Convolution in Convolution for Network in Network

Yanwei Pang, *Senior Member, IEEE*, Manli Sun, Xiaoheng Jiang, and Xuelong Li, *Fellow, IEEE*

***Abstract*— Network in network (NiN) is an effective instance and an important extension of deep convolutional neural network consisting of alternating convolutional layers and pooling layers. Instead of using a linear filter for convolution, NiN utilizes shallow multilayer perceptron (MLP), a nonlinear function, to replace the linear filter. Because of the powerfulness of MLP and 1 × 1 convolutions in spatial domain, NiN has stronger ability of feature representation and hence results in better recognition performance. However, MLP itself consists of fully connected layers that give rise to a large number of parameters. In this paper, we propose to replace dense shallow MLP with sparse shallow MLP. One or more layers of the sparse shallow MLP are sparely connected in the channel dimension or channel–spatial domain. The proposed method is implemented by applying unshared convolution across the channel dimension and applying shared convolution across the spatial dimension in some computational layers. The proposed method is called convolution in convolution (CiC). The experimental results on the CIFAR10 data set, augmented CIFAR10 data set, and CIFAR100 data set demonstrate the effectiveness of the proposed CiC method.**

***Index Terms*— Convolution in convolution (CiC), convolutional neural networks (CNNs), image recognition, network in network (NiN).**

I. Introduction

DEEP convolutional neural networks (CNNs) have achieved state-of-the-art performance in the tasks of image recognition and object detection [3], [9], [31]. CNNs are organized in successive computational layers alternating between convolution and pooling (subsampling). Compared with other types of deep neural networks, CNNs are relatively easy to train with backpropagation mainly because they have a very sparse connectivity in each convolutional layer [2]. In a convolutional layer, linear filters are used for convolution. The main parameters of CNNs are the parameters (i.e., weights) of the filters. To reduce the number of parameters, a parameter sharing strategy is adopted. Although parameter sharing reduces the capacity of the networks, it improves its generalization ability (see [14, Sec. 4.19]). The computational layers (i.e., convolutional layers) can be enhanced by replacing the linear filter with a nonlinear function: shallow multilayer perceptron (MLP) [29]. The CNN with shallow MLP is called network in network (NiN) [29]. With enough hidden units, MLP can represent arbitrary complex but smooth functions and hence can improve the separability of the extracted features. Thus, NiN is able to give lower recognition error than classical CNN.

As a filter in CNNs, a shallow MLP spatially convolves across the input channels. Because the filter itself is also a network, the resulting CNN is called NiN. But the MLP itself in NiN does not employ sparse connectivity. Instead, MLP is a fully (densely) connected network. Thus, many parameters of MLP have to be computed and stored. In addition, the dense connection characteristic of MLP makes NiN unable to extract local features in channel domain though it can extract local features in spatial domain. This limits the performance of NiN. To break through the limitation, in this paper, we propose to modify the fully connected MLP to a locally connected one. Throughout this paper, "fully connected" means "densely connected." Moreover, "locally connected" and "sparsely connected" are interchangeably used. This is accomplished by leveraging a kernel (a.k.a., filter) on each layer (or some layers) of the MLP. That is, the size of the kernel is smaller than that of the input. Because the convolution operation is conducted in the embedded MLP of the convolution neural networks, we call the proposed method convolution in convolution (CiC). In summary, the contributions of this paper and the merits of the proposed CiC method are as follows.

1) A fully sparse (locally connected) shallow MLP and several partially sparse shallow MLPs (e.g., MLP-010) are proposed and are used as convolutional filters. The convolutional filter itself is obtained by convolving a linear filter.
2) We develop a CNN method (called CiC) with either the fully sparse MLPs or partially sparse MLPs. In CiC, shared convolution is conducted in the spatial domain and unshared convolution is conducted along the channel dimension.
3) The basic version (i.e., CiC-1-D) of CiC utilizes 1×1 convolutions in spatial domain and applies 1-D filtering along the channel dimension. We then generalize CiC-1-D into CiC-3-D by replacing the 1×1 convolutions with $n \times n$ convolutions.
4) The proposed CiC method significantly outperforms NiN in reducing the test error rate at least on the CIFAR10 data set, CIFAR100 data set, and their augmented versions.
5) The proposed idea of sparse space–channel connection is general and can be generalized to other deep CNNs such as VggNet [43].

Manuscript received March 23, 2016; revised October 30, 2016 and January 24, 2017; accepted February 20, 2017. Date of publication March 16, 2017; date of current version April 16, 2018. This work was supported in part by the National Basic Research Program of China (973 Program) under Grant 2014CB340400, in part by the National Natural Science Foundation of China under Grant 61632081, and in part by the Research Fund of Hainan Tropical Ocean University under Grant QYXB201501.

Y. Pang, M. Sun, and X. Jiang are with the School of Electrical and Information Enginnering, Tianjin University, Tianjin 300072, China (e-mail: pyw@tju.edu.cn; sml@tju.edu.cn; jiangxiaoheng@tju.edu.cn).

X. Li is with the Center for OPTical IMagery Analysis and Learning, State Key Laboratory of Transient Optics and Photonics, Xi'an Institute of Optics and Precision Mechanics, Chinese Academy of Sciences, Xi'an 710119, China (e-mail: xuelongli@opt.ac.cn).

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

Digital Object Identifier 10.1109/TNNLS.2017.2676130

The rest of this paper is organized as follows. We review related work in Section II. The proposed method is presented in Section III. Subsequently, the experimental results are provided in Section IV. We then conclude in Section V based on the experimental results.

II. Related Work

In this section, we will first briefly the basic components of the CNNs. Next, we will review some directions of CNNs that are related to our work.

In general, CNNs are mainly organized in interweaved layers of two types: convolutional layers and pooling (subsampling) layers [2], [5], [7], [20], [23], [30], [50] with a convolutional layer or several convolutional layers followed by a pooling layer. The role of the convolutional layers is feature representation with the semantic level of the features increasing with the depth of the layers. Each convolutional layer consists of several feature maps (a.k.a., channels). Each feature map is obtained by sliding (convoluting) a filter over the input channels with predefined stride followed by a nonlinear activation. Different feature maps correspond to different parameters of filters with a feature map sharing the same parameters. The filters can be learned with backpropagation algorithm. Pooling is a process that replaces the output of its corresponding convolutional layers at certain location with summary statistic of the nearby outputs [2], [51]. Pooling over spatial regions contribute to make feature representation become translation and rotation invariant and also contribute to improve the computational efficiency of the network. The layers after the last pooling layer are usually fully connected and are aimed at classification. The number of layers is called the depth of the network, and the number of units of each layer is called the width of the network. The number of feature maps (channels) in each layer can also represent the width (breadth) of CNNs. The depth and width determine the capacity of CNNs.

Generally speaking, there are six directions to improve the performance of the CNNs with some of the them are overlapping: 1) increasing the depth; 2) increasing the width; 3) modifying the convolution operation [6], [13], [20], [25], [26]; 4) modifying the pooling operation [49], [15], [4], [27], [40], [8], [46], [12], [33]; 5) reducing the number of parameters; and 6) modifying the activation function. Our method is closely related to NiN, and NiN is relevant to the first three directions.

A. Increasing the Depth

Large depth of deep CNNs is one of the main differences between deep CNNs and traditional neural networks [5], [38]. LeNet-5 [23] is a seven-layer version of CNNs, which has three convolutional layers, two pooling (subsampling) layers, and two fully connected layers. AlexNet [20] contains eight learned layers (not accounting the pooling layers or taking a convolutional layer followed by a pooling layer as a whole) with five convolutional ones and three fully connected ones. In VggNet [43], the depth is up to 19. GoogLeNet [42] contains 22 layers. Using gating units to regulate the flow of information through a network, highway networks [44] open up the possibility of effectively and efficiently training hundreds of layers with stochastic gradient descent. In ResNet [16], the depth of 152 results in state-of-the-art performance. Zeiler and Fergus [32] showed that having a minimum depth to the network, rather than any individual section is vital to the model's performance. By incorporating micronetworks, NiN [29] also increases the depth. The depth of our method is the same as that of NiN so that the comparison of NiN and our method is fair.

B. Increasing the Width

Increasing the number of feature maps in a convolutional layer yields enriched features and hence is expected to improve the CNNs [23]. Zeiler and Fergus [32] found that increasing the size of the middle convolution layers gives a useful gain in performance. GoogLeNet [42] is famous for not only its large depth but also its large width. In GooLeNet, a group of convolution filters of different sizes form an Inception module. Such an inception module greatly increases the network width. The OverFeat network [45] utilizes more than 1000 feature maps in both the fourth and fifth convolutional layers. It is noted that large width implies large computational cost. Without increasing the width, our method outperforms NiN in terms of test error rate.

C. Modifying the Convolution Operation

There are several ways to modify the convolution operation: changing the sliding stride, filter size, and filter type. It was found that a large convolution stride leads to aliasing artifacts [32]. Therefore, it is desirable to use small stride. Moreover, decreasing the filter size of the first convolution layer from 11×11 to 7×7 is positive for performance improvement. In the NggNet [43], 3×3 convolutional filters are adopted. Rather than learning a set of separate set of weights at every spatial location, in titled convolution [13], we can learn a set of filters that we rotate through as we move through space [10]. Modifying the filter type is an important attempt to develop effective CNNs [32]. While most of methods employ linear filter, NiN [29] adopts a nonlinear filter: shallow MLP that significantly enhances the representational power of CNNs [42]. CSNet [21] utilizes cascaded subpatch filters for convolution computation. Our method directly modifies NiN in the sense of convolutional layers.

III. Proposed Method: CiC

In this section, we present an improved NiN [29], which we call CiC. One of the characteristics of CiC is that sparse shallow MLPs are used for convolutionally computing the convolutional layers and the shallow MLPs themselves are obtained by convolution. The proposed CiC is equivalent to apply unshared convolution (i.e., sparse connection) across the channel dimension and applying shared convolution across the spatial dimension in some computational layers. By contrast, dense connection along the channel dimension is adopted in the traditional NiN. The advantages of spare connection

(i.e., convolution) in the channel domain against the dense connection in the channel domain are threefold.

1) Sparse connection in the channel domain yields less number of parameters, which is helpful for improving computational efficiency and decreasing memory storage.
2) Sparse connection is positive to alleviate the well-known overfitting problem when the number of training data is limited (in most cases, only limited training data are available).
3) Sparse connection provides opportunity to extract local features in the channel dimension and local features are expected to enhance the generalization ability of the neural networks. However, applying sparse connection on all layer is not the best choice and the reason is given in Section III-A.

We first describe the basic idea of CiC (i.e., CiC-1-D) with sparse and shallow MLP and then extend it to 3-D case (i.e., CiC-3-D).

A. From Dense Shallow MLP to Sparse Shallow MLP

In classical CNNs, linear filters are used for calculating the convolutional layers. However, there is evidence that more complex and nonlinear filter such as shallow MLP is preferable to the simple linear one [29]. In NiN [29], dense (i.e., fully connected) MLP is used as a filter (kernel). In our method, we propose to modify the dense MLP [see Fig. 1(a) for an example] to sparse one [see Fig. 1(b)–(f) for examples]. We divide sparse MLPs into two categories: fully sparse MLP and partially MLP.

The idea of the proposed method is graphically supported by Fig. 1. Fig. 1(a) is two-hidden-layer dense MLP that has 48, 24, and 8 free parameters (weights) in hidden layer 1 (the first hidden layer), hidden layer 2 (the second hidden layer), and output layer, respectively. Totally, there are $48 + 24 + 8 = 80$ parameters in the dense MLP. Fig. 1(b) is a two-hidden-layer fully sparse MLP that is obtained by convolving a linear filter with three weights (called inner filter) across each layer. The sparse MLP with parameter sharing mechanism has only three free parameters in each layer and $3 + 3 + 3 = 9$ parameters in total. If unshared convolution is employed, then the number of parameters became 36. Therefore, the sparse MLP has fewer parameters than the dense counterpart. Few parameters are capable of reducing memory consumption, increasing statistical efficiency, and also reducing the amount of computation needed to perform forward and back-propagation [10].

In addition to fully sparse MLP, partially sparse MLP can be used. Fig. 1(c)∼(f) show several possible partially sparse MLPs. To distinguish the different partially spare MLPs, we use '1' to mean that a layer is locally connected and use '0' to mean that the layer is fully connected. A sequence of the labels is used for expressing a certain type of partial sparse MLP. For example, in Fig. 1(c), hidden layer 1 is densely connected (labeled by '0'), hidden layer 2 is sparsely connected (labeled by '1'), and the output layer is densely connected (labeled by '0'). Therefore, we utilize "010" to represent the specific type of the partially sparse MLP. Specifically, the MLP in

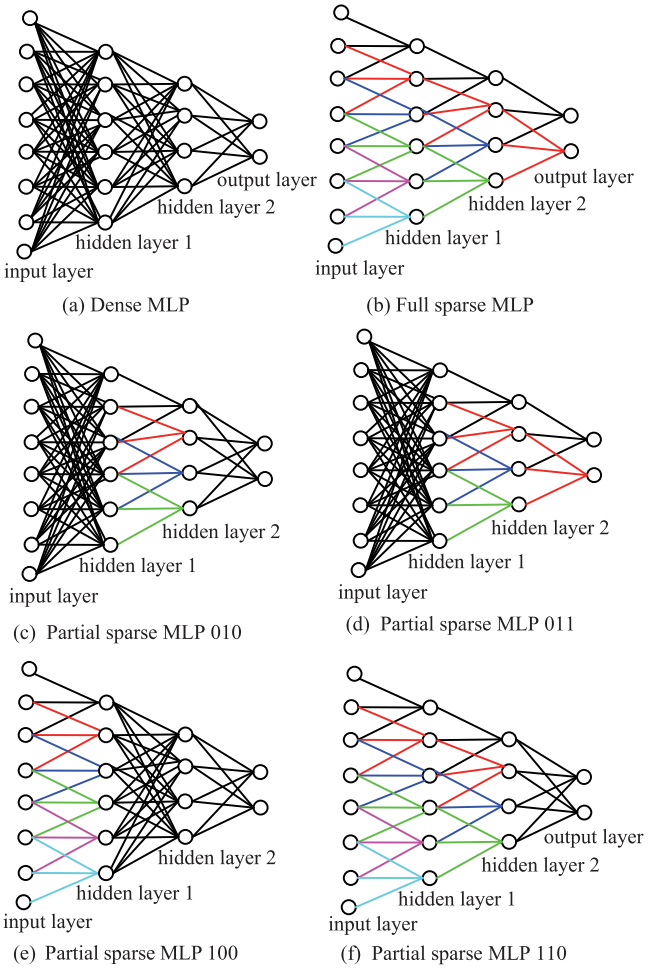


Fig. 1. Shallow MLP. (a) Dense (fully connected) MLP. (b) Full sparse MLP. (c)–(e) Different types of partial sparse MLPs.

Fig. 1(c) is called MLP-010. Similarly, the MLPs in Fig. 1(b) and (d)–(f) are called MLP-111, MLP-011, MLP-100, and MLP-110, respectively. For the sake of simplicity and clarity, '1' and '0' stand for "sparse" and "dense," respectively. Therefore, MLP-010 can be expressed as MLP-dense-sparse-dense. Similarly, MLP-111, MLP-100, and MLP-110 can be denoted by MLP-sparse-sparse-spare (i.e., fully sparse MLP), MLP-sparse-dense-dense, and MLP-sparse-sparse-dense, respectively.

It is noted that fully sparse MLP (i.e., MLP-111 or MLP-sparse-sparse-sparse) does not necessarily outperform partially sparse MLPs (e.g., MLP-010 or MLP-dense-sparse-dense). In fact, the experimental results show that MLP-dense-sparse-dense is superior to MLP-spare-spare-sparse. The reason is as follows. First, the input RGB color images have only three channels. The number of channels of the input layer is so small that convolution along the channel dimension is not necessary because the size of the filter is either two or one that is comparable with the number (i.e., three) of the input channels. Therefore, sparse connection cannot be applied on the input layer and dense connection is the unique choice. Sparse connection can applied only on the second layer. In this sense, MLP-dense-spare-dense is preferable to MLP-sparse-spare-sparse. Second, dense connection is able to extract

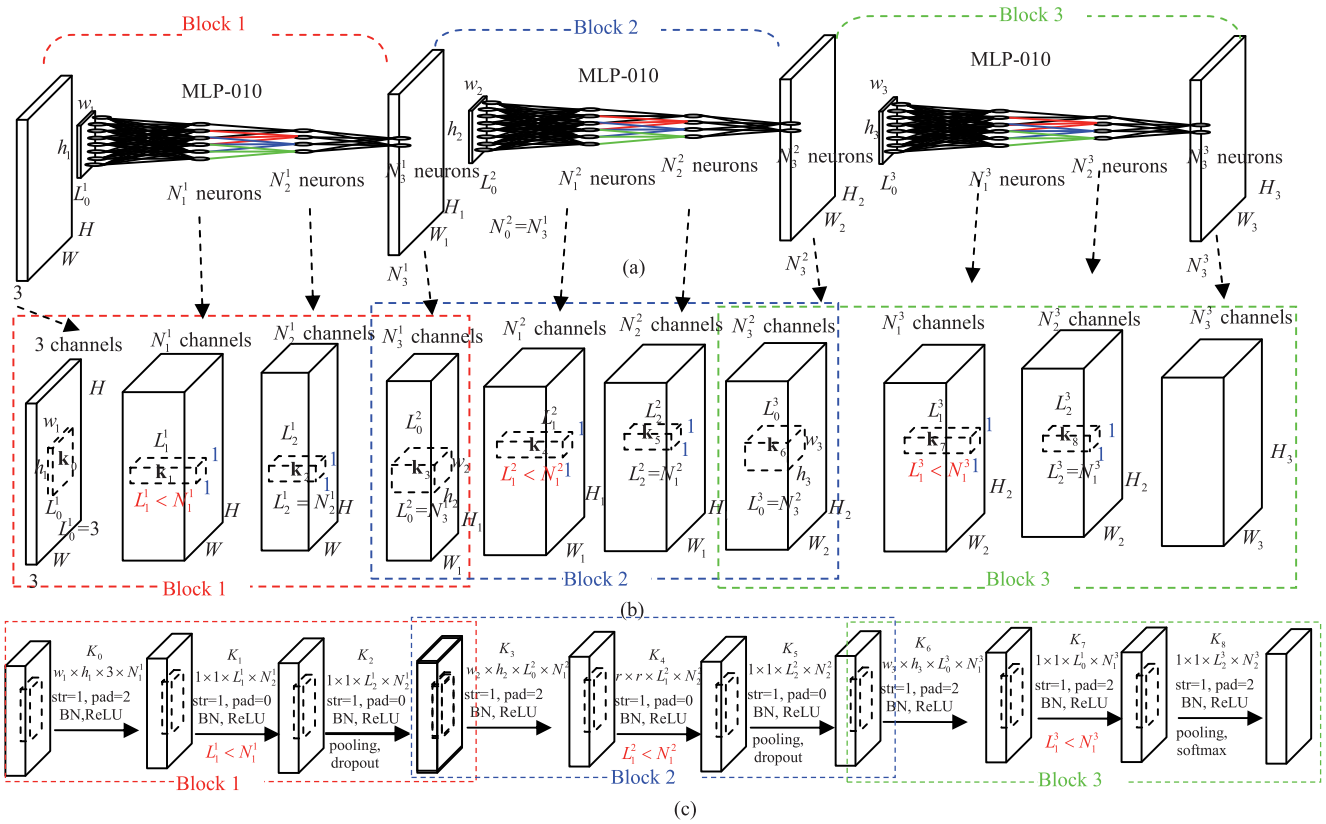


Fig. 2. Architecture of CiC-1-D. (a) Directly showing the role of MLP-010. (b) Kernels and their constraints for implementing CiC-1-D. (c) Architecture and main steps of CiC-1-D.

global features and local connection is good at extracting local features. Local extracts are discriminative when the correlation within a local region is large. It is well known that the correlation in the spatial domain is significant because neighboring elements (e.g., pixels) have similar appearance in most cases. This property has widely used in many tasks of computer vision (e.g., using contiguity prior with Markov random fields [28]). Compared with the correlation in spatial domain, the correlation in channel domain is weak. It is not guaranteed that neighboring elements in the channel domain are similar and highly correlated. Therefore, dense connection across channels is required for extracting global features. Nevertheless, some degree of correlation along the channel direction can be forcedly generated by sparse connection. However, applying sparse connection on all the layers makes the network weak in capturing global features. Consequently, it is desirable to interweavely applying dense connection and sparse connection for extracting both global and local features in the channel domain. Therefore, the partial sparse MLP is better than the fully sparse MLP.

It is also noted that our experimental results show that unshared convolution is better than shared convolution in the sense of reducing the test error rate. Therefore, unshared convolution is adopted along the channel dimension. Unshared convolution means that a set of different weights are used for sparse connection.

In summary, the effectiveness of the sparse channel connection stems from two aspects. First, sparse channel connections result in less number of parameters and hence are less prone to ovefitting. Second, sparse channel connections make it possible to extract local features across channels (i.e., in the channel domain).

B. CiC With Sparse and Shallow MLP (1-D Filtering Across Channels With Large Filter): CiC-1-D

As stated above, we propose to employ unshared convolution for computing sparse and shallow MLP. In our CiC method, the sparse MLP is regarded as a convolution filter and inserted into the framework of CNNs.

Fig. 2(a) shows the proposed CiC with MLP-010. Assume that all the input color images are of size $W \times H$. As NiN, CiC also has three building blocks (i.e., block 1, block 2, and block 3). In block i, the input of an MLP-010 is of size $w_i \times h_i$ in spatial domain and L_0^i in channel domain. The numbers of the neurons in the first hidden layer, the second hidden lay, and the output layer of an MLP-010 in block i are denoted by N_1^i, N_2^i, and N_3^i, respectively.

Fig. 2(a) explicitly shows the role of MLP-010 in constructing CiC, whereas Fig. 2(b) and (c) shows the architecture of CiC in the manner of convolutional layers. Fig. 2(b) and (c) is equivalent to Fig. 2(a). Note that the pooling layers exist but are not shown in Fig. 2(b). In Fig. 2(b), the small cube (i.e., the dashed cube) inside a large cube (i.e., solid cube) stands for a filter (kernel). The kernel is a three-order tensor $\mathbf{k} \in R^{x \times y \times z}$, where x and y index the spatial domain (i.e., x and y are the length of the kernel in horizontal and

TABLE I
SIZES AND CONSTRAINTS OF THE KERNELS ($\mathbf{k}_i$ IS A THREE-ORDER TENSOR AND $\mathbf{K}_i$ IS ITS CORRESPONDING FOUR-ORDER TENSOR)

block 1	$\mathbf{k}_0$	$\mathbf{K}_0$	$\mathbf{k}_1$	$\mathbf{K}_1$	$\mathbf{k}_2$	$\mathbf{K}_2$
	$w_1 \times h_1 \times L_0^1$	$w_1 \times h_1 \times L_0^1 \times N_1^1$	$1 \times 1 \times L_1^1$	$1 \times 1 \times L_1^1 \times N_2^1$	$1 \times 1 \times L_2^1$	$1 \times 1 \times L_2^1 \times N_3^1$
	$L_0^1 = 3$		$L_1^1 < N_1^1$		$L_2^1 = N_2^1$	
block 2	$\mathbf{k}_3$	$\mathbf{K}_3$	$\mathbf{k}_4$	$\mathbf{K}_4$	$\mathbf{k}_5$	$\mathbf{K}_5$
	1	2	3	4	5	6
	$L_0^2 = N_3^1$		$L_1^2 < N_1^2$		$L_2^2 < N_2^2$	
block 3	$\mathbf{k}_6$	$\mathbf{K}_6$	$\mathbf{k}_7$	$\mathbf{K}_7$	$\mathbf{k}_8$	$\mathbf{K}_8$
	1	2	3	4	5	6
	$L_0^3 = N_3^2$		$L_1^3 < N_1^3$		$L_2^3 < N_2^3$	

vertical axis, respectively) and z indexes the channel domain (i.e., z is the length of the kernel in the channel axis). In each block, there are three kernels, corresponding to the input layer, the first hidden layer, and the second layer of an MLP-010. The kernels in block 1, block 2, and block 3 are ($\mathbf{k}_0$, $\mathbf{k}_1$, $\mathbf{k}_2$), ($\mathbf{k}_3$, $\mathbf{k}_4$, $\mathbf{k}_5$), and ($\mathbf{k}_6$, $\mathbf{k}_7$, $\mathbf{k}_8$) [see Fig. 2(b)], respectively. If the number of output channels obtained by a kernel $\mathbf{k}_i$ is taken into account, the kernel can be expressed as a four-order tensor $\mathbf{K}_i \in R^{x \times y \times z \times c}$ [see Fig. 2(c)], where the first three indices (i.e., x, y, and z) have the same meaning as that of $\mathbf{k}_i$ and the last index c stands for the number of feature maps (channels) obtained by the kernel $\mathbf{k}_i$.

Now we discuss the three kernels in block 1.

1) First Kernel $\mathbf{k}_0$ *(*$\mathbf{K}_0$*):* In the input layer of block 1 (also the input of the first MLP010), the kernel $\mathbf{k}_0 \in R^{w_1 \times h_1 \times L_0^1}$ (or equivalently $\mathbf{K}_0 \in R^{w_1 \times h_1 \times L_0^1 \times N_1^1}$) is used for filtering. The small numbers w_1 and h_1 are larger than 1 (e.g., $w_1 = 5$ and $h_1 = 5$). Because the input is usually a color image, so the channel number L_0^1 equals to three (i.e., three color channels). Because the number of channels is very small, the channels have to be densely (fully) connected using the kernel $\mathbf{k}_0$ with its size being $w_1 \times h_1 \times 3$. Denote N_1^1 the number of channels output by $\mathbf{K}_0$.

2) Second Kernel $\mathbf{k}_1$*(*$\mathbf{K}_1$*):* The output of the first kernel is the input of the second kernel $\mathbf{k}_1$ (or $\mathbf{K}_1$), which is of size $1 \times 1 \times L_1^1$ (or $1 \times 1 \times L_1^1 \times N_2^1$) with $L_1^1 < N_1^1$. The convolution in spatial domain with the 1×1 kernel plays an important role in dimensionality reduction [29], [42]. Because of $L_1^1 < N_1^1$(i.e., the length L_1^1 of the filter in channel dimension is smaller than the number N_1^1 of the input channels), the kernel $\mathbf{k}_1$ connects the N_1^1 channels in a convolutional (i.e., sparse) manner. In Section IV-B, we state how to choose the optimal kernel length in channel dimension.

N_2^1 is the number of channels output by applying the kernels $\mathbf{K}_1 \in R^{1 \times 1 \times L_1^1 \times N_2^1}$. Usually, zero padding is used for convolutional implementation [10]. However, throughout this paper, no padding is applied along the channel direction. Consequently, the number N_2^1 of channels output by applying the kernels $\mathbf{K}_1 \in R^{1 \times 1 \times L_1^1 \times N_2^1}$ is completely determined by the kernel length L_1^1 and the number N_1^1 of input channels. Specifically, we have

$$N_2^1 = N_1^1 - L_1^1 + 1. \tag{1}$$

The success of CiC is due to the following two factors related to the second kernel in each block: 1) the convolutional manner of sparsely connecting different channels in channel dimension and 2) the sparse connection is accomplished by unshared convolution.

3) Third Kernel $\mathbf{k}_2$*(*$\mathbf{K}_2$*):* $\mathbf{k}_2 \in R^{1 \times 1 \times L_2^1}$ (or $\mathbf{K}_2 \in R^{1 \times 1 \times L_2^1 \times N_3^1}$) is also a 1×1 filter in spatial domain. The length L_2^1 of the filer in channel dimension is equal to the number N_2^1 of the input channels (i.e., $L_2^1 = N_2^1$), meaning that the channels corresponding to the same spatial location are fully connected.

The output of block 1 is used as the input of block 2. The computation process of block 2 and block 3 is similar to that of block 1. The sizes and constraints of the three and four order tensors are given in Table I.

Batch normalization (BN) [18] is adopted to normalize the convolutional layers. Rectified linear unit (ReLU) nonlinearilty [20], [34] is used for modeling a neuron's output. Max-pooling is applied on the results of ReLU. Moreover, dropout [17], [20] is also conducted. Fig. 2(c) shows the main steps of the proposed CiC method where "str" and "pad" stand for the convolution stride and padding pixels, respectively.

C. Generalized CiC With 3-D Filtering Across Channel–Spatial Domain: CiC-3-D

It can be seen from Section III-B that the second kernel (i.e., $\mathbf{K}_1 \in R^{1 \times 1 \times L_1^1 \times N_2^1}$, $\mathbf{K}_4 \in R^{1 \times 1 \times L_1^2 \times N_2^2}$, and $\mathbf{K}_7 \in R^{1 \times 1 \times L_1^3 \times N_2^3}$) in each block is the key of the proposed CiC where MLP-010 is adopted. These kernels perform 1×1 convolutions in spatial domain and 1-D convolutions in the channel dimension. Hence, the convolutions in different spatial locations are independently implemented. In this section, we propose to breakthrough this independence by changing the 1×1 convolutions to $n \times n$ convolutions with $n > 1$. Accordingly, in the generalized CiC (called CiC-3-D), the sizes of $\mathbf{K}_1$, $\mathbf{K}_4$, and $\mathbf{K}_7$ are changed from $1 \times 1 \times L_1^1 \times N_2^1$, $1 \times 1 \times L_1^2 \times N_2^2$, and $1 \times 1 \times L_1^3 \times N_2^3$ to $n \times n \times L_1^1 \times N_2^1$, $n \times n \times L_1^2 \times N_2^2$, and $n \times n \times L_1^3 \times N_2^3$, respectively. Fig. 3 shows the architecture of the proposed CiC-3-D. In general, $n = 5$ is able to yield good performance.

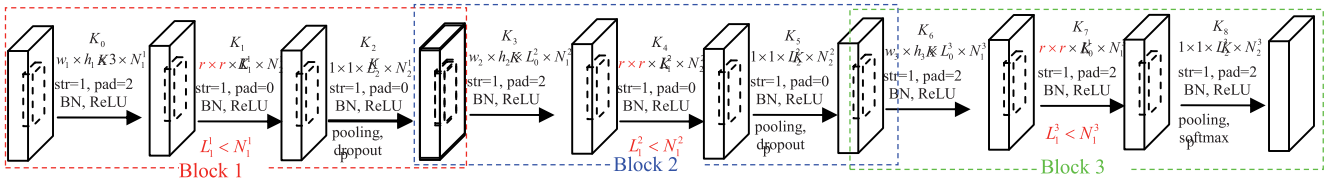


Fig. 3. Architecture of CiC-3-D.

TABLE II

TEST ERROR RATE (%) WITH DIFFERENT KERNEL LENGTH L_1^2 IN CHANNEL DIMENSION OF $\mathbf{K}_4$ IN BLOCK 2 OF CiC-1-D. NOTE THAT MLP-010 IS APPLIED IN ONLY BLOCK 2 AND DENSE MLP IS APPLIED IN BOTH BLOCK 1 AND BLOCK 3

		$L_1^2=3$	$L_1^2=6$	$L_1^2=12$	$L_1^2=24$	$L_1^2=48$
block 1	$\mathbf{K}_0$	$5\times 5\times 3\times 192$	$5\times 5\times 3\times 192$	$5\times 5\times 3\times 192$	$5\times 5\times 3\times 192$	$5\times 5\times 3\times 192$
	$\mathbf{K}_1$	$1\times 1\times 192\times 192$	$1\times 1\times 192\times 192$	$1\times 1\times 192\times 192$	$1\times 1\times 192\times 192$	$1\times 1\times 192\times 192$
	$\mathbf{K}_2$	$1\times 1\times 192\times 192$	$1\times 1\times 192\times 192$	$1\times 1\times 192\times 192$	$1\times 1\times 192\times 192$	$1\times 1\times 192\times 192$
block 2	$\mathbf{K}_3$	$5\times 5\times 192\times 192$	$5\times 5\times 192\times 192$	$5\times 5\times 192\times 192$	$5\times 5\times 192\times 192$	$5\times 5\times 192\times 192$
	$\mathbf{K}_4$	$1\times 1\times 3\times 190$	$1\times 1\times 6\times 187$	$1\times 1\times 12\times 181$	$1\times 1\times 24\times 169$	$1\times 1\times 48\times 143$
	$\mathbf{K}_5$	$1\times 1\times 190\times 192$	$1\times 1\times 187\times 192$	$1\times 1\times 181\times 192$	$1\times 1\times 169\times 192$	$1\times 1\times 143\times 192$
block 3	$\mathbf{K}_6$	$3\times 3\times 192\times 192$	$3\times 3\times 192\times 192$	$3\times 3\times 192\times 192$	$3\times 3\times 192\times 192$	$3\times 3\times 192\times 192$
	$\mathbf{K}_7$	$1\times 1\times 192\times 192$	$1\times 1\times 192\times 192$	$1\times 1\times 192\times 192$	$1\times 1\times 192\times 192$	$1\times 1\times 192\times 192$
	$\mathbf{K}_8$	$1\times 1\times 192\times 10$	$1\times 1\times 192\times 10$	$1\times 1\times 192\times 10$	$1\times 1\times 192\times 10$	$1\times 1\times 192\times 10$
error rate (%)		9.07	9.15	9.15	9.47	9.68

IV. EXPERIMENTAL RESULTS

We call the method proposed in Section III-B CiC-1-D and the method proposed in Section III-C CiC-3-D. In this section, we compare the proposed method CiC-3-D with NiN [29], Maxout (maxout networks) [11], networks with learned activation functions (NiN+LA) [1], FitNet (thin and deep networks) [37], deeply supervised network (DSN) [22], DropConnect (networks using dropconnect) [48], deep attention selective networks (dasNet) [39], highway (networks allowing information flow on information highways) [44], all convolutional networks (ALL-CNN) [40], recurrent cnns (RCNNs) [24], VggNet-like network [43], and ResNet (deep residual networks) [16] on the CIFAR10 data set [19], the CIFAR100 data set [19], and their augmented versions. The proposed methods are implemented using the MatConvNet toolbox [47].

A. Data Sets

The CIFAR10 data set contains ten classes of images with 6000 images per class and 60 000 images in total. Among the 60 000 images, 50 000 ones are used for training and the rest 10 000 ones are used for testing. All the images have three-color channels, and the image size is 32×32.

By randomly flipping each training images, we obtain an augmented data set that we call CIFAR10+. CIFAR10+ is used for tuning the parameters and showing the intermediate results. Moreover, we obtain a much larger augmented dataset (called CIFAR10++) by padding four pixels on each side and then randomly cropping and flipping on the fly.

The CIFAR100 data set [38] has the same number of training images and test images as the CIFAR10. The difference is that the CIFAR100 contains 100 instead of ten classes. Therefore, the number of images in each class is only one-tenth of that of CIFAR10. The 100 classes are grouped into 20 superclasses. Each image has two labels. One is the "fine" label indicating the specific class, and the other one is the "coarse" label indicating the superclass. CIFAR100 is much challenging than CIFAR10 for classification. The augmented version of CIFAR100 is called CIFAR100+.

B. Configurations and Intermediate Results on the CIFAR10+ Data Set

The CIFAR10+ data set is used for parameter selection, and the selected parameters are to be used for evaluating the proposed method on the CIFAR10, CIFAR10++, and CIFAR100 data sets. To quickly obtain the optimal parameters, only 60 training epochs are employed. It is note that 230 epochs are adopted in Sections IV-C–IV-E.

The sizes (parameters) of the kernels $\mathbf{K}_i$, $i = 0, \ldots, 8$, determine the performance of CiC-1-D and CiC-3-D. It is difficult to jointly choose the optimal parameters. We experimentally choose the optimal parameters of CiC-1-D in a greedily manner where a parameter varies until the optimal value is found and the other parameters are kept unchanged during the greedy optimization process. The obtained optimal parameters are shared with CiC-3-D.

To investigate the influence of the length of the kernel in the channel dimension of CiC-1-D, in Table II, MLP-010 is used for constructing block 2 (where sparsity holds because of $L_1^2 < N_1^2$) and dense MLPs are used for constructing block 1 and block 3 (where $L_1^1 = N_1^1$ and $L_2^1 = N_2^1$ hold for block 1 and $L_1^3 = N_1^3$ and $L_2^3 = N_2^3$ hold for block 3). The kernels

TABLE III

TEST ERROR RATES (%) VARY WITH THE NUMBER N_1^2 (MARKED IN RED) OF THE INPUT CHANNELS OF $\mathbf{K}_4$ IN BLOCK 2 WHEN THE KERNEL LENGTH L_1^2 IN CHANNEL DIMENSION OF $\mathbf{K}_4$ IS FIXED $L_1^2 = 3$ (MARKED IN GREEN)

		$N_1^2 = 160$	$N_1^2 = 192$	$N_1^2 = 224$	$N_1^2 = 256$
block 1	$\mathbf{K}_0$	$5\times 5\times 3\times 192$	$5\times 5\times 3\times 192$	$5\times 5\times 3\times 192$	$5\times 5\times 3\times 192$
	$\mathbf{K}_1$	$1\times 1\times 192\times 192$	$1\times 1\times 192\times 192$	$1\times 1\times 192\times 192$	$1\times 1\times 192\times 192$
	$\mathbf{K}_2$	$1\times 1\times 192\times 192$	$1\times 1\times 192\times 192$	$1\times 1\times 192\times 192$	$1\times 1\times 192\times 192$
block 2	$\mathbf{K}_3$	$5\times 5\times 192\times 160$	$5\times 5\times 192\times 192$	$5\times 5\times 192\times 224$	$5\times 5\times 192\times 256$
	$\mathbf{K}_4$	$1\times 1\times 3\times 158$	$1\times 1\times 3\times 190$	$1\times 1\times 3\times 222$	$1\times 1\times 3\times 254$
	$\mathbf{K}_5$	$1\times 1\times 158\times 192$	$1\times 1\times 190\times 192$	$1\times 1\times 222\times 192$	$1\times 1\times 254\times 192$
block 3	$\mathbf{K}_6$	$3\times 3\times 192\times 192$	$3\times 3\times 192\times 192$	$3\times 3\times 192\times 192$	$3\times 3\times 192\times 192$
	$\mathbf{K}_7$	$1\times 1\times 192\times 192$	$1\times 1\times 192\times 192$	$1\times 1\times 192\times 192$	$1\times 1\times 192\times 192$
	$\mathbf{K}_8$	$1\times 1\times 192\times 10$	$1\times 1\times 192\times 10$	$1\times 1\times 192\times 10$	$1\times 1\times 192\times 10$
error rate (%)		9.42	9.07	**8.97**	9.33

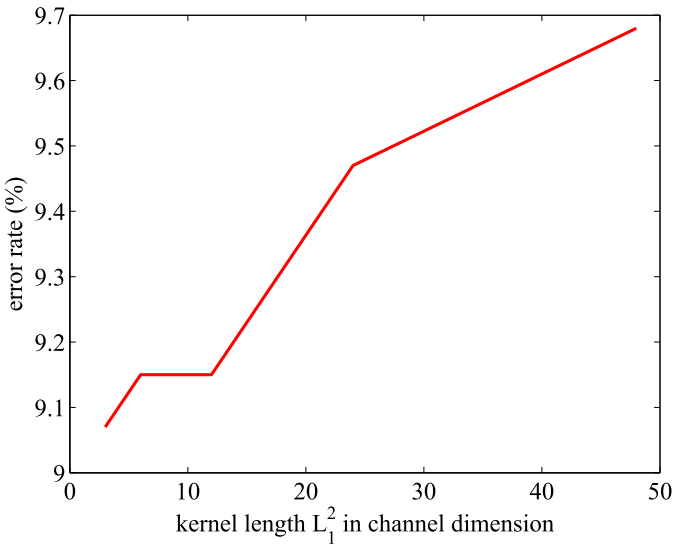


Fig. 4. Test error rates (%) with different kernel lengths L_1^2 in channel dimension of $\mathbf{K}_4$ in block 2 of CiC-1-D.

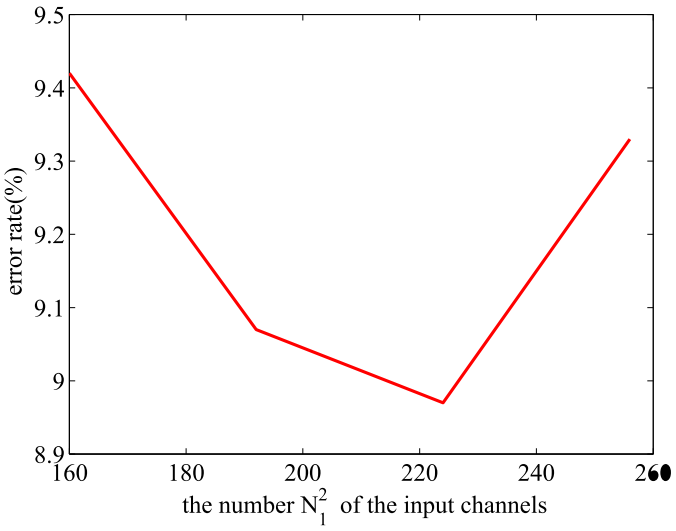


Fig. 5. Test error rates versus the number N_1^2 of the input channels of $\mathbf{K}_4$ in block 2.

($\mathbf{K}_0$, $\mathbf{K}_1$, and $\mathbf{K}_2$) in bock 1 and the kernels ($\mathbf{K}_6$, $\mathbf{K}_7$, and $\mathbf{K}_8$) in block 3 are fixed. In block 2, the size of $\mathbf{K}_4$ is expressed as $1 \times 1 \times L_1^2 \times N_2^2$, where L_1^2 is the kernel length in the channel dimension. Several values of L_1^2 are used with N_2^2 and L_1^3 changing with L_1^2 according to the $N_2^2 = N_2^1 - L_1^2 + 1$ and $L_1^2 = N_2^2$, respectively (i.e., valid convolution without zero padding). The corresponding test error rates are given in the bottom of Table II and are visualized in Fig. 4. The red numbers in Table II are the various L_1^2, and the blue numbers are the corresponding N_2^2 and N_1^2.

From Fig. 4, one can find that small kernel length $L_1^2 = 3$ in channel dimension results in the lowest error rate 9.07%. *Thus, we choose three as the optimal kernel length in channel dimension whenever MLP-010 is used for sparse connection.*

The number N_1^2 of the input channels of $\mathbf{K}_4$ in block 2 is also important for classification. To seek the optimal value of N_1^2, we fix the kernel length $L_1^2 = 3$ in channel dimension and utilize different N_1^2 (i.e., 160, 192, 224, and 256). The resulting classification error rates on the CIFAR10+ data set are given in Table III and shown in Fig. 5. *It is observed that the best classification performance is obtained when the number of input channels of an MLP-010 is 224.*

In Tables II and III, MLP-010 is applied only in block 2 and dense MLPs are employed in both block 1 and block 3. To further investigate the importance of MLP-010, MLP-010 is applied in both block 1 and block 2, whereas dense MLP is applied in block 3. Moreover, all the three blocks can employ MLP-010. Note that the number of the input channels of the sparsely connected layer of MLP-010 is 224 and the kernel length in channel dimension is three. The corresponding results are given in the second row of Table IV from which one can observe that it is beneficial to apply MLP-010 in more blocks. The third row of Table IV shows the test error rates of CiC-3-D where the number of the input channels of the sparsely connected layer of MLP-010 is also 224 and the kernel length in channel dimension is also three. For CiC-3-D, it is also desirable to apply MLP-010 in all the three blocks.

TABLE IV
TEST ERROR RATES (%) WHEN MLP-010 IS APPLIED IN ONLY BLOCK 1, BLOCK 1 AND BLOCK 2, AND ALL THE THREE BLOCKS, RESPECTIVELY

	only block 2	block 1 and 2	block 1, 2, and 3
CiC-1D	9.23	—	—
CiC-3D	9.05	8.46	7.94

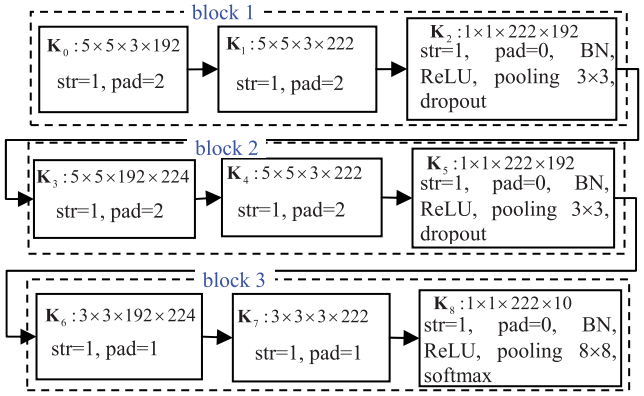


Fig. 6. Configuration of the proposed CiC-3-D.

TABLE V
LEARNING RATES USED FOR TRAINING CiC

Epoch interval	[1, 80]	[81, 180]	[181, 230]
From	0.5	0.5	0.005
To	0.5	0.005	0.0005
Step	0	-0.005	-0.0001

Comparing the second row and the third row of Table III, one can conclude that CiC-3-D outperforms CiC-1-D significantly. Thus, in the following experiments, only CiC-3-D is employed with MLP-010 being used in all the three blocks. The main parameters and operation of the proposed CiC-3-D are shown in Fig. 6.

In Fig. 6, "str" and "pad" stands for the convolution stride and padding pixels. "pooling 3×3" and "pooling 8×8" mean that max pooling are conducted with 3×3 template and 8×8 template, respectively.

The learning rate of a CNN is important for training. The learning rates in the first 80 epochs are identical to 0.5. From the 81th epoch to the 180th epoch, the learning rate decreases from 0.5 to 0.005 with step -0.005. From the 181th epoch to the 230th epoch, the learning rate decreases from 0.005 to 0.00005 with step -0.0001. Table V shows the learning rates of different training epochs.

C. Comparison With Other Methods on the CIFAR10 Data Set

In Section IV-B, only 60 training epochs are employed. Hereinafter, the training epochs of CiC-3-D are up to 230 epochs. Note that the CIFAR10+ data set is used in Section IV-B and the original CIFAR10 data set is used in this section.

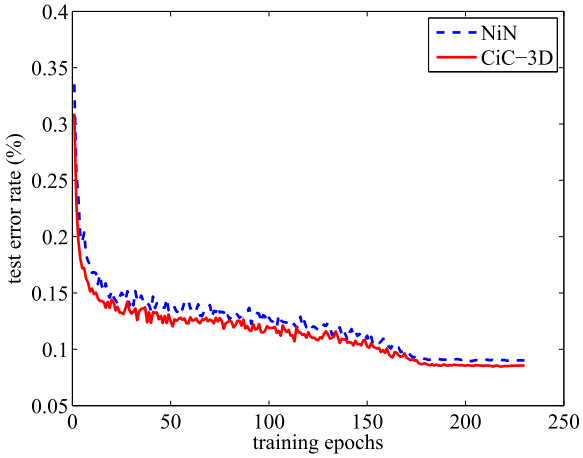


Fig. 7. Training error (%) versus training epoch of NiN and CiC-3-D on the CIFAR-10 data set.

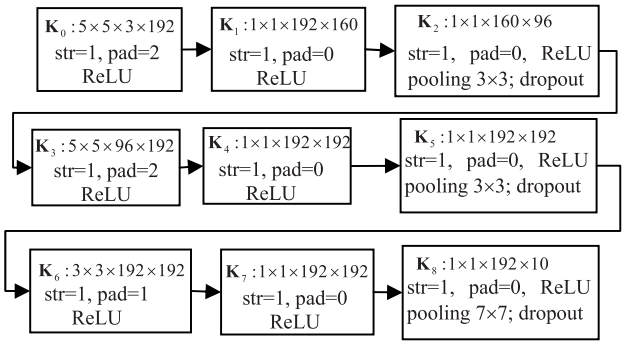


Fig. 8. Configuration of NiN [29] for the CIFAR10 data set.

TABLE VI
TEST ERROR RATES (%) ON THE CIFAR10 DATA SET

NiN [29]	DSN [22]	NiN-LA [1]	RCNN-160 [24]
10.41	9.78	9.59	8.69
CiC-3D	DropConnect [48]	dasNet [39]	All-CNN [40]
8.46	9.41	9.22	9.08

We first show in Fig. 7 the curves of training error rates versus training epochs of NiN [29] (see Fig. 8 for its architecture) and the proposed CiC-3-D (see Fig. 6) on the CIFAR-10 data set. It is observed that CiC-3-D has smaller training error rates than NiN. Moreover, the proposed method CiC-3-D convergences much faster than NiN.

Table VI gives the test error rates of NiN [29], DSN [22], NiN-LA units [1], RCNN-160 [24], DropConnect [48], dasNet [39], All-CNN [40], and CiC-3-D on the CIFAR10 data set. One can see from Table VI that CiC-3-D outperforms NiN by 1.95%. In addition, CiC-3-D gives 1.13% improvement over the NiN-LA. In Table VI, the proposed CiC-3-D outperforms all the other methods.

TABLE VII
TEST ERROR RATES (%) ON THE CIFAR10++ DATA SET

NiN [29]	ResNet-1202 [24]	NiN-LA [1]	RCNN-160 [24]
8.81	7.93	7.51	7.09
CiC-3D	Maxout [11]	FitNet [37]	DSN [22]
6.68	9.38	8.39	8.22
Highway [44]	All-CNN [40]	ResNet-110 [16]	-
7.54	7.25	6.43	-

TABLE VIII
TEST ERROR RATES (%) ON THE CIFAR100 DATA SET WITHOUT AUGMENT

NiN [29]	NiN-LA [1]	Highway [44]	RCNN-160 [24]
35.68	34.40	32.24	31.75
CiC-3D	Maxout [11]	FitNet [37]	DSN [22]
31.40	38.57	35.04	34.57
dasNet [39]	All-CNN [40]	-	-
33.78	33.71	-	-

D. Comparison With Other Methods on the CIFAR10++ Dataset

As stated in Section IV-A, the CIFAR10++ data set is a large version of CIFAR10. The configuration of CiC-3-D is the same as that in Section IV-C (i.e., Fig. 6).

The test error rates on the CIFAR10++ data set are given in Table VII. The test error rates of NiN and CiC-3-D are 8.81% and 6.68%, respectively. Consequently, CiC-3-D gives 2.13% improvement over NiN. The superiority of CiC-3-D grows as the training set increases. It is observed from Table VII that the proposed CiC-3-D is top two among all the 11 methods and the proposed CiC-3-D is comparable with the top one method, ResNet-110, which utilizes 110 computational layers.

E. Comparison With Other Methods on the CIFAR100 Data Set

The CIFAR100 data set is challenging than the CIFAR10 data set. But we still adopt the configuration in Fig. 6 for constructing CiC-3-D.

Table VIII shows that the test error rates of NiN [29], NiN+LA units [1], highway [44], RCNN-160 [24], and CiC-3-D are 35.68%, 34.40%, 32.24%, 31.75%, and 31.40%, respectively. CiC-3-D arrives at the lowest test error rate and outperforms NiN and NiN-LA by 4.28% and 3%, respectively.

The above experimental results show that the proposed CiC-3-D method significantly outperforms NiN in reducing the test error rate.

F. Generalize the Idea to VggNet-Like Network

The nature of the proposed method is applying fully sparse channel connection or partially sparse channel connection deep CNN. In addition to applying partially sparse channel connection to NiN, the proposed idea can be generalized to other types of CNNs. In this section, we generalize the idea to VggNet-like network [43].

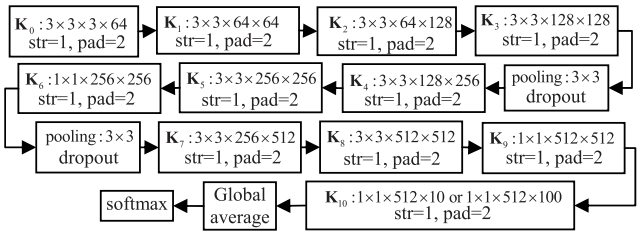


Fig. 9. Architecture of a VggNet-like network for the CIFAR data sets.

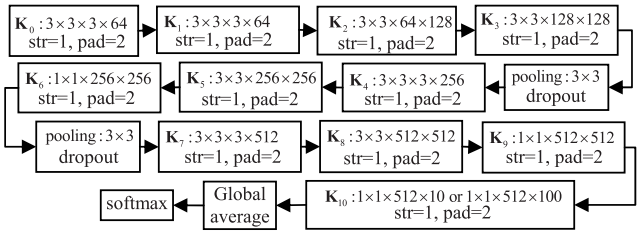


Fig. 10. Architecture of the CiC-VggNet for the CIFAR data sets.

TABLE IX
COMPARISON WITH VggNet AND CiC-VGGNet, AND NiN IN TERMS OF THE TEST ERROR RATE (%) ON THE CIFAR DATA SETS

Method	CIFAR10	CIFAR10+	CIFAR100+
VggNet	7.91	6.36	27.33
CiCNet	6.43	5.06	24.82

The basic idea of VggNet is constructing an architecture with very small (3 ×3) convolutional filters. Given the data sets of CIFAR10, CIFAR10+, and CIFAR100+ in Fig. 9, we construct a VggNet-like network. In Fig. 9, the last kernel $\mathbf{K_9}$ is of size $1\times1\times512\times10$ for the CIFAR10 and CIFAR10+ data sets, and its size is $1 \times 1 \times 512 \times 100$ for the CIFAR100+ data set. The CIFAR100+ is the augmented version of the CIFAR100 data set. We denote the method by VggNet. The depth of the VggNet is 11. The first nine kernels are of spatial size 3×3, and the last two kernels are of spatial size 1×1.

Corresponding to the architecture shown in Fig. 9, we show in Fig. 10 the architecture of the proposed network where sparse channel connection is adopted. Our method is called CiC-VggNet. Sparse channel connection is applied on $\mathbf{K_1}$, $\mathbf{K_4}$, and $\mathbf{K_7}$.

The experimental results are given in Table IX. On the CIFAR10, CIFAR10+, and CIFAR100+ data sets, the test error rates of VggNet are 7.91%, 6.36%, and 27.33%, respectively. On the three data sets, the test error rates of the proposed CiC-VggNet are 6.43%, 5.06%, and 24.82%, respectively. The results also demonstrate the effectiveness of the proposed idea.

V. CONCLUSION

In this paper, we have presented a CNN method (called CiC) where sparse shallow MLP is used for convolution.

Full sparse MLP and several types of partial sparse MLPs (e.g., MLP-010, MLP 011, MLP-100, and MLP-110) were proposed. The MLP-010 was employed in the experiments. The main idea is to sparsely connect different channels in an unshared convolutional manner. The basic version of CiC is CiC-1-D and its generalized version is CiC-3-D. In CiC-1-D, a 1-D filter is employed for connecting the second layer of each block. CiC-1-D was then generalized to CiC-3-D by utilizing a 3-D filter across channel–spatial domain.

In the experiments, a partially spare MLP, MLP-010, was adopted. In the future, other types of partially sparse MLPs can be implemented. Moreover, the proposed idea can be integrated to other state-of-the-art CNNs [21] so that better results can be expected. In addition, experiments on the task of object detection [35], [36] will be conducted.

REFERENCES

[1] F. Agostinelli, M. Hoffman, P. Sadowski, and P. Baldi, "Learning activation functions to improve deep neural networks," *CoRR*, vol. abs/1412.6830, 2014. [Online]. Available: http://arxiv.org/abs/1412.6830

[2] Y. Bengio, "Learning deep architectures for AI," *Found. Trends Mach. Learn.*, vol. 2, no. 1, pp. 127–131, 2009.

[3] J. Cao, Y. Pang, and X. Li, "Pedestrian detection inspired by appearance constancy and shape symmetry," in *Proc. IEEE Int. Conf. Comput. Vis. Pattern Recognit.*, Sep. 2016, pp. 1316–1324.

[4] T. Chan, K. Jia, S. Gao, J. Lu, Z. Zeng, and Y. Ma, "PCANet: A simple deep learning baseline for image classification?" *CoRR*, vol. abs/1404.3606, 2014. [Online]. Available: http://arxiv.org/abs/1404.3606

[5] C.-H. Chang, "Deep and shallow architecture of multilayer neural networks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 26, no. 10, pp. 2477–2486, Oct. 2015.

[6] D. Ciresan, U. Meier, and J. Schmidhuber, "Multi-column deep neural networks for image classification," *CoRR*, vol. abs/1202.2745, 2012. [Online]. Avaliable: http://arxiv.org/abs/1202.2745

[7] M. Gong, J. Liu, H. Li, Q. Cai, and L. Su, "A multi objective sparse feature learning model for deep neural networks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 26, no. 12, pp. 3263–3277, Dec. 2015.

[8] Y. Gong, L. Wang, R. Guo, and S. Lazebnik, "Multi-scale orderless pooling of deep convolutional activation features," in *Proc. Eur. Conf. Comput. Vis.*, 2014, pp. 392–407.

[9] M. Gong, J. Zhao, J. Liu, Q. Miao, and L. Jiao, "Change detection in synthetic aperture radar images based on deep neural networks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 27, no. 1, pp. 125–138, Jan. 2016.

[10] I. Goodfellow, A. Courville, and Y. Bengio, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2015.

[11] I. J. Goodfellow, D. Warde-Farley, M. Mirza, A. Courville, and Y. Bengio, "Maxout networks," *CoRR*, vol. abs/1302.4389, 2013. [Online]. Available: https://arxiv.org/abs/1302.4389

[12] B. Graham, "Fractional max-pooling," *CoRR*, vol. abs/1412.6071, 2014. [Online]. Available: https://arxiv.org/abs/1412.6071

[13] K. Gregor and Y. LeCun, "Emergence of complex-like cells in a termporal product network with local receptive fields," *CoRR*, vol. abs/1006.0448, 2010. [Online]. Available: http://arxiv.org/abs/1006.0448

[14] S. Haykin, *Neural Networks: A Comprehensive Foundation*, 2nd ed. Upper Saddle River, NJ, USA: Prentice-Hall, 1999.

[15] K. He, X. Zhang, S. Ren, and J. Sun, "Spatial pyramid pooling in deep convolutional networks for visual recognition," in *Proc. Eur. Conf. Comput. Vis.*, 2014, pp. 346–361.

[16] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *CoRR*, vol. abs/1512.03385, 2015. [Online]. Available: http://arxiv.org/abs/1512.03385

[17] G. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," *CoRR*, vol. abs/1207.0580, 2012. [Online]. Avaliable: http://arxiv.org/abs/1207.0580

[18] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *CoRR*, vol. abs/1502.03167, 2015. [Online]. Available: http://arxiv.org/abs/1502.03167

[19] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," M.S. thesis, Dept. Comput. Sci., Univ. Toronto, Toronto, ON, Canada, 2009.

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

[21] X. Jiang, Y. Pang, M. Sun, and X. Li, "Cascaded subpatch networks for effective CNNs," *IEEE Trans. Neural Netw.*, to be published.

[22] C. Lee, S. Xie, P. Gallagher, Z. Zhang, and Z. Tu, "Deeply-supervised nets," *CoRR*, vol. abs/1409.5185, 2014. [Online]. Avaliable: https://arxiv.org/abs/1409.5185

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

[24] M. Liang and X. Hu, "Recurrent convolutional neural network for object recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Sep. 2015, pp. 3367–3375.

[25] K. Jarrett, K. Kavukcuoglu, M. A. Ranzato, and Y. LeCun, "What is the best multi-stage architecture for object recognition?" in *Proc. IEEE Int. Conf. Comput. Vis.*, Sep. 2009, pp. 2146–2153.

[26] Y. LeCun, K. Kavukcuoglu, and C. Farabet, "Convolutional networks and applications in vision," in *Proc. IEEE Int. Symp. Circuits Syst.*, Jun. 2010, pp. 253–256.

[27] C. Lee, P. Gallagher, and Z. Tu, "Generalizing pooling functions in convolutional neural networks: Mixed, gated, and tree," *CoRR*, vol. abs/1509.08985, 2015. [Online]. Avaliable: https://arxiv.org/abs/1509.08985

[28] S. Li, *Markov Random Field Modeling in Image Analysis*. Berlin, Germany: Springer-Verlag, 2009.

[29] M. Lin, Q. Chen, and S. Yan, "Network in network," *CoRR*, vol. abs/1312.4400, 2013. [Online]. Available: http://arxiv.org/abs/1312.4400

[30] J. Liu, M. Gong, J. Zhao, H. Li, and L. Jiao, "Difference representation learning using stacked restricted Boltzmann machines for change detection in SAR images," *Soft Comput.*, vol. 20, no. 12, pp. 4645–4657, 2016.

[31] J. Liu, M. Gong, A. K. Qin, and P. Zhang, "A deep convolutional coupling network for change detection based on heterogeneous optical and radar images," *IEEE Trans. Neural Netw. Learn. Syst.*, to be published, doi: 10.1109/TNNLS.2016.2636227.

[32] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *Proc. Eur. Conf. Comput. Vis.*, 2014, pp. 818–833.

[33] N. Murray and F. Perronnin, "Generalized max pooling," in *Proc. IEEE Int. Conf. Comput. Vis. Pattern Recognit.*, Sep. 2014, pp. 2473–2480.

[34] V. Nair and G. Hinton, "Rectified linear units improve restricted Boltzmann machines," in *Proc. Int. Conf. Mach. Learn.*, 2010, pp. 1–8.

[35] Y. Pang, H. Zhu, X. Li, and J. Pan, "Motion blur detection with an indicator Function for surveillance robots," *IEEE Trans. Ind. Electron.*, vol. 63, no. 9, pp. 5592–5601, Sep. 2016.

[36] Y. Pang, J. Cao, and X. Li, "Learning sampling distributions for efficient object detection," *IEEE Trans. Cybern.*, vol. 47, no. 1, pp. 117–129, Jan. 2017, doi: 10.1109/TCYB.2015.2508603.

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

[38] L. Shao, D. Wu, and X. Li, "Learning deep and wide: A spectral method for learning deep networks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 25, no. 12, pp. 2303–2308, Dec. 2014.

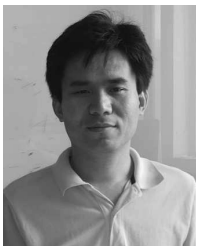
[39] M. F. Stollenga, J. Masci, F. Gomez, and J. Schmidhuber, "Deep networks with internal selective attention through feedback connections," in *Proc. Adv. Neural Inf. Process. Syst.*, 2014, pp. 3545–3553.

[40] J. Springenberg, A. Dosovitskiy, T. T. Brox, and M. Riedmiller, "Striving for simplicity: The all convolutional net," *CoRR*, vol. abs/1412.6806, 2014. [Online]. Available: http://arxiv.org/abs/1412.6806

[41] J. T. Springenberg and M. Riedmiller, "Improving deep neural networks with probabilistic maxout units," *CoRR*, vol. abs/1312.6116, 2013. [Online]. Avaliable: http://arxiv.org/abs/1312.6116

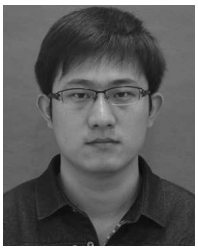
[42] C. Szegedy *et al.*, "Going deeper with convolutions," *CoRR*, vol. abs/1409.4842, 2014. [Online]. Available: http://arxiv.org/abs/1409.4842

[43] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *CoRR*, vol. abs/1409.1556, 2014. [Online]. Available: http://arxiv.org/abs/1409.1556
[44] R. K. Srivastava, K. Greff, and J. Schmidhuber, "Highway networks," *CoRR*, vol. abs/1505.00387, 2015. [Online]. Available: http://arxiv.org/abs/1505.00387
[45] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun, "Overfeat: Integrated recognition, localization and detection using convolutional networks," *CoRR*, vol. abs/1312.6229, 2013. [Online]. Available: http://arxiv.org/abs/1312.6229
[46] D. Yoo, S. Park, J. Lee, and I. Kweon, "Multi-scale pyramid pooling for deep convolutional representation," in *Proc. IEEE Workshop Comput. Vis. Pattern Recognit.*, Sep. 2015, pp. 1–5.
[47] A. Vedaldi and K. Lenc, "MatConvNet: Convolutional neural networks for MATLAB," in *Proc. ACM Multimedia*, Sep. 2015, pp. 689–692.
[48] L. Wan, M. Zeiler, S. Zhang, Y. LeCun, and R. Fergus, "Regularization of neural networks using dropconnect," in *Proc. Int. Conf. Mach. Learn.*, 2013, pp. 1058–1066.
[49] M. D. Zeiler and R. Fergus, "Stochastic pooling for regularization of deep convolutional neural networks," *CoRR*, vol. abs/1301.3557, 2013. [Online]. Available: http://arxiv.org/abs/1301.3557
[50] P. Zhang, M. Gong, L. Su, J. Liu, and Z. Li, "Change detection based on deep feature representation and mapping transformation for multi-spatial-resolution remote sensing images," *Photogram. Remote Sens.*, vol. 116, pp. 24–41, Sep. 2016.
[51] H. Zhang, M. Gong, P. Zhang, L. Su, and J. Shi, "Feature-level change detection using deep representation and feature change analysis for multi-spectral imagery," *IEEE Geosci. Remote Sens. Lett.*, vol. 13, no. 11, pp. 1666–1670, Nov. 2016.



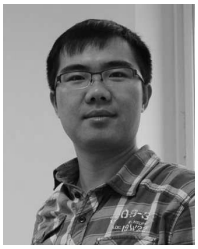
Yanwei Pang (M'07–SM'09) received the Ph.D. degree in electronic engineering from the University of Science and Technology of China, Hefei, China, in 2004.

He is currently a Professor with Tianjin University, Tianjin, China. His current research interests include deep learning, object detection, image recognition, image processing, and their applications in self-driving cars, visual surveillance, human-machine interaction, and biometrics, in which he has authored more than 100 scientific papers including 27 IEEE Transaction papers.



Manli Sun received the B.S. degree in communication engineering from Tianjin University, Tianjin, China, in 2014, where he is currently pursuing the M.S. degree.

His current research interests include deep learning and image recognition.



Xiaoheng Jiang received the B.S. and M.S. degrees in electronic engineering from Tianjin University, Tianjin, China, in 2010 and 2013, respectively, where he is currently pursuing the Ph.D. degree.

His current research interests include deep learning, object detection, and image analysis, in which he has authored five scientific papers.

Xuelong Li (M'02–SM'07–F'12) is currently a Full Professor with the Center for OPTical IMagery Analysis and Learning (OPTIMAL), State Key Laboratory of Transient Optics and Photonics, Xi'an Institute of Optics and Precision Mechanics, Chinese Academy of Sciences, Xi'an, China.