Computer Vision and Image Understanding
journal homepage: www.elsevier.com

SdcNet for Object Recognition

Yunlong Ma, Chunyan Wang*

Department of Electrical and Computer Engineering, Concordia University, Montreal, Canada

ABSTRACT

In this paper, a CNN architecture for object recognition is proposed, aiming at achieving a good processing-quality at the lowest computation-cost. The work includes the design of SdcBlock, a convolution module, for feature extraction, and that of SdcNet, an end-to-end CNN architecture. The module is designed to extract the maximum amount of high-density feature information from a given set of data channels. To this end, successive depthwise convolutions (Sdc) are applied to each group of data to produce feature elements of different filtering orders. To optimize the functionality of these convolutions, a particular pre-and-post-convolution data control is applied. The pre-convolution control is to organize the input channels of the module so that the depthwise convolutions can be performed with a single channel or a combination of multiple data channels, depending on the nature of the data. The post-convolution control is to combine the critical feature elements of different filtering orders to enhance the quality of the convolved results. The SdcNet is mainly composed of cascaded SdcBlocks. The hyper-parameters in the architecture can be adjusted easily so that each module can be tuned to suit its input signals in order to optimize the processing-quality of the entire network. Three different versions of SdcNet have been proposed and tested using CIFAR dataset, and the results demonstrate that the architecture gives a better processing-quality at a significantly lower computation cost, compared with networks performing similar tasks. Two other versions have also been tested with samples from ImageNet to prove the applicability of SdcNet in object recognition with images of ImageNet format. Also, a SdcNet for brain tumor detection has been designed and tested successfully to illustrate that SdcNet can effectively perform the detection with a high computation efficiency.

1. Introduction

Object recognition is widely used in computer-vision-based applications, such as security and medical systems (Hampapur et al., 2005; Wang et al., 2019; Pontil and Verri, 1998; Brunelli and Poggio, 1993). In general, it is performed by (i) extracting various features and (ii) classifying the objects based on the extracted features. It is a challenging task since there can be a huge amount of feature variations for the same object class, whereas similar feature elements can appear in different object classes.

Object recognition systems can be designed by two approaches, knowledge-based filtering and machine learning. The knowledge-based approach (Lowe, 2004; Sutton and Hall, 1972; Chapelle et al., 1999), using specially designed filters for specific feature elements, is normally computation-efficient, but has a limitation on handling a large number of variations in object features. Machine learning approach, in particular convolutional neural networks(CNNs) (LeCun et al., 1998), can be promising. This kind of networks uses a large number of samples to progressively determine the system parameters in order to be able to detect various features corresponding to target objects (Polyak and Wolf, 2015; Yang et al., 2017; Simonyan and Zisserman, 2014; He et al., 2016). However, CNN normally requires a large number of parameters to deal with feature variations in order to achieve good performance, which, in consequence, leads to a huge computation volume and limits its implementation and applications.

Since the development of object recognition systems is highly demanded, a lot of research efforts have been made to improve the performance. The convolutions in VGGNet (Si-

*Corresponding author
e-mail: chunyan@ece.concordia.ca (Chunyan Wang)

monyan and Zisserman, 2014) are performed with simple 3×3 kernels to reduce the complexity in computation. ResNet uses residual modules to extend the depth of CNNs for better performance (He et al., 2016). In XceptionNet, for a better computation efficiency, the convolutions are made depthwise, assuming that the cross-channels correlations and spatial correlations in the feature maps can be decoupled (Chollet, 2016). Pruning a trained network (Li et al., 2016; Liu et al., 2017; He et al., 2017) can reduce the computation volume in the testing process, but training the un-pruned network can be very much computation-consuming. In MobileNetV2 (Sandler et al., 2018), a combination of 1×1 convolution and 3×3 depthwise convolution, instead of 3×3 standard convolution, is used to make both testing and training processes computation efficient. ShuffleNet (Zhang et al., 2018) also features similar combinations, but the 1×1 convolutions are specified to be group-wise and channels are shuffled before the 3×3 depthwise convolution. In Squeeze-and-Excitation Networks (Hu et al., 2018), a dynamic channel-wise feature recalibration is performed to improve the representational capacity, and Deep Expander Networks (Prabhu et al., 2018) use expander graphs that give strong theoretical guarantees on connectivity.

The objective of the work presented in this paper is to develop a computation-efficient CNN architecture for object recognition tasks. It seeks to use simple architectures to achieve a good recognition quality with a view to enabling the implementation in diverse applications. As the recognition quality is closely related to the quality of the image features extracted, it is important to design the convolution layers with the principle of 2-D filtering for efficient feature extraction. The work is composed of two parts, (i) the design of CNN modules for feature extraction, and (ii) end-to-end CNN architectures for object recognition and other processing tasks. The module and the architecture are designed following an investigation of different convolution modes in CNNs and an analysis of the data in the different computation stages.

The paper is structured as follows. A detailed description of the proposed module SdcBlock is presented in Section 2. The design of five versions of the CNN architecture SdcNet with different emphases on performance is presented in Section 3. The performance evaluation of these networks is presented in Section 4. In Section 5, a design example of SdcNet for brain tumor detection is presented to illustrate the applicability of SdcNets for processing tasks other than object recognition.

2. Proposed Module SdcBlock

The convolution module is designed to generate feature vectors from 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. A good use of different convolution modes can improve the processing quality while reducing computation volume. In general, an input data of N_I channels can be transformed to an output data of N_O channels by a convolution with N_O kernels in one of the following three modes.

- Standard convolution(S-Conv). In this mode, each of the N_O convolution kernels is applied to all the N_I input channels to generate one output channel, and mathematically it can be expressed as

$$\vee j \in [1, 2, ..., N_O]:$$
$$S\text{-}Conv(I, K)_{m,n,j} = \sum_{i=1}^{N_I} \sum_{p=-a}^{a} \sum_{q=-a}^{a} I_{m+p,n+p,i} K_{p,q,i,j} \quad (1)$$

  where K is a set of N_O convolution kernels, each of which is sized $(2a+1) \cdot (2a+1) \cdot N_I$. If K is applied to the input I, sized $H_I \cdot W_I \cdot N_I$, with $stride = 1$, the computation volume, measured by MACs (multiply-accumulate operations), will be

$$(2a+1)^2 \cdot H_I \cdot W_I \cdot N_I \cdot N_O \quad (2)$$

  Standard convolutions are often used when all the input channels are correlated.

- Depthwise convolution (DWConv). It is, in fact, the conventional 2-D spatial convolution with the weights progressively updated in the training process. Each kernel, sized $(2a+1) \cdot (2a+1)$, is applied to a single input channel. The size of K is $(2a+1) \cdot (2a+1) \cdot 1 \cdot N_O$. Its mathematical expression can be written as follows.

$$\vee j \in [1, 2, ..., N_O]:$$
$$DWConv(I, K)_{m,n,j} = \sum_{p=-a}^{a} \sum_{q=-a}^{a} I_{m+p,n+p,j} K_{p,q,1,j} \quad (3)$$

  To generate N_O output channels from the same number of the inputs, the computation volume required for the depthwise convolution is

$$(2a+1)^2 \cdot H_I \cdot W_I \cdot N_O \quad (4)$$

  which is only $1/N_I$ of that needed for the standard convolution.

  Depthwise convolutions, if performed successively to a single input channel, can generate feature maps of different filtering orders.

- Group convolution (G-Conv). The N_I input channels are divided into g groups and each of them gets convolved with a kernel sized $(2a+1) \cdot (2a+1) \cdot N_I/g$ to produce one out channel. The size of K is $(2a+1) \cdot (2a+1) \cdot N_I/g \cdot N_O$. A standard convolution can be seen as a special case with $g = 1$, whereas a depthwise convolution is another special case with $g = N_I$. A group convolution reduces computation volume by a factor of g, compared with the standard convolution. It also gives a flexibility to select and to group input channels according to their properties.

2.1. Processing in SdcBlock

The basic scheme of the proposed convolution module, SdcBlock (Successive Depthwise Convolution Block), is illustrated in Fig. 1. A preliminary version of the work has been presented at a conference (Ma and Wang, 2018). In this module, three functions, Successive Depthwise Convolutions (Sdc), data preparation for the convolutions and arrangement of the convoluted data to form the output, are performed.

2.1.1. Successive Depthwise Convolutions(Sdc)

Object recognition needs feature information of different natures that can be generated by filtering operations of different orders. Successive convolutions are used in this module to efficiently generate such features. These convolutions must be applied exclusively to the same set of data, for which only the depthwise convolution mode is suitable. Hence the Successive Depthwise Convolutions(Sdc) make the pivotal part in this module. They are indicated by the two boxes of *3×3 DWConv* shown in Fig. 1. Let I denote the input of the module, X denote the input of the first depthwise convolution, X' and X'' the results of the first and second order filtering operations, respectively. The features of the input I are represented in different ways in X, X' and X'', and the output of the module are produced based on them. It should be mentioned that each channel in X can be formed either from a single channel or a group of multiple channels in I.

If the module is placed in an early processing part of a network to handle 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 middle or final stages of a CNN, these operations will produce vectors containing higher order feature information.

In the basic version of SdcBlock illustrated in Fig. 1, the kernel size of the two successive depthwsise convolutions is 3×3. It can, however, be extended to other sizes if needed.

2.1.2. Data Preparation for the Successive Depthwise Convolutions(Sdc)

The component of data preparation is to convert the input data I to the data set X to meet the requirements of the successive depthwise convolutions. It is done by an 1×1 group convolution applied to I, as illustrated in Fig. 1, for the purposes stated as follows.

- Scaling the input elements. This 1×1 convolution provides a scaling function to the input channels of the module and the weights can be adjusted to facilitate the feature extraction in the succeeding convolutions.
- Adjustment of the number of data channels. By applying the 1×1 group convolution, the input data I with N_I channels can be expanded to $E \times N_I$ channels, if $E > 1$, for the succeeding successive depthwise convolutions.
- Grouping the input channels. As mentioned previously, the output of the two successive convolutions, namely X' and X'', are produced exclusively from the input set X. Each channel in X can be formed by a single channel, or a group of multiple channels, from the input data I. It should be mentioned that by grouping multiple channels from I, more input information is involved in each channel in X and the succeeding depthwise convolutions can generate feature maps containing information of higher density.

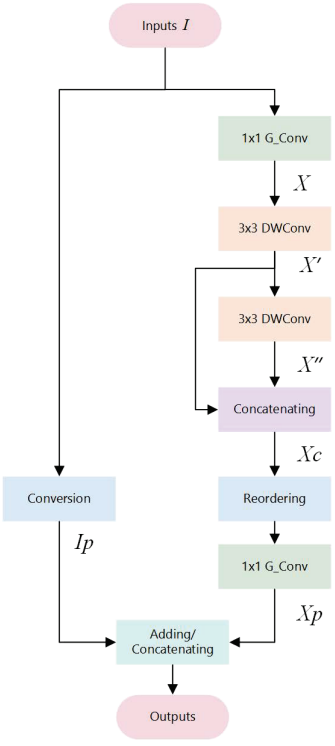

Fig. 1. Basic SdcBlock. G_Conv denotes group convolution and DWConv denotes dethwise convolution.

This component of 1×1 group convolution allows the data scale and format to be changed easily. In particular, by grouping and expending the input channels, it is possible to effectively involve multiple input channels in each depthwise convolution to optimize the filtering process.

2.1.3. Data Arrangement to Generate the Output

This part of the SdcBlock is used to handle the three sets of data, *i.e.* I, the input of the module, X' and X'', produced respectively by the two successive depthwise convolutions. The data X' and X'' are combined by means of concatenation followed by an 1×1 group convolution, as shown in Fig. 1, to generate Xp. This group convolution is used to perform, from mathematic point of view, additions of weighted channels. From signal processing point of view, it is to implement a series of modulations of the data of X' by those of X'', or vice versa, with learnable modulation depths, if channels from both X' and X'' are involved in each of the additions. The channels of Xp can thus carry enhanced features from X' and X''.

It should be mentioned that, as the data of X' and X'' are concatenated, they are sequenced naturally in Xc. A channel reordering is performed to re-index the sequence of these channels to make it possible that each group in this group convolution has channels from both X' and X''.

The output data of the module is generated from I and Xp. In general, it can be done in two different ways.

- *Addition*. The convolved data are added to the input data of the block, which can result in another feature enhancement and/or a generation of new features.

- *Concatenation*. The two sets of data are concatenated to preserve feature information of the three different orders for further processing.

The decision to use the addition or concatenation is based on the nature of the input data I and the purposes of the processing in the block. In order to implement the combination, the two sets of data, I and Xc, need to have the same dimensions and formats. Hence, the 1×1 group convolution is also used to compress or extend the concatenated data Xc to a desirable data form Xp. Also, a similar process is applied to I, in order to meet the requirements of dimensions and channel sequences.

2.2. Variations of SdcBlock

As convolutions with $stride = 1$ normally give more precise filtering results than those with $stride = 2$ and the dimension of feature map in each channel remains unchanged in the operations, $stride = 1$ is used in the basic version of SdcBlock. However, in some cases, the module with $stride = 2$ is needed to downsize the feature maps. Since there is information redundancy in the feature maps, the size can be reduced without a high risk of information loss. Meanwhile, this down-sizing decreases overall computation cost and increases the convergence speed in the training process. Hence SdcBlock with $stride = 2$ is used in CNN processes to improve the computation efficiency and the concentration of feature information in each channel.

In SdcBlock-S2 shown in Fig. 2, the dimension down-sizing by means of $stride = 2$ is used only in the first 1×1 group convolution, whereas $stride = 1$ is applied in the other convolutions. The input channels of the module and the convolved data, *i.e.*, I_p and X_p, are merged by concatenation to better preserve the feature information in these channels. For this concatenation, the input channels need a dimension down-sizing while maintaining the nature of the data. To this end, an average pooling is applied to the input channels to convert I to I_p, as shown in Fig. 2. It should, however, be mentioned that other pooling methods can also be used for the conversion, depending on the nature of the feature data.

In this version, the output contains both the sampled input information and the successive depthwise convolution results. By adjusting the number of kernels to generate X_p, one can determine the proportion of the convolved data in the output of the module. If the number of channels of X_p is made more than that of I_p, X_p will be dominant in the output.

One can create another version of SdcBlock with $stride = 2$ by removing I_p. In this version, the output is generated exclusively by the successive depthwise convolutions. It can be useful in some specific filtering processes.

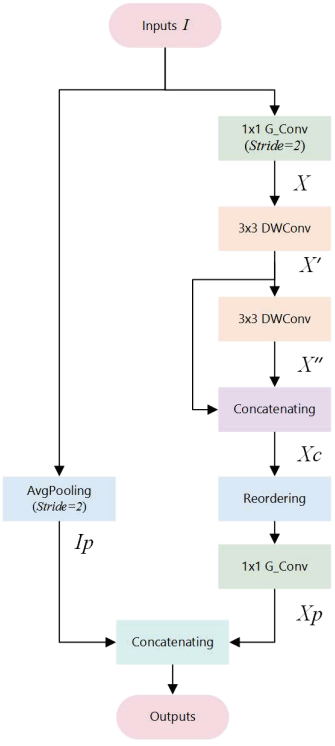

Fig. 2. SdcBlock with *Stride=2*

3. SdcNet Design

SdcNet is a CNN architecture proposed with a view to optimizing the processing for object recognition tasks. It is composed of cascaded SdcBlocks. One can use a number of hyper-parameters in each SdcBlock to achieve the optimization. The basic scheme of SdcNet was presented very briefly in (Ma and Wang, 2018). As more work in the network design has been carried out after it, the presentation of the SdcNet design in this section is, hopefully, insightful to provide readers with precise information about the networks in detail.

The general scheme of the proposed SdcNet architecture is shown in Fig. 3. From an input image, it generates the data representing classification decisions, such as object classification labels. The network can be divided into three functional parts for early processing, feature extraction and decision making. The first part is to extract low-level features and is designed to suit the characters of the input images. The high-level features are extracted by the feature extraction part, and are used in the third part to generate the final decisions. The decision making part is composed of a conversion layer and a fully connected layer, as shown in Fig. 3.

Since the quality of the extracted feature information can make a significant difference in the final decision making, it is critical to design the early processing and feature extraction parts to extract varieties of feature information and to present it in a high-density form. The design of SdcNet is to make good use of SdcBlock by determining a number of hyper-parameters in each block according to its input signal and its specific computation purpose. These hyper-parameters are listed as follows.

- The group number g. In a SdcBlock, it is used to organize

the data channels in the group convolution, and should be determined according to the nature and correlation of the data channels in each layer. Meanwhile, it is one of the elements determining the total computation cost since convolutions with a larger value of g require a smaller amount of computation.

- The number of output channels. In each SdcBlock, this number should be sufficiently large to extract enough varieties of feature elements for further processing. However, it should be mentioned that excessively increasing the number of output channels may have adverse effect, such as increasing the computation volume, reducing the information density and decreasing the rate of convergence in the training process.

- The hyper-parameter *stride*. SdcBlocks with *stride* = 1 are used in the stages where precise convolution results are required. In SdcBlocks with *stride* = 2, the channel dimension is down-sized by three quarters, which reduces the computation volume and helps to make the feature information more concentrated if such blocks are used in an appropriate manner to minimize data loss.

- The combinations of the data from different filtering operations. As mentioned previously, it can be either concatenation or addition.

- The expansion parameter E is the ratio of the number of the output channels over that of the inputs of a convolution layer. It is applied to the first 1×1 group convolution in a SdcBlock to expand the number of input data channels of the succeeding 3×3 depthwise convolutions.

- The rearrangement of the convolved data X' and X'' by means of the second 1×1 group convolution in the module. These data channels can be re-indexed by randomly shuffling or in a specific way based on the nature, such as color, of the data in each SdcBlock.

3.1. Processing in SdcNet

The early processing part of a SdcNet contains an initial convolution layer and one SdcBlock, as shown in Fig. 3. A group convolution is performed in the initial convolution layer. As the input image is composed of three color channels, the hyper-parameter g is chosen to be 3 so that each group of convolution kernels is applied to an individual color channel to generate monochromatic features. The SdcBlock used in this part is to organize and enhance these early feature elements for the succeeding feature extraction.

The feature extraction part is a stack of SdcBlocks, as shown in Fig. 3. The input channels of this part are low-level feature maps of large-dimension generated by the early processing part. The feature elements, which may contribute to the final decisions, are normally dispersed widely in the input channels. This part is to perform layers of filtering operations to generate,

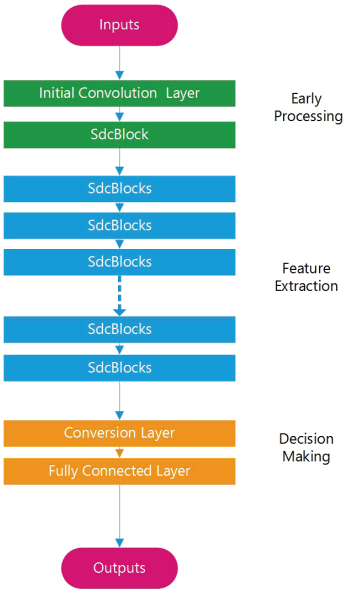

Fig. 3. General scheme of SdcNet.

from these low-level feature maps, condensed high-level features represented in small-dimension channels that will be applied to the final decision part. The number of the output channels of the feature extraction part should be sufficiently large to accommodate different kinds of specific feature information. Hence, the stack of SdcBlocks are used to convert a small number of large feature maps into a large number of small-format data channels.

In order to implement the process of the feature extraction, the number of the output channels in each stage increases progressively, and SdcBlocks with *stride* = 2, instead of pooling, are applied in particular stages to downsize the feature maps. However, in general SdcBlokcs with *stride* = 2 are not cascaded successively to minimize the risk of information loss.

Based on the principles described above, three SdcNets, namely SdcNet-S, SdcNet-M and SdcNet-L, have been developed with different emphases for CIFAR classification tasks. The network SdcNet-S is considered as the standard version of SdcNet and it has the lowest computation cost among the three. SdcNet-M and SdcNet-L are designed to achieve higher processing quality.

Besides the three above-mentioned SdcNets aiming at images of small-format and low-resolution, two other versions of SdcNet are also designed to process images of larger format, such as those in ImageNet, and to have a finer classification.

3.2. Three Versions of SdcNet for CIFAR Datasets

3.2.1. SdcNet-S

The detailed configuration of SdcNet-S is shown in Table 1. As mentioned previously, this network can be divided into three functional parts. The early processing part consists of a group convolution layer with 3×3 kernel size and a SdcBlock. The succeeding feature extraction part is composed of sixteen SdcBlocks divided into six stages, namely Stage 1 to Stage 6, as shown in Table 1. A global average pooling layer and a fully connected layer are used to generate the final decisions.

Table 1. Details of SdcNet-S Configuration
Input image: sized 32x32 pixels and having 3 channels.

	Stride	Repeat Times	Group No. g	No. of Output Channels	Size of Output Channels	No. of Weights	Computation Complexity(Flops)
G-Conv*	1	1	3	36	32x32	3.2K	2.91M
SdcBlock	1	1	3	24	32x32	7.0K	6.19M
Stage 1	1	2	3	24	32x32	12.7K	8.39M
Stage 2	2	1	3	36	16x16	39.1K	7.24M
	1	2	3	36	16x16		
Stage 3	2	1	3	72	8x8	173.4K	10.40M
	1	3	3	72	8x8		
Stage 4	1	3	3	96	8x8	198.8K	10.54M
Stage 5	2	1	3	150	4x4	313.8K	4.85M
	1	2	3	150	4x4		
Stage 6	1	1	3	300	4x4	247.2K	3.86M
Avg Pool**	2	1		300	1x1	0	0.02M
FC		1		10	1x1	48.6K	0.72M
Complexity***	No. of Weights: 1.04 M				55.12 M Flops		

* G-Conv stands for group convolution with 3x3 kernel and $g = 3$.
** The kernel size of this average pooling is 4x4.
***The complexity is evaluated on images of 10 classes.

SdcNet-S has the simplest structure among the three networks. It is designed to target a wide range of users. Of the seventeen SdcBlocks used in this network, fourteen are the basic SdcBlocks with *stride* = 1 shown in Fig. 1. Among the hyper-parameters, the most important ones are determined based on the principles stated as follows.

1). Group number g. The group number $g = 3$ is carefully chosen and kept identical throughout the network for the sake of simplicity. In the early processing, $g = 3$ matches the number of the input color channels, corresponding to the three kinds of monochromic information. In the feature extraction part, it is also appropriate to use 1×1 group convolutions with a fixed $g = 3$ in the SdcBloks. It should be noted that as the correlation among the data channels gets stronger stage by stage, the number of channels in each group also increases progressively with the fixed group number $g = 3$. Hence, there is always a sufficient number of channels involved in each convolution in different stages to satisfy the processing quality.

2). The number of output channels of each SdcBlock. The initial group convolution with *stride* = 1 generates 36 feature channels, 12 from each of the three input color channels. The 12 channels are adequate to accommodate basic monochromic feature elements. The number of output channels increases from 36 to 300 progressively, as shown in Table 1, meanwhile the size of each channel is decreasing. In other words, the number of channels containing high-level features for decision making is 300. In this network, this number is considered to be large enough to accommodate various features critical to produce a result with acceptable accuracy.

3). The hyper-parameter *stride* of each SdcBlock. Since the size of each channel needs to be reduced from 32×32 to 4×4 in the feature extraction part, as shown in Table 1, 3 SdcBlock-S2s with *stride* = 2 are used in Stage 2, Stage 3 and Stage 5 to downsize the feature maps progressively. There are at least 3 SdcBlocks with *stride* = 1 between two SdcBlock-S2, in order to minimize the risk of information loss.

4). Expansion number E. In this network, $E = 6$ is used in all the SdcBlocks, except the first one where $E = 1$, so that there are enough channels to accommodate various intermediate feature data during the process of SdcBlock.

5). The combinations of the data from different filtering operations. In all the SdcBlocks with *stride* = 1, the convolved data are added to the input data in order to enhance the features and generate new feature maps. In all the SdcBlocks with *stride* = 2, the data of different orders are concatenated for feature-reusing.

6). Data conversion in the decision part. In order to facilitate the process in the decision making part, an average pooling layer is placed before the fully connected layer to compress the 4×4 feature elements in each feature map to one value.

SdcNet-S has the simplest network configuration among the three SdcNets. It uses a small computation amount of 55.12M Flops and 1.03M weights. However, a good processing quality of the network can be predicted since the hyper-parameters in this network are determined to achieve appropriate signal filtering.

3.2.2. SdcNet-M

The detailed configuration of SdcNet-M is presented in Table 2. This network is an updated version of SdcNet-S, aiming

at improving the processing quality without an excessively increase in computation volume. The updates are found in the first SdcBlock of the network and the first two stages of the feature extraction part, namely Stages 1 and 2. The rest of the network is identical to that in SdcNet-S. The updates in designing SdcNet-M are made in the following aspects, with respect to SdcNet-S.

1). As the quality of the feature extraction is related to the number of feature maps produced in each stage, SdcNet-M has a relatively larger number of output channels in the early stages, indicated by the bold number in Table 2, compared to those in SdcNet-S, to accommodate more features.

2). In order to effectively use the larger number of the data channels, the hyperparameter g is not uniform in the network. To be specific, g is chosen to be 12 in the first SdcBlock, 6 in Stage 1, and 3 in the rest of the network. Increasing the number of channel groups in a convolution layer implies fewer channels per group and, in consequence, less computation required to perform the convolution, which partially cancels the rise in the computation cost resulting from the increased number of the data channels. Moreover, a large value of g allows more flexibility in channel grouping.

3). Data channel grouping is particularly explored in the design of this network. In the first filtering stages, the input data channels can be grouped to form monochromic groups, or specifically-weighted polychromatic groups, and they are then convolved to produce varieties of features with different chromic emphases. This fashion of channel grouping is referred to as Specific Color Reordering (SCR). It can be seen as a specific data channel indexing, in contrast to the channel indexing by shuffling used in the 6 stages of the feature extraction part of the network.

Understandably, channel grouping can be based on various characters of the data channels. For example, one group can consist of the data produced by the filtering operations of the same order and another group of different orders with a specific proportion.

In this network, the very first convolution layer produces 36 data channels, 12 from each original RGB channel. SCR is performed in the first SdcBlock. The 36 channels are divided into 12 groups, 3 per group, by the first 1×1 group convolution. After concatenating the results of the 2 successive depthwise convolutions, the number of the data channels is doubled. Then the 72 channels are divided into 12 groups, 6 per group, by the special color Reordering layer, *i.e*. the SCR layer, before the second 1×1 group convolution layer in the SdcBlock shown in Fig. 1. Of the 12 groups, 6 are monochromic groups, *i.e*., the channels in each group originated from the same color channel. The other 6 are full-color groups, *i.e*., each group consisting of 3 data channels originated from the RGB input. Meanwhile, each of the 12 groups contains both the first and the second filtering results.

The Specific Color Reordering (SCR) method can help to make good use of the early features by combining, in a rational manner, data channels in order to generate useful varieties of feature information. It should be noted that, if the convolved data channels are arranged by SCR and they are then combined with the input data by addition, the input data should also be rearranged in the same manner to have the same data sequence as the convolved data.

SdcNet-M is designed to obtain a better processing quality, with respect to that of SdcNet-S, by improving the quality of the basic features in the early stages. The total computation amount of the network is 73.41M Flops with 1.11M weights, which is still a modest amount compared to those reported for similar tasks.

3.2.3. SdcNet-L

The configuration of SdcNet-L is shown in Table 3. Based on SdcNet-S, SdcNet-L is designed to obtain a better processing quality by applying more convolution kernels in Stages 2 to 6 and thus generating more data channels. As shown in Table 3, the number of output channels of Stage 6 is 600, twice of that in SdcNet-S, providing the decision part with more precise feature information.

Two measures are taken to compensate for the increase of computation volume caused by the additional channels. Firstly, the number of channels in the initial convolution or Stage 1 is kept the same as that in SdcNet-S before downsizing the map dimension. The increasing of the channel number starts from Stage 2, where the feature map size is reduced from 32×32 to 16×16. The second measure is to increase the group number g in all the stages, except the initial group convolution, from $g = 3$ to $g = 4$, reducing the number of parameters in order to decrease the computation volume. It should be noted that this change does not reduce the numbers of channels per convolution group as the total numbers of the channels are increased, which avoid the degradation of the processing quality.

In this design, SdcNet-L is built to achieve the best processing quality among the three SdcNets by a larger number of the channels. It requires 103.3M Flops and uses 2.53M weights. Compared with the reported CNNs for similar tasks, the computation volume is still a modest amount.

3.3. Two Versions of SdcNet for ImageNet Datasets

3.3.1. SdcNet-IS

SdcNet-IS is another version of SdcNet, developed to test the applicability of SdcNet for images of large format, such as those from ImageNet.

Comparing the image samples from CIFAR and those from ImageNet, one can see that, though the latter have a higher resolution, the object in each image, no matter it is from CIFAR or ImageNet, occupies a dominant portion of the space. Hence, image downsizing can be used to reduce the image format without serious concern of missing the object. However, it should be applied appropriately to minimize the loss of image details.

The configuration of SdcNet-IS is shown in Table 4. Like the three SdcNets described in previous sub-sections, it consists mainly of SdcBlocks, and some of its details are, nevertheless,

Table 2. Details of SdcNet-M Configuration
Input image: sized 32x32 pixels and having 3 channels.

	Stride	Repeat Times	Group No. g	No. of Output Channels	Size of Output Channels	No. of Weights	Computation Complexity(Flops)
G-Conv*	1	1	3	36	32x32	3.2K	2.91M
SdcBlock	1	1	**12**	**36**	32x32	9.1K	7.96M
Stage 1	1	2	**6**	**36**	32x32	21.3K	12.80M
Stage 2	2	1	3	**66**	16x16	91.4K	19.86M
	1	2	3	**66**	16x16		
Stage 3	2	1	3	72	8x8	173.4K	10.40M
	1	3	3	72	8x8		
Stage 4	1	3	3	96	8x8	198.8K	10.54M
Stage 5	2	1	3	150	4x4	313.8K	4.85M
	1	2	3	150	4x4		
Stage 6	1	1	3	300	4x4	247.2K	3.86M
Avg Pool**	2	1		300	1x1	0	0.02M
FC		1		10	1x1	48.6K	0.72M
Complexity***	No. of Weights: 1.11 M				73.41 M Flops		

* G-Conv stands for group convolution with 3x3 kernel and $g = 3$.
** The kernel size of this average pooling is 4x4.
***The complexity is evaluated on images of 10 classes.

adjusted to handle the input images of 224x224. Compared to SdcNet-S, SdcNet-IS has more convolutions with *stride* = 2 applied in different layers to reduce the image size, while extracting the features of different levels. Also, more filtering kernels are used in the convolution layers to have a finer classification.

3.3.2. SdcNet-IM

SdcNet-IM is also a version of SdcNet for images of format ImageNet. Its configuration, shown in Table 5, is similar to that of SdcNet-IS, but the number of output channels in most convolution layers is extended in order to process more varieties of feature data. Hence, a better classification result is expected.

4. Performance Evaluations of SdcNets

In this section, the performance evaluation of different versions of SdcNet for object recognition is presented. The dataset, training/testing information and an ablation study are found in Subsections 4.1, 4.2 and 4.3, respectively. The comparison of the test results of SdcNets with the CNNs performing similar tasks is found in Subsections 4.4.

4.1. Datasets

Datasets CIFAR-10 and CIFAR-100 (Krizhevsky and Hinton, 2009) have been used to evaluate the performance of the 3 SdcNets, SdcNet-S, SdcNet-M and SdcNet-L. CIFAR-10 consists of 60000 color images of 32x32 pixels in 10 classes, 6000 per class. The testing pool of 10000 images is built by randomly selecting 1000 from each class. The remaining 50000 images are for training. CIFAR-100 has the same number of images as CIFAR-10, but the number of classes is 100. Thus, the number of images per class is 10 times smaller than that in CIFAR-10.

ImageNet-1K (Deng et al., 2009) dataset has been used to assess the applicability of SdcNet-IS and SdcNet-IM for images of large format. ImageNet-1K has 1.2 million samples in its training pool and 50,000 samples in its testing pool, and they are in1000 classes. All these images are resized to 224 x 224 pixels to train and test the 2 SdcNets.

4.2. Training Details

The training elements are as follows.

- Training epochs and batch size. As the computation volumes in the SdcNets are modestly ranged from 55M to 202M Flops, a relatively small number of epochs to complete the training procedure can be expected. The SdcNets have been trained for 300 epochs with a batch size of 128 for CIFAR dataset, and for 90 epochs with a batch size of 384 for ImageNet dataset.

- Optimizer. Stochastic gradient descent (SGD) (LeCun et al., 1989) has been chosen to minimize the loss in the experiments. Moreover, in order to increase the convergence rate, Nesterov momentum (Nesterov, 1983) with a momentum weight of 0.9 and a weight decay of 0.0001 has also been adopted.

- Learning rate. Cosine-shape decreasing method (Loshchilov and Hutter, 2016) has been used to make the learning rate decrease, starting from 0.1, during the training process.

- Loss function. Cross entropy loss function (LeCun et al., 2015) has been used to calculate the loss in this work.

- Training data augmentation. It has been done by padding the original training images with four zero pixels per side,

Table 3. Details of SdcNet-L Configuration
Input image: sized 32x32 pixels and having 3 channels.

	Stride	Repeat Times	Group No. g	No. of Output Channels	Size of Output Channels	No. of Weights	Computation Complexity(Flops)
G-Conv*	1	1	3	36	32x32	3.2K	2.91M
SdcBlock	1	1	**4**	24	32x32	7.0K	6.19M
Stage 1	1	2	**4**	**36**	32x32	23.1K	14.90M
Stage 2	2 1	1 2	**4** **4**	**72** **72**	16x16 16x16	91.1K	18.79M
Stage 3	2 1	1 3	**4** **4**	**96** **96**	8x8 8x8	235.9K	14.16M
Stage 4	1	3	**4**	**144**	8x8	362.3K	18.16M
Stage 5	2 1	1 2	**4** **4**	**300** **300**	4x4 4x4	898.0K	14M
Stage 6	1	1	**4**	**600**	4x4	719.4K	11.32M
Avg Pool**	2	1		**600**	1x1	0	0.04M
FC		1		10	1x1	187.2K	2.89M
Complexity***	No. of Weights: 2.53 M				103.3 M Flops		

* G-Conv stands for group convolution with 3x3 kernel and $g = 3$.
** The kernel size of this average pooling is 4x4.
***The complexity is evaluated on images of 10 classes.

cropping the padded images randomly and flipping half of the cropped images horizontally. All the cropped samples have been used as training samples.

- Initialization of the weights. The weights of the network have been initialized in such a way that the standard deviation between inputs and outputs is the same in each layer (He et al., 2015).

The characteristics of training loss versus training epochs, obtained with the three SdcNets for CIFAR-format images are shown in Fig. 4. It is illustrated that all the 3 networks have approached their steady states, after the training of 256 epochs, with the loss of around 0.002, demonstrating a fast convergence and small residue error of the network. The characteristics of validation error rate of the three SdcNets are illustrated in Fig. 5, demonstrating that the validation error rates follow the trend of the loss, and there is no underfitting observed.

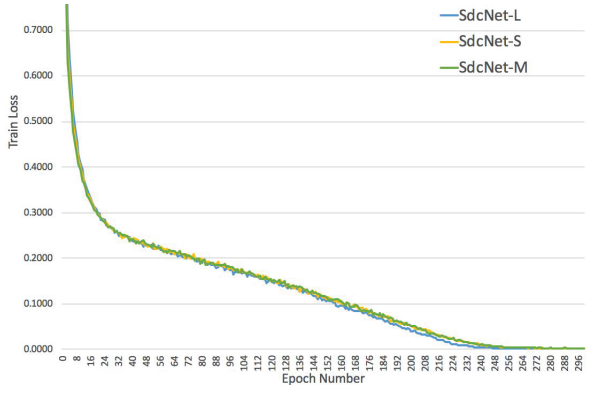

Fig. 4. Characteristics of training loss versus training epochs with CIFAR dataset.

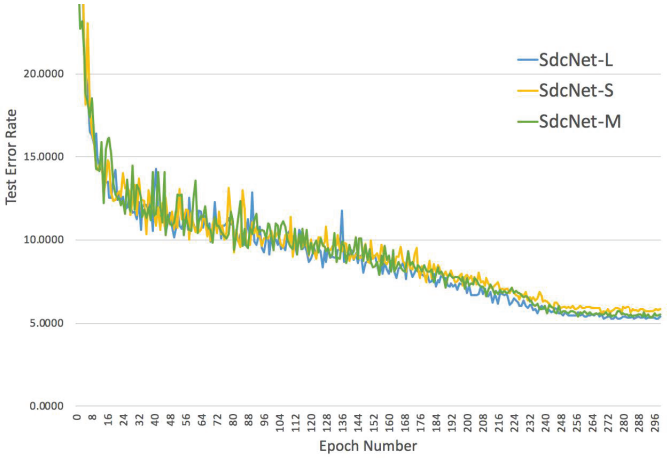

Fig. 5. Characteristics of validation error rate versus training epochs with CIFAR dataset.

4.3. Ablation study

The ablation study has been conducted to assess the performance of SdcNets from different aspects. Various tests have been done to demonstrate how the structure of SdcBlocks and hyperparameters concerning data handling can impact the quality of the processing and computation complexity.

SdcBlock distinguishes itself mainly with the two Successive 3x3 depthwise convolutions (Sdc). As shown in Fig. 1, these convolutions allow to generate, from the same set of data X, 2 sets of filtered data X' and X'' representing different kinds of feature data.

The first part in this ablation study is to test the effectiveness in signal filtering of the Sdc structure. To this end, we have created a block, referred to as dcBlock, by removing the first 3×3 depthwise convolution of SdcBlock. Evidently dcBlock

Table 4. Details of SdcNet-IS Configuration
Input image: sized 224x224 pixels and having 3 channels.

	Stride	Repeat Times	Expansion	Group No. g	No. of Output Channels	Size of Output Channels	No. of Weights	Computation Complexity(Flops)
G-Conv*	2	1		3	24	112x112	0.70K	8.43M
Stage 1	2	1	1	3	36	56x56	0.98K	5.04M
Stage 2	2	1	3	3	54	28x28	30.7K	25.15M
	1	2	3	3	54	28x28		
Stage 3	2	1	3	3	108	14x14	270.6K	52.11M
	1	6	3	3	108	14x14		
Stage 4	2	1	3	3	196	7x7	510.8K	26.05M
	1	3	3	3	196	7x7		
Conv	1	1		3	1024	7x7	223.2K	10.89M
Avg Pool**	1	1			1024	1x1	0	0.05M
FC		1			1000	1x1	1025.0K	1.02M
Complexity	No. of Weights: 2.06 M					128.69 M Flops		

* G-Conv stands for group convolution with 3x3 kernel and $g = 3$.
** The kernel size of this average pooling is 7x7.

Table 5. Details of SdcNet-IM Configuration
Input image: sized 224x224 pixels and having 3 channels.

	Stride	Repeat Times	Expansion	Group No. g	No. of Output Channels	Size of Output Channels	No. of Weights	Computation Complexity(Flops)
G-Conv*	2	1		3	24	112x112	0.70K	8.43M
Stage 1	2	1	1	3	36	56x56	0.98K	5.04M
Stage 2	2	1	3	3	72	28x28	48.3K	38.36M
	1	2	3	3	72	28x28		
Stage 3	2	1	3	3	144	14x14	458.0K	89.29M
	1	6	3	3	144	14x14		
Stage 4	2	1	3	3	288	7x7	883.2K	45.41M
	1	3	3	3	288	7x7		
Conv	1	1		3	1024	7x7	296.9K	14.50M
Avg Pool**	1	1			1024	1x1	0	0.05M
FC		1			1000	1x1	1025.0K	1.02M
Complexity	No. of Weights: 2.71 M					202.05 M Flops		

* G-Conv stands for group convolution with 3x3 kernel and $g = 3$.
** The kernel size of this average pooling is 7x7.

does not have the character of Successive depthwise convolutions (Sdc), as $X' = X$ and Xc is the combination of X and X''. Then, we have replaced SdcBlocks in the networks of SdcNet-S, SdcNet-M, and SdcNet-L by dcBlocks and tested these networks using CIFAR dataset. The results are presented in Table 6, in comparison to those of SdcNets with SdcBlocks. One can see that SdcNets with SdcBlocks perform better than those with dcBlocks. By means of the successive depthwise convolutions, more varieties of feature data can be extracted from the same set of inputs, which leads to a visible decrease of the error rate with only very small increase in computation volume.

Table 6. SdcNets With SdcBlocks and dcBlocks

Network config	Block	Error Rate	Weights	Flops
SdcNet-S*	SdcBlock	5.60%	1.04M	55.12M
	dcBlock	6.51%	0.96M	49.05M
SdcNet-M**	SdcBlock	5.41%	1.11M	73.41M
	dcBlock	6.23%	1.02M	64.08M
SdcNet-L***	SdcBlock	5.24%	2.53M	103.3M
	dcBlock	5.78%	2.39M	94.31M

* Configuration details are shown in Table 1.
** Configuration details are shown in Table 2.
*** Configuration details are shown in Table 3.

Another element in SdcBlock is the channel reordering. It is placed after the 2 successive depthwise convolutions. This reordering is to re-index the sequence of the channels X' and X'', produced by the 2 convolution layers, to prepare the input data groups for the second 1x1 group convolution. By doing so, each of the input group can have data from both X' and X'' containing information of different filtering orders. Table 7 illustrates the test results of the SdcNets for CIFAR dataset, ob-

tained with and without the reordering in the SdcBlocks. The networks with the channel reordering outperform, with a significant difference, those without reordering. It confirms that it is important, not only to generate the data of different filtering orders by means of successive depthwise convolutions, but also to include them in the same group in the succeeding data processing, and the channel reordering is an effective and simple way to do it.

Table 7. SdcNets With/Without Reordering

Model	Config	Error Rate
SdcNet-S	With Reordering	5.6%
	Without Reordering	7.31%
SdcNet-M	With Reordering	5.41%
	Without Reordering	6.93%
SdcNet-L	With Reordering	5.24%
	Without Reordering	6.82%

In a SdcBlock, 1x1 group convolutions (G-conv) are used to organize the data channels before and after the 2 successive depthwise convolutions. The input channels of the 1x1 G-conv are divided into g groups. For a given number of the input channels, the smaller g, the more channels are involved in the computation of each group convolution, i.e., more data being processed together to generate each output channel, which is, however, at the expense of more learnable parameters and more calculations. Hence, one needs to find a balance point of the value of g. Based on SdcNet-S, we have created 3 variations, namely SdcNet-S-G1, SdcNet-S-G2 and SdcNet-S-G3, and they differ from one another only on g. The test results of SdcNet-S and its variations, with CIFAR dataset, are presented in Table 8. One can observe that $g = 3$ is suitable to obtain good processing results at a relatively low computation cost.

Table 8. SdcNet-S with Different Group No. g
Input image: sized 32x32 pixels and having 3 channels.

g	SdcNet-S	SdcNet-S-G1	SdcNet-S-G2	SdcNet-S-G3
G-Conv*	3	3	3	3
SdcBlock	3	6	4	12
Stage 1	3	6	4	6
Stage 2	3	4	3	3
Stage 3	3	4	3	3
Stage 4	3	3	2	3
Stage 5	3	3	2	3
Stage 6	3	3	2	3
Avg Pool**				
FC				
Weights	1.05M	1.01M	1.38M	1.04M
Flops	55.12M	47.92M	62.53M	50.77M
Error Rate	5.6%	6.32%	5.83%	5.88%

Note: Parameters of *stride*, Repeat Times, Number of Output Channels are the same as those presented in Table 1.

The expansion parameter E determines, in a SdcBlock, the number of the output channels of the first 1x1 group convolution, and one can use it to adjust the capacity of feature generation/accommodation. A larger E implies more capacity to generate and to accommodate the feature data, and may lead to better classification results, but demanding more computation. The data presented in Table 9 are the test results of the 4 versions of SdcNets configured for the inputs from ImageNet. Comparing the results given by SdcNet-IS and SdcNet-IS-E6, one can find that the error rate is reduced from 38.4% to 36.2% by increasing E from 3 to 6, while increasing significantly the volume of computation, measure by the number of Flops, by almost 80%. Observing the results given by the two SdcNet-IM versions that differ each other only in E, one can get the same conclusion.

Table 9. Different SdcNet Architectures for ImageNet

	SdcNet-IS-E3		SdcNet-IS-E6		SdcNet-IM-E3		SdcNet-IM-E6	
	E	N_O*	E	N_O*	E	N_O*	E	N_O*
G-Conv		24		24		24		24
Stage 1	1	36	1	36	1	36	1	36
Stage 2	3	54	6	54	3	72	6	72
Stage 3	3	108	6	108	3	144	6	144
Stage 4	3	196	6	196	3	288	6	288
Conv		1024		1024		1024		1024
Avg Pool		1024		1024		1024		1024
FC		1000		1000		1000		1000
Weights	2.06M		2.87M		2.71M		4.10M	
Flops	128.69M		231.60M		202.05M		374.65M	
ErrorRate	38.4%		36.2%		36.3%		34.5%	

Note: Parameters of *stride*, Repeat Times of SdcNet-IS-E3 and SdcNet-IS-E6 are found in Table 4, and those of SdcNet-IM-E3 and SdcNet-IM-E6 in Table 5.
*: Number of Output Channels

In conclusion, by testing the networks with variations in structure and in hyperparameters, one can observe the effects produced by these elements. The test results confirm that the successive depthwise convolutions play an important role in different stages of processing, and it is equally important to handle the data produced by the 2 convolutions in an optimized manner to make good use of the data. It is also shown that the two hyperparameters, group number g and expansion parameter E, can be used to find a good balance of processing quality and computation.

4.4. Testing Results with Samples from CIFAR

The performance metrics include the processing quality measured by the classification error rate(ER), and the computation volume, in terms of the number of Flops and the number of weights. As mentioned previously, for the test with the dataset of CIFAR-10 or CIFAR-100, the number of images used in the test is 10000. Each test has been repeated 3 times. In order to assess the computation efficiency of SdcBlock and to evaluate the performance of the three SdcNets, the test results are compared with those reported recently and having similar computation complexity, namely VGG-pruned and ResNet-pruned networks (Li et al., 2016; Liu et al., 2017; He et al., 2017).

The test results are found in Table 10. As the error rates of SdcNet-S, SdcNet-M and SdcNet-L are the average values obtained from multiple tests, they are presented with the standard

Table 10. Comparison of classification error rate and computation volume

Model	FLOPs	No. of Weights	*ER* (C10)*	*ER* (C10) STD**	*ER* (C100)***
VGG-16-pruned (Li et al., 2016)	206M	5.40M	6.60%	-	25.28%
VGG-19-pruned (Liu et al., 2017)	195M	2.30M	6.20%	-	-
VGG-19-pruned (Liu et al., 2017)	250M	5.00M	-	-	26.20%
ResNet-56-pruned (He et al., 2017)	62M	-	8.20%	-	-
ResNet-56-pruned (Li et al., 2016)	90M	0.73M	6.94%	-	-
ResNet-110-pruned (Li et al., 2016)	213M	1.68M	6.45%	-	-
ResNet-164-B-pruned (Liu et al., 2017)	124M	1.21M	5.27%	-	25.28%
SdcNet-S	**55.12M**	**1.04M**	**5.60%**	**0.026**	**25.01%**
SdcNet-M	**73.41M**	**1.11M**	**5.41%**	**0.01**	**24.28%**
SdcNet-L	**103.3M**	**2.53M**	**5.24%**	**0**	**23.12%**

* The Error Rates(*ER*) for CIFAR-10 dataset.
** Standard Deviation of Error Rates(*ER*) for CIFAR-10 dataset.
*** The Error Rates(*ER*) for CIFAR-100 dataset.

Table 11. Inference performance on a x86 CPU

Model	Test Image Size	No. of Images in Test	Total Inference Time	Inference Time Per Image	Peak Mem Usage
SdcNet-S	3x32x32	100	283ms	2.83ms	38MB
SdcNet-M			383ms	3.83ms	67MB
SdcNet-L			532ms	5.32ms	80MB

deviations. It is shown that SdcNet require significantly less computation, with respect to the pruned networks, for a similar or even better processing quality. In particular, in the case of the test with CIFAR-10, one can find the evaluation details given as follows.

- SdcNet-S. The total computation volume of this network is 55.12 M Flops, the smallest among those listed in the table, but it has, nevertheless, achieved a low error rate of 5.60%, among the best ones in the table.
- SdcNet-L. It has achieved the best processing quality, i.e. the error rate of 5.24%, at a computation cost of 103 M Flops, with respect to 5.27% given by ResNet-164-B-pruned using 124 M Flops.
- SdcNet-M. It achieves a low error rate of 5.41% using only 73.41 M Flops, a balance between the processing quality and the computation complexity.

The inference performance of a CNN is related to the type of processors used for computation. In case of GPU, as the modes of data transfer can make a significant difference in computation time, a GPU designed to suit a particular kind of network may place other kinds at a disadvantage. To assess the inference performance of the SdcNets for CIFAR data, the test is conducted on an Intel i7 8700K CPU running at 4.6 GHz, with Python 3.7.7 and PyTorch v1.2.0. The result, presented in Table 11, demonstrates that the computation of SdcNet can be performed with one CPU core and a memory usage of less than 100 MB, and the time required to complete an object recognition of an image from CIFAR dataset is not more than 5.32 mS.

Table 12. Comparison of classification error rate and computation volume with ImageNet

Model	FLOPs	No. of Weights	Top-1 *ER*
IGCV3-D (0.7) (Sun et al., 2018)	210M	2.8M	31.5%
MobileNetV2 (0.7) (Sun et al., 2018)	210M	2.8M	33.5%
Xception (Chollet, 2016)	145M	-	34.1%
MobileNet-0.5 (Xie, 2018)	149M	1.3M	36.3%
IGCV2-0.5 (Xie, 2018)	156M	1.3M	34.5%
DenseNet (Huang et al., 2017)	142M	-	45.2%
ShuffleNetV1 (Zhang et al., 2018)	140M	-	32.6%
SdcNet-IM	**202M**	**2.71M**	**36.3%**
SdcNet-IS	**128M**	**2.06M**	**38.4%**

Table 13. Result comparison - BTD – Dice scores and computation volumes

Systems	*Dice*			No. of Weights
	ET	WT	TC	
(Chen et al., 2019)	0.703	0.891	0.782	6.3*M*
(Zhou et al., 2020)	0.730	0.894	0.816	> 1*M*
(Zhou et al., 2021)	0.708	0.871	0.783	> 1*M*
(Aboelenein et al., 2020)	0.745	0.865	0.808	> 1*M*
SdcNet BTD	0.775	0.887	0.769	0.083M

4.5. Testing Results with Samples from IMAGENET

Two versions of SdcNet, namely SdcNet-IS and SdcNet-IM, have been trained with the image samples of ImageNet, and tested with objects of 1000 classes. The test results are presented in Table 12. They are comparable with those given by the networks of similar computation complexity. It demonstrates that, though the basic scheme of SdcNets has been proposed to process images of small format, it can be extended for applications to images such as those from ImageNet.

5. SdcBlocks and SdcNet for Detection

SdcBlock has initially been designed for object recognition. However, as it is able to extract effectively varieties of image features, it can also be used for other tasks. In this section, as

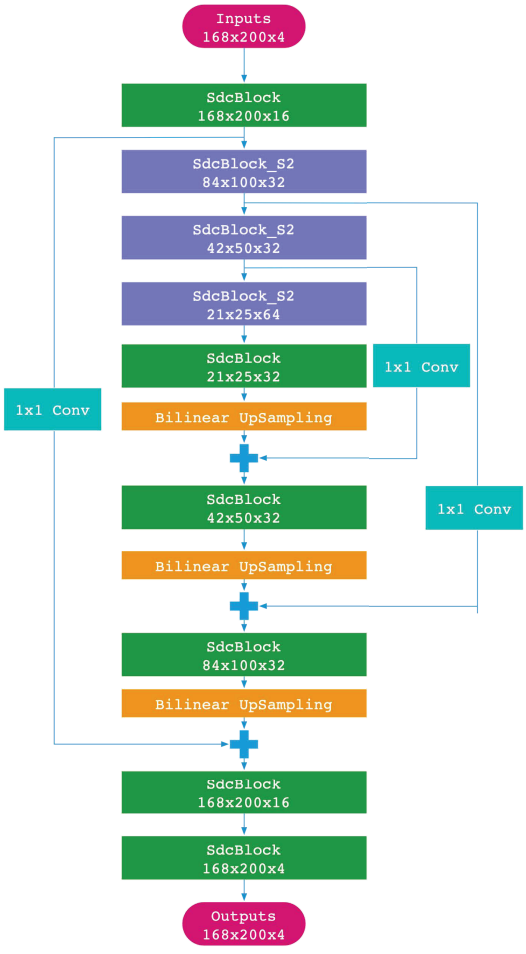

Fig. 6. SdcNet for brain tumor detection (BTD). In all the SdcBlocks, $E = 1$ and $g = 1$, except the very first SdcBlock where $g = 4$ is applied to the first 1×1 convolution and $g = 1$ to the second one.

a design example of SdcBlock and SdcNet for detection tasks, a special SdcNet for brain tumor detection (BTD) is presented and so are the test results.

5.1. SdcNet for Brain Tumor Detection

The input data of the SdcNet for brain tumor detection (BTD) are 3D MRI brain images. Each patient case consists of four MRI modalities, namely Flair, T1, T1c and T2. The objective of BTD is to determine, with voxel-wise precision, the locations of enhancing tumor (ET), tumor core (TC), and whole tumor (WT).

As a 3D image can be sliced into 2D slices, the brain tumor detection is often performed in 2D CNNs. The structure of the SdcNet for BTD is illustrated in Fig. 6. Like the other SdcNets presented in this paper, in this SdcNet, the convolutions are performed by cascaded SdcBlocks. However, as it needs to deliver the result of multi-class detection with voxel-wise precision, upsampling and skip connection are used, which is commonly seen in SSD or Unet, so that detailed image feature information produced by the first 3 SdcBlocks is included, step by step, in the convolutions in the last 3 SdcBlocks, as shown in Fig. 6.

5.2. Performance Evaluation of the SdcNet for BTD

The SdcNet for BTD has been trained and tested with BRATS2018 dataset. As BRATS2018 does not disclose the ground truth of the 66 test cases, the test results have been assessed by CBICA Image Processing Portal (Menze et al., 2014; Bakas et al., 2017, 2018). The processing quality of BTD systems is measured mainly by Dice score (*Dice*), defined as

$$Dice(P_1, T_1) = \frac{P_1 \Lambda T_1}{(P_1 + T_1)/2} \tag{5}$$

where P_0 and P_1 are the predicted results, indicating the number of voxels in the tumor-free regions and that in the tumor regions, respectively, whereas T_0 and T_1 are those in the ground truth. If all the predicted tumor voxels are completely overlapped with those in the ground truth, $Dice = 1$. There are 3 Dice scores for enhancing tumor (ET), tumor core (TC), and whole tumor (WT), respectively, and the ET Dice score is considered the most important.

The test results are found in Table 13, in comparison with those of 4 Unet-based CNNs that requiring moderate level of computation and are reported in reputed research journals in recent years. The SdcNet for BTD delivers its detection results significantly better than the other 4 systems, which is, moreover, achieved with a very small fraction of computation resources needed by others. The computation efficiency of this SdcNet has been demonstrated.

The data of each patient case are of four 3D images and each has 155 2D slices. Under the same computing hardware/software conditions specified in Subsection 4.4, the inference time is 2.64 S the peak memory usage is 352 MB. The number of Flops required to complete the test of each patient case is 1562.39M.

6. Conclusion

Aiming at achieving a good processing quality at the lowest computation cost, SdcNet, a simple CNN architecture for object recognition, has been proposed in this paper. In its basic block, called SdcBlock, successive depthwise convolutions (Sdc) are performed to extract the maximum amount of high-density feature information from each data channel. To optimize the functionality of these convolutions, a particular pre-and-post-convolution data control method has been applied. The pre-convolution data control is to organize the input data channels of the module in such a way that the depthwise convolutions can be performed with a single channel, or multiple channels, depending on the nature of the data. The post-convolution data control is to handle the data of different filtering orders, i.e., feature data generated by the 2 convolutions and the data transmitted from the input of the module, to enhance the quality of the convolution results.

SdcNet is mainly composed of cascaded SdcBlocks. Its structure can be adjusted easily so that each module can be tuned to suit its input signals in order to optimize the processing quality of the entire network. Six versions of SdcNets have been developed. Three of them aim at processing input images of CIFAR format. The simulation results have confirmed

that, to achieve a given recognition quality, these three SdcNets require a significantly smaller computation volume than many pruned networks. They can provide a good processing quality and be implemented in a computation-restricted environment. Two other versions have been developed to test the applicability of the scheme of SdcNets for images of ImageNet. The test results have illustrated that, with an adequate adjustment, SdcNets can be used to perform object recognition tasks with images of large format and deliver a processing quality in the same level as those of similar computation complexity. The other version of SdcNet is constructed for brain tumor detection. Its test results confirm not only the applicability of SdcBlock/SdcNet for the detection but also its superior computation efficiency, *i.e.*, significantly better processing quality achieved with much less computation.

Acknowledgments

This work was supported in part by Compute Canada and in part by the Natural Sciences and Engineering Research Council (NSERC) of Canada.

The authors would like to thank Yanming Sun for his providing them with useful information about available brain tumor data resources and for his help in the experiments.

References

Aboelenein, N.M., Songhao, P., Koubaa, A., Noor, A., Afifi, A., 2020. Httu-net: hybrid two track u-net for automatic brain tumor segmentation. IEEE Access 8, 101406–101415.

Bakas, S., Akbari, H., Sotiras, A., Bilello, M., Rozycki, M., Kirby, J.S., Freymann, J.B., Farahani, K., Davatzikos, C., 2017. Advancing the cancer genome atlas glioma mri collections with expert segmentation labels and radiomic features. Scientific data 4, 1–13.

Bakas, S., Reyes, M., Jakab, A., Bauer, S., Rempfler, M., Crimi, A., Shinohara, R.T., Berger, C., Ha, S.M., Rozycki, M., et al., 2018. Identifying the best machine learning algorithms for brain tumor segmentation, progression assessment, and overall survival prediction in the brats challenge. arXiv preprint arXiv:1811.02629 .

Brunelli, R., Poggio, T., 1993. Face recognition: Features versus templates. IEEE transactions on pattern analysis and machine intelligence 15, 1042–1052.

Chapelle, O., Haffner, P., Vapnik, V.N., 1999. Support vector machines for histogram-based image classification. IEEE transactions on Neural Networks 10, 1055–1064.

Chen, S., Ding, C., Liu, M., 2019. Dual-force convolutional neural networks for accurate brain tumor segmentation. Pattern Recognition 88, 90–100.

Chollet, F., 2016. Xception: Deep learning with depthwise separable convolutions. arXiv preprint arXiv:1610.02357 .

Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L., 2009. ImageNet: A Large-Scale Hierarchical Image Database, in: Proc. of the IEEE Conference on Computer Vision and Pattern Recognition.

Hampapur, A., Brown, L., Connell, J., Ekin, A., Haas, N., Lu, M., Merkl, H., Pankanti, S., 2005. Smart video surveillance: exploring the concept of multiscale spatiotemporal tracking. IEEE Signal Processing Magazine 22, 38–51. doi:10.1109/MSP.2005.1406476.

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

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

He, Y., Zhang, X., Sun, J., 2017. Channel pruning for accelerating very deep neural networks, in: Proc. of IEEE International Conference on Computer Vision, p. 6.

Hu, J., Shen, L., Sun, G., 2018. Squeeze-and-excitation networks, in: Proc. of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 7132–7141.

Huang, G., Liu, Z., Weinberger, K.Q., van der Maaten, L., 2017. Densely connected convolutional networks, in: Proc. of the IEEE Conference on Computer Vision and Pattern Recognition, p. 3.

Krizhevsky, A., Hinton, G., 2009. Learning multiple layers of features from tiny images .

LeCun, Y., Bengio, Y., Hinton, G., 2015. Deep learning. Nature 521, 436.

LeCun, Y., Boser, B., Denker, J.S., Henderson, D., Howard, R.E., Hubbard, W., Jackel, L.D., 1989. Backpropagation applied to handwritten zip code recognition. Neural computation 1, 541–551.

LeCun, Y., Bottou, L., Bengio, Y., Haffner, P., et al., 1998. Gradient-based learning applied to document recognition. Proceedings of the IEEE 86, 2278–2324.

Li, H., Kadav, A., Durdanovic, I., 2016. Pruning filters for efficient convnets. arXiv preprint arXiv:1608.08710 .

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

Loshchilov, I., Hutter, F., 2016. Sgdr: stochastic gradient descent with restarts. Learning 10, 3.

Lowe, D.G., 2004. Distinctive image features from scale-invariant keypoints. International journal of computer vision 60, 91–110.

Ma, Y., Wang, C., 2018. Sdcnet: A computation-efficient cnn for object recognition, in: Proc. of 2018 IEEE 23rd International Conference on Digital Signal Processing (DSP), pp. 1–5. doi:10.1109/ICDSP.2018.8631567.

Menze, B.H., Jakab, A., Bauer, S., Kalpathy-Cramer, J., Farahani, K., Kirby, J., Burren, Y., Porz, N., Slotboom, J., Wiest, R., et al., 2014. The multimodal brain tumor image segmentation benchmark (brats). IEEE transactions on medical imaging 34, 1993–2024.

Nesterov, Y., 1983. A method for unconstrained convex minimization problem with the rate of convergence o (1/kˆ 2), in: Doklady AN USSR, pp. 543–547.

Polyak, A., Wolf, L., 2015. Channel-level acceleration of deep face representations. IEEE Access 3, 2163–2175.

Pontil, M., Verri, A., 1998. Support vector machines for 3d object recognition. IEEE transactions on pattern analysis and machine intelligence 20, 637–646.

Prabhu, A., Varma, G., Namboodiri, A., 2018. Deep expander networks: Efficient deep networks from graph theory, in: Proc. of the European Conference on Computer Vision (ECCV), pp. 20–35.

Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., Chen, L.C., 2018. Mobilenetv2: Inverted residuals and linear bottlenecks, in: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, IEEE. pp. 4510–4520.

Simonyan, K., Zisserman, A., 2014. Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556 .

Sun, K., Li, M., Liu, D., Wang, J., 2018. Igcv3: Interleaved low-rank group convolutions for efficient deep neural networks. arXiv preprint arXiv:1806.00178 .

Sutton, R.N., Hall, E.L., 1972. Texture measures for automatic classification of pulmonary disease. IEEE Transactions on Computers 100, 667–676.

Wang, L., Yu, X., Bourlai, T., Metaxas, D.N., 2019. A coupled encoder–decoder network for joint face detection and landmark localization. Image and Vision Computing 87, 37–46.

Xie, G., 2018. Jingdong wang, ting zhang, jianhuang lai, richang hong, and guo-jun qi. igcv2: Interleaved structured sparse convolutional neural networks, in: Proc. of the IEEE Conference on Computer Vision and Pattern Recognition, CVPR.

Yang, Y., Yang, M., Huang, S., Que, Y., Ding, M., Sun, J., 2017. Multifocus image fusion based on extreme learning machine and human visual system. IEEE access 5, 6989–7000.

Zhang, X., Zhou, X., Lin, M., Sun, J., 2018. Shufflenet: An extremely efficient convolutional neural network for mobile devices, in: Proc. of the IEEE conference on computer vision and pattern recognition, pp. 6848–6856.

Zhou, T., Canu, S., Ruan, S., 2020. Fusion based on attention mechanism and context constraint for multi-modal brain tumor segmentation. Computerized Medical Imaging and Graphics 86, 101811.

Zhou, T., Canu, S., Vera, P., Ruan, S., 2021. Latent correlation representation learning for brain tumor segmentation with missing mri modalities. IEEE Transactions on Image Processing 30, 4263–4274.