IEICE TRANS. INF. & SYST., VOL.E102–D, NO.12 DECEMBER 2019

LETTER

SDChannelNets: Extremely Small and Efficient Convolutional Neural Networks

JianNan ZHANG[†], **JiJun ZHOU**[†], **JianFeng WU**[†a)], ***Nonmembers***, ***and*** **ShengYing YANG**[†], ***Member***

SUMMARY Convolutional neural networks (CNNS) have a strong ability to understand and judge images. However, the enormous parameters and computation of CNNS have limited its application in resource-limited devices. In this letter, we used the idea of parameter sharing and dense connection to compress the parameters in the convolution kernel channel direction, thus greatly reducing the number of model parameters. On this basis, we designed Shared and Dense Channel-wise Convolutional Networks (SDChannelNets), mainly composed of Depth-wise Separable SD-Channel-wise Convolution layer. The advantage of SDChannelNets is that the number of model parameters is greatly reduced without or with little loss of accuracy. We also introduced a hyperparameter that can effectively balance the number of parameters and the accuracy of a model. We evaluated the model proposed by us through two popular image recognition tasks (CIFAR-10 and CIFAR-100). The results showed that SDChannelNets had similar accuracy to other CNNs, but the number of parameters was greatly reduced.

key words: *convolutional neural networks, parameter sharing, convolution kernel, reducing the number of model parameters*

1. Introduction

In recent years, convolutional neural networks have achieved remarkable success in the field of computer vision. Since AlexNet [7] won the championship of the Imagenet Large Scale Visual Recognition Challenge [6] in 2012, convolutional neural networks have been widely applied and studied. Later, various convolutional neural network models were proposed, such as VGG16 [4], which has 128 million parameters, and ResNet [5], which is a 152-layer convolutional neural network. As can be seen, in order to improve the accuracy, deeper and wider convolutional neural networks are designed. However, it will lead to a great increase in the number of parameters and the amount of computation. In many practical applications, models need to run on resource-limited devices, such as embedded devices and mobile phones, so more efficient models are required.

In this paper, we designed a new convolutional neural network structure for resource-limited environment. Our main contribution is to propose a novel method for channel-wise convolution. In this method, each convolutional layer has only one convolution kernel, and an input feature map is convolved with only a part of the convolution kernel each time and generates a single-channel feature map, and then multiple single-channel feature maps are connected into multi-channel feature map output. Our idea of parameter sharing and dense connection has not only reduced a large number of parameters but also enhanced the communication among the channels of feature maps. We substituted this convolution algorithm for the 1×1 convolution in MobileNetV2 [9], which greatly reduced the number of network parameters without or with little loss of accuracy.

Manuscript received June 9, 2019.
Manuscript revised August 8, 2019.
Manuscript publicized September 10, 2019.

[†]The authors are with The Institute of Electron Device and Application, Hangzhou Dianzi University, Hangzhou, China.
a) E-mail: wujianfengwz1020@163.com

DOI: 10.1587/transinf.2019EDL8120

2. Related Work

Tuning deep neural architectures to strike an optimal balance between accuracy and performance has been an active research field for the recent years [9]. Many teams have designed models with higher or similar accuracy and much fewer parameters compared with early models such as AlexNet, VGGNet, GoogLeNet [8] and ResNet.

MobileNets [10], proposed depthwise separable convolution i.e. 3×3 convolution is decomposed into 3×3 single-channel convolution and 1×1 convolution, features of each channel are extracted by single-channel convolution at first, and then fused by 1×1 convolution.

ChannelNets [11] proposed a channel-wise convolution algorithm, which replaces dense connections between feature maps with sparse connections. To be specific, channel-wise convolution is equivalent to performing one-dimensional convolution using a one-dimensional small convolution kernel in the channel direction, and then carrying out the above operation on each channel using this convolution kernel.

However, both MobileNets and ChannelNets have disadvantages. In MobileNets, the parameters of 1×1 convolution account for the majority of the total number of network parameters, reaching 74.59%. In ChannelNets, firstly, the channel-wise convolution algorithm has certain limitations in changing the number of channels, and can only maintain or reduce other than increase the number of input channels; secondly, ChannelNet-v2 only changed the last depth-wise separable convolution layer of ChannelNet-v1 into depth-wise separable channel-wise convolution layer at a cost of loss of 1% accuracy in ImageNet data set. It can be seen that sparse connection between channels of feature maps will cause certain information loss.

3. SDChannelNets Architecture

3.1 SD-Channel-Wise Convolution

First, let's start with a standard convolutional layer. The standard convolutional layer chooses a feature map F with a size of $D_f \times D_f \times m$ as an input and outputs a feature map G with a size $D_g \times D_g \times n$, where D_f is the width and height of the input feature map, m is the number of channels of the input feature map, D_g is the width and height of the output feature map, and n is the number of channels of the output feature map. The standard convolution layer also contains a convolution kernel K with a size of $D_k \times D_k \times m \times n$, where D_k is the width and height of the convolution kernel, and m and n are the previously defined number of input channels and number of output channels respectively.

The number of parameters of a convolutional layer depends on the number of parameters of the convolutional kernel, which largely depends on the size of the $m \times n$ term. Therefore, the fact that the parameters of the convolutional kernel are mutually independent in the channel direction is an important factor leading to the large number of parameters of the convolutional layer. In order to compress the size of $m \times n$ term, we proposed an SD-channel-wise convolution algorithm with parameter sharing and dense connection.

The input and output feature maps of the SD-channel-wise convolution layer are the same as the input and output feature maps of a standard convolution layer, but the SD-channel-wise convolution kernel has only one long convolutional kernel. The input feature map is convolved with only one part of the long convolutional kernel each time and this part is shared by the parameters, which replaces the convolution operation of the mutually independent convolution kernels in the standard convolution layer. To be specific, we started with a special case of $D_f = 1$ and $D_k = 1$, dotted the input feature map of $1 \times 1 \times m$ with channels $(1 + x * S) \sim (m + x * S)$ of the long convolutional kernel, and obtained an output feature map with dimensions of $1 \times 1 \times n$, where $x \in \mathrm{N}$ and $x < n$, S is the stride in the channel direction. In the case of $D_f > 1$, the above convolution operation is carried out for each position in the space.

When S is not 1, in order to keep m and n constant, the size of the convolution kernel will increase with the increase of S. Thus, in SD-channel-wise convolution, the number of parameters is:

$$D_k \cdot D_k \cdot (m + (n-1) \cdot S) \tag{1}$$

With the approach of parameter sharing, we avoided the $m \times n$ term and greatly reduced the number of parameters. It is worth noting that when the stride S in the channel direction is greater than or equal to the number m of input channels, the SD-channel-wise convolution is equivalent to the standard convolution.

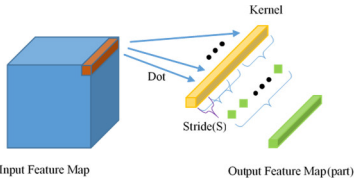


Fig. 1 1×1 SD-channel-wise convolution.

3.2 Depth-Wise Separable SD-Channel-Wise Convolution

Depth-wise separable convolution is a key building block for many efficient neural network architectures [9]. It decomposes standard convolution into depth-wise convo-lution and 1×1 convolution. The depth-wise convolution is used to filter the features of each channel, and the 1×1 convolution is used to merge these features. This idea of factorization greatly reduces the amount of computation and the number of parameters. However, 1×1 convolution still contains many parameters. In MobileNets, the number of parameters of 1×1 convolution accounts for 75% of the total number of parameters in the model. To tackle this problem, we replaced 1×1 convolution with 1×1 SD-channel-wise convolution. The 1×1 SD-channel-wise convolution is shown in Fig. 1.

The number of parameters in the 1×1 convolution:

$$1 \cdot 1 \cdot m \cdot n \tag{2}$$

The number of parameters in the 1×1 SD-channel-wise convolution:

$$1 \cdot 1 \cdot (m + (n-1) \cdot S) \tag{3}$$

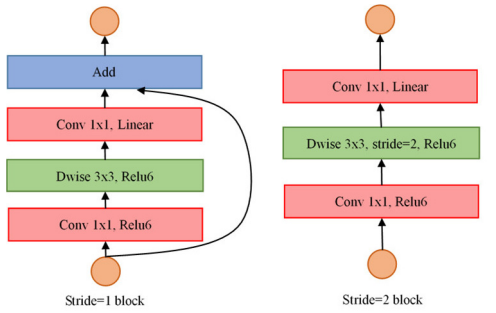
With the approach of parameter sharing, we reduced the number of parameters:

$$\frac{1 \cdot 1 \cdot (m + (n-1) \cdot S)}{1 \cdot 1 \cdot m \cdot n} \approx \frac{1}{n} + \frac{S}{m}$$

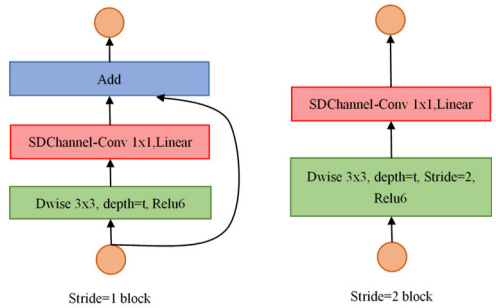
The model size and accuracy are balanced by changing the stride S in the channel direction. In the extreme case of $S = 1$, the number of parameters of the 1×1 SD-channel-wise convolution is tens to hundreds of times smaller than the number of parameters of the 1×1 convolution.

3.3 SDChannelNets

SDChannelNets are constructed using depth-wise separable SD-channel-wise convolutions. We followed the basic architecture of MobileNetV2 and made a fair comparison with MobileNetV2. Firstly, in order to minimize computa-tions, we removed the first 1×1 convolution from the bottleneck block of MobileNetV2. Then, in order to keep the number of channels consistent with it, we set the depth multiplier of the depth-wise convolution as 6 or 1. Lastly, we replaced the second 1×1 convolution in the bottleneck blocks with 1×1 SD-channel-wise convolution. After the above modification, the bottleneck block constituted the SDC-bottleneck block in SDChannelNet. The input and output sizes of the



(a)MobileNetV2



(b) SDChannelNet

Fig. 2 Illustration of SDC-bottleneck block and comparison with the bottleneck block of MobileNetV2.

SDC-bottleneck block were exactly the same as the bottleneck block. The difference between the SDC-bottleneck block and the bottleneck block is shown in Fig. 2. At the same time, we also noticed that the number of parameters in the dense connection layer of MobileNets accounted for 24.3% of the total number of parameters, so the convolution layer with 1280 output channels was removed. After the removal, the number of parameters and computation amount of the dense connection layer of the model were only 1/4 of the original. It is worth mentioning that when the basic features are fewer and activated by a nonlinear activation function, more basic features will be lost, resulting in fewer available features. Therefore, we removed the activation function Relu behind the first convolutional layer. This operation raised the accuracy of our model on the CIFAR-10 dataset by about 0.5%. The SDChannelNets structure is shown in Table 1.

3.4 Model Expanding Hyperparameter

We chose the stride S of the SD-channel-wise convolution in the channel direction as a hyperparameter, which can be adjusted according to the required accuracy and number of parameters. Therefore, we designed three models, SDChannelNet-S1, SDChannelNet-S64 and SDChannelNet-S192, with a stride in the channel direction being 1, 64 and 192 respectively. Their total number of parameters ranged from 0.43 million to 0.7 million, about 12.6% to 20.6% of MobileNetV2, and the computation was 150 multiply-adds, 50% of MobileNetV2. It is worth noting that the reduction of computation is entirely due to the reduction of MobileNetV2, and the SD-channel-wise convolution only reduces parameters without reducing computation.

Table 1 SDChannelNet: each row describes a sequence composed of one or more identical layers (blocks), which is repeated for n times. All layers in the same sequence have the same number of c output channels. The first layer of each sequence has a stride s, and all other sequences use a stride 1. All convolutions use 3×3 convolution kernels. t denotes a depth multiplier of the depth-wise convolution. The above structure takes reference from MobileNetV2.

Input	Operator	t	c	n	s
$224^2 \times 3$	Conv2d	-	32	1	2
$112^2 \times 32$	SDC-bottleneck	1	16	1	1
$112^2 \times 16$	SDC-bottleneck	6	24	2	2
$56^2 \times 24$	SDC-bottleneck	6	32	3	2
$28^2 \times 32$	SDC-bottleneck	6	64	4	2
$14^2 \times 64$	SDC-bottleneck	6	96	3	1
$14^2 \times 96$	SDC-bottleneck	6	160	3	2
$7^2 \times 160$	SDC-bottleneck	6	320	1	1
$7^2 \times 320$	Avgpool7x7	-	-	1	-
$1 \times 1 \times 320$	Conv2d1x1	-	k	-	-

4. Experiments

We evaluated our model on CIFAR [13] dataset and compared it with other structural models and parameter reduction methods, such as pruning [17] and knowledge distillation [18].

4.1 Datasets

Both CIFAR-10 and CIFAR-100 are composed of 32×32 color images. Their training set and test set contained 50,000 images and 10,000 images respectively. The difference between them was that the images of CIFAR-10 were classified into 10 categories, while the images of CIFAR-100 were classified into 100 categories. For preprocessing, we followed [15]. We trained with all the training images and reported the error rate on the test set at the end of the training.

4.2 Training

Following [15], all networks were trained by stochastic gradient descent and used a weight decay of 10^{-4} and a Nesterov momentum of 0.9 [14]. In CIFAR, the batch size we used for training was 64 and 250 epoch. The initial learning rate was set as 0.1, and the learning rate was divided by 10 at 60% and 80% of the total training epochs. We added BN [2] layer and Dropout [3] layer behind each convolution layer, except for the first layer.

4.3 Classification Results on CIFAR

We trained SDChannelNets on CIFAR dataset using different channel strides, and compared them with other CNNs and parameter reduction methods in terms of error rate and number of parameters. The results are shown in Table 2.

Since the image size of the CIFAR dataset was $32 \times$

Table 2 Comparison between SDChannelNets and other CNNs in terms of the error rates(%) and the number of total parameters on the CIFAR-10 and CIFAR-100 test set. ∗ indicates results run by ourselves. + indicates error with data augmentation.

Dataset	CIFAR-10			CIFAR-100		
Method	Params	Error(%)	Error+(%)	Params	Error(%)	Error+(%)
Network in Network [12]	-	10.41	8.81	-	35.68	-
FractalNet [16]	38.6M	10.18	5.22	38.6M	35.34	23.30
ResNet (reported by [1])	1.7M	13.63	6.41	1.7M	44.74	27.22
ResNet with Stochastic Depth [1]	1.7M	11.66	5.23	1.7M	37.80	24.58
ResNet-164(pre-activation)	1.7M	11.26	5.46	1.7M	35.58	24.33
ResNet-1001(pre-activation) (reported by [15])	10.2M	10.56	4.62	10.2M	33.47	22.71
MobileNetV2 [9]	2.3M	10.60*	4.87*	2.6M	**31.36***	**21.92***
ChannelNet-v2 [11]	1.6M	11.03*	5.74*	1.7M	33.21*	23.89*
ResNet-18 (pruned) [17]	0.49M	-	5.56	-	-	-
KSANC [18]	0.27M	-	7.32	0.27M	-	31.42
SDChannelNet-S1	0.11M	12.59	6.37	0.14M	41.23	29.88
SDChannelNet-S64	0.21M	9.92	5.02	0.24M	34.65	24.74
SDChannelNet-S192	0.38M	**8.84**	**4.54**	0.41M	32.47	22.57

32, in order to enable SDChannelNets to better adapt to this input size, we changed the stride of the first three blocks or layers with a stride of 2 to 1. That is to say, the input size of the feature map of the average pooling layer is 8×8. To compete fairly with MobileNetV2 and ChannelNet-v2, we made the same change to them, too.

CIFAR-10. In Table 2, we can notice that the SDChannelNet-S1 has the fewest parameters only 0.11 million, decreasing by 95.2% compared with MobileNetV2, and shows certain performance, thanks to SD-channel-wise convolutions. When we set the channel stride of SDChannelNets as 64, the number of parameters of SDChannelnet-S64 was still small, only 0.21 million, 90.9% less than the number of parameter of MobileNetV2. Meanwhile, the error rate was 0.68% less than that of MobileNetV2 without data augmentation. In addition, the error rate of SDChannelNet-S192 was lower than those of other CNNs and parameter reduction methods in Table 2.

CIFAR-100. In Table 2, although the error rate of SDChannelNet-S192 was 1.11% higher than that of MobileNetV2, the number of parameters of SDChannelNet-S192 was only 0.41 million, 84.2% lower than that of MobileNetV2. Moreover, the results of SDChannelNet-S192 were better than those of other CNNs and parameter reduction methods except MobileNetV2 in Table 2. This confirms the superiority of our convolutional compression method.

4.4 Parameters Accuracy Trade-Off

As shown in Table 2, the positive correlation between the number of parameters SDChannelNets have and the accuracy they achieve. It can be seen that increasing the stride in the channel direction is an effective method to improve the model capability, and enables the model to achieve a good compromise between the number of parameters and the capability of the model.

5. Conclusion and Future Work

In this letter, we proposed a novel convolution algorithm, i.e., SD-Channel-Wise Convolution to replace traditional convolution and reduce the number of model parameters. We designed and used the Depth-Wise Separable SD-Channel-Wise Convolutions to build a compact and effective convolution neural network, called SDChannel-Nets. Then, we used the stride of the SD-Channel-Wise Convolutions as a hyperparameter to balance the accuracy and size of the model. Finally, compared with other CNNs and parameter reduction methods in experiments, SDChannelNets had fewer parameters and a better identification effect, which verified its compact and effective characteristics. We plan to combine SD-Channel-Wise Convolutions with other parameter reduction methods in the future work to explore a smaller and faster convolution neural network.

References

[1] G. Huang, Y. Sun, Z. Liu, D. Sedra, and K.Q. Weinberger, "Deep networks with stochastic depth," European Conference on Computer Vision (ECCV), 2016.

[2] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," International Conference on Machine Learning (ICML), 2015.

[3] N. Srivastava, G.E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," Journal of Machine Learning Research, pp.1929–1958, 2014.

[4] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A.C. Berg, and L. Fei-Fei, "Imagenet large scale visual recognition challenge," International Journal of Computer Vision (IJCV), vol.115, no.3, pp.211–252, 2015.

[5] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp.770–778, 2016.

[6] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: a Large-Scale Hierarchical Image Database," 2009 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR), pp.248–255, 2009.

[7] A. Krizhevsky, I. Sutskever, and G.E. Hinton, "Imagenet classification with deep convolutional neural networks," Conference and Workshop on Neural Information Processing Systems (NIPS), 2012.

[8] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," Proc. Comput. vis. pattern recognit., pp.1–9, June 2015.

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

[10] A.G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," arXiv preprint: 1704.04861, 2017.

[11] H. Gao, Z. Wang, and S. Ji, "ChannelNets: Compact and efficient convolutional neural networks via channel-wise convolutions," Conference and Workshop on Neural Information Processing Systems (NIPS), 2018.

[12] C.Y. Lee, S. Xie, P. Gallagher, Z. Zhang, and Z. Tu, "Deeply-supervised nets," AISTATS, 2015.

[13] A. Krizhevsky, "Learning multiple layers of features from tiny images," Technical Report, University of Toronto, 2009.

[14] I. Sutskever, J. Martens, G. Dahl, and G. Hinton, "On the importance of initialization and momentum in deep learning," International Conference on Machine Learning (ICML), 2013.

[15] G. Huang, Z. Liu, L.V.D. Maaten, and K.Q. Weinberger, "Densely Connected Convolutional Networks," IEEE Conference on Computer Vision & Pattern Recognition, pp.2261–2269, 2017.
[16] G. Larsson, M. Maire, and G. Shakhnarovich, "Fractalnet: Ultra-deep neural networks without residuals," arXiv preprint: 1605.07648, 2016.
[17] C. Lin, Z. Zhong, W. Wu, and J. Yan, "Synaptic strength for convolutional neural network," arXiv preprint: 1811.02454, 2018.
[18] C. Shu, P. Li, Y. Xie, Y. Qu, L. Dai, and L. Ma, "Knowledge squeezed adversarial network compression," arXiv preprint: 1904.05100, 2019.