network has been trained with mini-batch size of 128 for 300 epochs. The cross entropy between the distribution of the network outputs and that of the ground truth data has been calculated to measure the loss. It is defined as $H(y,p) = \Sigma_i y_i log(p_i)$, where $y_i$ is the ground truth data and $p_i$ is the model outputs. The stochastic gradient descent (SGD) [20] optimization method has been applied using similar optimization parameters as those in [8]. Besides, Nesterov momentum with a momentum weight of 0.9 and a weight decay of 0.0001 has been adopted. A variable learning rate starting from 0.1 has been used and it was reduced to 0.002 following a non-linear cosine-curve in the 300 epochs [21]. The simple data augmentation in [22] has also been applied for training, four zero pixels are padded on each side, and then a 32x32 crop is randomly sampled from the padded image or its horizontal flip. The weights of the network have been initialized by using the method reported in [23], $i.e.$, the weights being initialized in such a way that the variance between inputs and outputs is the same in each layer.

### B. Results on CIFAR 10 and CIFAR 100

The test result is presented in Table II. They are compared with those given by the VGG-pruned [9, 10] and ResNet-pruned [9–11]. In general, the error rates achieved by SdcNets are not above 5.79% in CIFAR-10 and 25.83% in CIFAR-100, which are better than those given by ResNet-pruned and VGG-pruned nets under the similar computation conditions, in terms of Flops.

With the restriction of low computation cost, SdcNet can achieve an error rate of 5.60% in CIFAR-10 with 55M Flops in comparison with 8.20% by ResNet-56 with 62M Flops. In terms of quality of processing, the error rate achieved by SdcNet can be as low as 5.24% in CIFAR-10 using 103M Flops, versus 5.27% given by ResNet-164-B-pruned using 124M Flops. These results confirm that the proposed modules and networks have a better performance in terms of efficiency.

## IV. Conclusion

In this paper, a computation-efficient convolutional module, named SdcBlock, has been proposed and based on it, the convolution network SdcNet introduced for object recognition tasks. The pivotal part of the SdcBlock is the optizimized successive depthwise convolutions supported by the appropriate data management to generate vectors containing higher density and more variety feature information. The hyperparameters can be adjusted for varieties of tasks under different computation restrictions without significantly jeopardizing the performance. Examples of SdcNet have been designed and tested. It has been demonstrated that SdcNet achieved an error rate of 5.60% in CIFAR-10 with only 55M Flops, and also reduced further the error rate to 5.24% using a moderate volume of 103M Flops. The computation efficiency of the SdcNet has been confirmed although the results can be further improved by fine adjustments of hyperparameters.

## Acknowledgment

This research was enabled in part by support provided by Compute Canada (www.computecanada.ca). Computations were made on the supercomputers Helios and Graham managed by Calcul Quebec and Compute Canada.

## References

[1] M. Bojarski, D. Del Testa, and Dworakowski, "End to end learning for self-driving cars," *arXiv preprint arXiv:1604.07316*, 2016.

[2] O. M. Parkhi, A. Vedaldi, and A. Zisserman, "Deep face recognition." in *Proc. of British Machine Vision Conference*, vol. 1, no. 3, 2015, p. 6.

[3] A. Graps, "An introduction to wavelets," *IEEE computational science and engineering*, vol. 2, no. 2, pp. 50–61, 1995.

[4] D. G. Lowe, "Object recognition from local scale-invariant features," in *Proc. of the Seventh IEEE International Conference on Computer Vision*, vol. 2, 1999, pp. 1150–1157.

[5] O. Chapelle, P. Haffner, and V. N. Vapnik, "Support vector machines for histogram-based image classification," *IEEE transactions on Neural Networks*, vol. 10, no. 5, pp. 1055–1064, 1999.

[6] P. Viola and M. Jones, "Fast and robust classification using asymmetric adaboost and a detector cascade," in *Proc. of Advances in Neural Information Processing Systems*, 2002, pp. 1311–1318.

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

[8] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 770–778.

[9] H. Li, A. Kadav, and I. Durdanovic, "Pruning filters for efficient convnets," *arXiv preprint arXiv:1608.08710*, 2016.

[10] Z. Liu, J. Li, Z. Shen, G. Huang, S. Yan, and C. Zhang, "Learning efficient convolutional networks through network slimming," in *Proc. of IEEE International Conference on Computer Vision*, 2017, pp. 2755–2763.

[11] Y. He, X. Zhang, and J. Sun, "Channel pruning for accelerating very deep neural networks," in *Proc. of IEEE International Conference on Computer Vision*, vol. 2, 2017, p. 6.

[12] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Inverted residuals and linear bottlenecks: Mobile networks for classification, detection and segmentation," *arXiv preprint arXiv:1801.04381*, 2018.

[13] X. Zhang, X. Zhou, M. Lin, and J. Sun, "Shufflenet: An extremely efficient convolutional neural network for mobile devices," *arXiv preprint arXiv:1707.01083*, 2017.

[14] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," *arXiv preprint arXiv:1610.02357*, 2016.

[15] B. Cabral and L. C. Leedom, "Imaging vector fields using line integral convolution," in *Proc. of the 20th Annual Conference on Computer Graphics and Interactive Techniques*, 1993, pp. 263–270.

[16] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proc. of Advances in Neural Information Processing Systems*, 2012, pp. 1097–1105.

[17] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *arXiv preprint arXiv:1502.03167*, 2015.

[18] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, p. 436, 2015.

[19] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," 2009.

[20] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Backpropagation applied to handwritten zip code recognition," *Neural computation*, vol. 1, no. 4, pp. 541–551, 1989.

[21] I. Loshchilov and F. Hutter, "Sgdr: stochastic gradient descent with restarts," *Learning*, vol. 10, p. 3, 2016.

[22] C.-Y. Lee, S. Xie, P. Gallagher, Z. Zhang, and Z. Tu, "Deeply-supervised nets," in *Prof. of Artificial Intelligence and Statistics*, 2015, pp. 562–570.

[23] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *Proc. of IEEE International Conference on Computer Vision*, 2015, pp. 1026–1034.